ABSTRACT

Title of Dissertation: COMPUTATIONAL METHODS FOR NATURAL WALKING IN VIRTUAL REALITY

Niall L. Williams
Doctor of Philosophy, 2024

Dissertation Directed by: Professor Dinesh Manocha
Department of Computer Science

Virtual reality (VR) allow users to feel as though they are really present in a computer-generated virtual environment (VE). A key component of an immersive virtual experience is the ability to interact with the VE, which includes the ability to explore the virtual environment. Exploration of VEs is usually not straightforward since the virtual environment is usually shaped differently than the user's physical environment. This can cause users to walk on virtual routes that correspond to physical routes that are obstructed by unseen physical objects or boundaries of the tracked physical space.

In this dissertation, we develop new algorithms to understand how and enable people to explore large VEs using natural walking while incurring fewer collisions with physical objects in their surroundings. Our methods leverage concepts of alignment between the physical and virtual spaces, robot motion planning, and statistical models of human visual perception. Through a series of user studies and simulations, we show that our algorithms enable users to explore large VEs with fewer collisions, allow us to predict the navigability of a pair of environments without collecting any locomotion data, and deepen our understanding of how human perception functions during locomotion in VR.

COMPUTATIONAL METHODS FOR NATURAL WALKING IN VIRTUAL REALITY

by

Niall L. Williams

Dissertation submitted to the Faculty of the Graduate School of the
University of Maryland, College Park in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
2024

Advisory Committee:
Professor Dinesh Manocha, Chair/Advisor
Professor Jae Shim, Dean's Representative
Professor Aniket Bera
Professor Ming C. Lin
Professor Huaishu Peng

To all of my teachers and mentors.

Acknowledgments

Access to education is an extreme privilege, so I am extraordinarily grateful to the systems in place that allowed me to spend 5.5 years of my life essentially studying whatever I find interesting. Hopefully the research produced from this dissertation will find its place in contributing back to society one day.

I thank my PhD advisors/mentors, Dr. Dinesh Manocha, Dr. Aniket Bera, and Dr. Ming C. Lin, for trusting me enough to let me study a new topic that nobody in our lab was working on. Your flexibility in which topics I could study has made the PhD a very fulfilling and enjoyable experience. I especially thank Dr. Manocha for giving me the freedom to study whatever I happened to find interesting at the time, Dr. Bera for meeting with me at all hours of the day to discuss the low-level details that I got stuck on, and for Dr. Lin's infectious enthusiasm for VR locomotion and research in general. I also thank the rest of my committee, Dr. Huaishu Peng and Dr. Jae Kun Shim, for providing their outside perspectives and helpful feedback on this dissertation's work.

Outside of my PhD committee, there are many other teachers and mentors whom I must thank. I owe the greatest thanks to Dr. Tabitha C. Peck for formally introducing me to computer graphics and research, for always challenging and encouraging me, for patiently answering my career and life questions for the last 7+ years, and for teaching me the importance of staying positive and believing in myself. I would not have made it this far without your support. I also

thank Mary C. Whitton for giving me feedback on the first paper that I submitted for peer review, for providing words of encouragement and positivity, for helping me realize the importance of history, and for always chatting with me about life and research at conferences. Thanks to Dr. Kerry McIntyre Magee for providing me with an outstanding education at what I view as the start of my journey as a scientist, for explaining to me what a PhD even is, for teaching me how to understand a complex system as the sum of individual components working together, for giving me exam questions that challenged my logic and reasoning skills, and for teaching me to always label my axes! Thanks to Ashok Pillai and the Davidson Computer Science & Mathematics faculty for introducing me to computer science and providing me with the foundational skills needed to become a strong programmer. I thank Karl Savoury for teaching me how to write a proof, for getting me hooked on geometry, for challenging me with new math problems that I did not understand, and for being the first teacher to treat me like a peer and a collaborator instead of a subordinate student. Thanks to Joan Cogle, Jane Harper, Jai Oshun, Cherry Robinson, and Rosie Smalling for teaching me how to organize my work, manage my time, and develop my public speaking skills. I also thank Flloyd Logan for teaching me the importance of discipline and respecting other people's time.

I am also very grateful to my mentors from my internships in industry research labs, who opened my eyes to a whole new way to think about perception and computer graphics. Thanks to Dr. Ian Erkelens, Dr. Phillip Guan, and the rest of the Applied Perception Science team at Meta Reality Labs for showing me what vision science is and for teaching me the full value of conducting precise, well-controlled experiments. Thanks to Dr. Ruth Rosenholtz at NVIDIA, who taught me a great deal about "smart models" of vision, how to choose my wording *very* carefully, how to think carefully about which component of perception an experiment is *actually*

measuring, and for emphasizing the value of developing computational, predictive models of human vision. I wish I had the chance to work with you earlier in my career so that your mentorship could have had a greater influence on the work in this dissertation.

I would be remiss not to thank my many friends, old and new, who made the PhD even more fun than it already was. Thanks to my GAMMA lab mates Jaehoon Choi, Vishnu Sashank Dorbala, Jason Fotso-Puepi, Alexander Gao, Tianrui Guan, Pooja Guhan, Divya Kothandaraman, Geonsun Lee, Yonghan Lee, Bhrij Patel, Yiling Qiao, Shreelekha Shriram Revankar, Logan Stevens, Xijun Wang, Ruiqi Xian, Laura Zheng, and the rest of the lab for all the advice, fun times, and insightful research discussions. I also thank my other friends in the CS department, including Yusuf Alnawakhtha, Connor Baumler, Dr. Marina Knittel, Jiasheng Li, Pedro Sandoval, Manasi Shingane, Zeyu Yan, and (honorary GAMMA member) Kevin Zhang. I thank my friends from outside UMD, including Zubin Choudhary, Dr. Shakiba Davari, Alexander Giovanelli, Matt Gottsacker, Dr. Ryan Hamilton, Dr. Saad Hassan, Dr. SeulAh Kim, Dr. Lee Lisle, Enrique Melendez, Dr. Cassidy R. Nelson, Dr. Missie Smith, Dr. Ashutosh Srivastava, and Dr. Zoe (Jing) Xu for the fun adventures and great company at conferences and internships. A big thanks also goes out to my friends from college and high school, including Dr. Christopher Brooks, George Cai, Arthur Chen, My Doan, Ryan Ewing, Walker Griggs, Yeonjae Han, Sarah Hancock, Hoot Hennesy, Mingyu Kim, Ben Kuschner, Josh Kuschner, Jaeyoung Lee, Dr. Lillian Lowrey, James Ni, Khoa Phan, Jimmy Plaut, Tom Ren, Ryan Strauss, and Terry Zervos.

Finally, I am very grateful for the administrative and financial support I have received. Thanks to Migo Gui, Jodie Gray, Tom Hurst, and the rest of the CS department administration for helping with research and course logistics. I also thank the Link Foundation for graciously funding part of my PhD via the Modeling, Simulation, & Training Fellowship.

"I don't mistrust reality, of which I know next to nothing. I mistrust the picture of reality conveyed to us by our senses, which is imperfect and circumscribed."

– Gerhard Richter, 1972

Table of Contents

List of Tables

List of Figures

Part I

Introduction & Background

Chapter 1: Overview

Virtual reality (VR) is a system that allows the user to experience and interact with a computer-generated digital environment [101]. VR is an interesting technology because it creates a feeling of *presence*, the feeling that the user is really in the virtual environment (VE) that they are viewing [178]. Key to this feeling that one is actually *in* the VE is the ability to interact with the VE—the user's actions in VR create a change in their experience of the virtual surroundings. For example, picking up a virtual basketball and throwing it should create a plausible dynamic response that causes the ball to bounce away according to the laws of physics and the geometry of the virtual objects that make up the VE.

One important aspect of interaction is the ability to *explore* the VE. The ability to actively explore an environment (i.e., using one's own volition) leads to better acquisition of route and survey knowledge in the environment [36]. Furthermore, it has been shown that the ability to explore an environment using natural walking contributes significantly to a user's feeling of presence in VR [209] and their performance on search and wayfinding tasks [84, 162]. Here, we define "natural walking" as step-driven locomotion that does not use treadmills or other mechanical devices, makes use of the entire gait cycle, and, ideally, is perceived by the user as identical to how they walk in the real world while not in VR [116, 188]. However, a fundamental problem with natural walking in VR is that the user's physical movements are usually mapped

one-to-one to their virtual movements, which means that an unobstructed path in the virtual world may correspond to an *obstructed* path in the physical environment (see Figure 1.1).

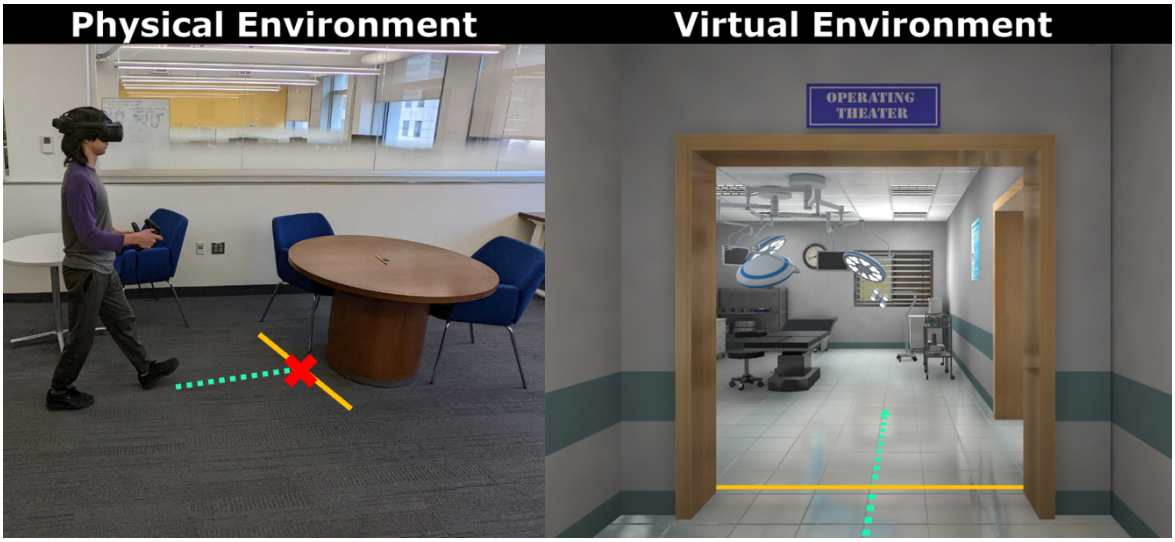


Figure 1.1: An example of the VR locomotion problem. The user wishes to travel in a straight line in the direction they are facing (dashed teal line). In the virtual environment (right), this corresponds to a valid path that takes the user into the virtual operation room. However, in the physical environment (left), this desired path is invalid because it yields a collision with the table front of them. Locomotion interfaces avoid these collisions by allowing the user to move through the virtual environment without requiring them to carry out the same movements in the physical space.

Locomotion interfaces (LIs) are techniques that mitigate this problem by changing the mapping between the user's physical and virtual movements or otherwise allowing users to control their virtual movements (e.g., via a joystick) [52, 188]. Indeed, researchers have invested a significant amount of work into developing LIs that are comfortable, easy to use, and effective for moving the user through the VE. Each interface comes with different advantages and disadvantages in terms of its learning curve, the level of presence afforded, and its efficiency. In this dissertation, we focus on locomotion interfaces that allow users to explore VEs using natural walking due to the benefits it provides to the user experience. Most of the research in natural walking interfaces for VR build upon a technique known as redirected walking (RDW) [157, 158], which

works by subtly manipulating the mapping between the user's physical and virtual movements which allows the VR system to steer the user away from physical obstacles in their surroundings that they cannot see. There are three main questions that the majority of natural walking (and especially RDW) research has studied:

1. *How severely can we manipulate the mapping between a user's physical and virtual motions before the discrepancy in their self-motion signals becomes detrimental to their virtual experience?*

2. *What is the optimal direction to steer a user in the physical environment to avoid as many collisions with physical obstacles as possible?*

3. *For a given pair of physical and virtual environments, what is the optimal natural walking interface that yields the best user experience?*

Note that this third question has received comparatively much less attention than the first two questions.

In this dissertation, we develop methods that make progress on all three of the above questions. We focus on *computational* approaches to these questions since computational methods have many benefits for modeling, simulating, and understanding complex systems and large amounts of data [90, 171] For example, by redefining the redirection problem using a rigorous mathematical framework (Chapter 4), our RDW algorithms can be more readily applied to a wide range of scenarios as long as the correct inputs (e.g., walking trajectories, environment layouts, human factors) can be fed into our algorithms. Indeed, prior work has demonstrated that redirection (i.e., simultaneous, decoupled locomotion in a real and virtual environment) exhibits complex behaviors akin to chaotic systems [81] and has shown the usefulness of simulations for

measuring the efficacy of different redirection algorithms [10]. In an effort to develop computational methods for VR locomotion, we build upon the concept of *alignment* [207] as a way of comparing the similarity of the structure of the physical and virtual environments. To achieve this, we introduce environment similarity metrics that are precisely defined based on the geometric structure of the environments. We also adapt techniques from robot motion planning to define a rigorous mathematical framework that allows us to reason about VR locomotion in generalized, abstract terms. This framework enables us to develop locomotion interfaces that can operate in any environment as long as particular constraints are fulfilled. Furthermore, we show that this framework allows us to better understand and make predictions about locomotion in a pair of physical and virtual environments *without* the need to run user studies to collect locomotion data. Additionally, we measure the correlation between a user's physiological signals (gaze and postural stability) and the intensity of VE manipulations introduced by redirected walking (RDW), a popular natural locomotion interface. To evaluate my methods, we use both quantitative and qualitative metrics. This includes statistical models of human perception, quantifications of distance walked and frequency of collisions in VR, presence and simulator sickness questionnaires, and semi-structured interviews with participants.

1.1 Main Contributions

The main contributions of this thesis are:

1. **Improved redirected walking algorithms based on alignment and distractors:** We develop two new algorithms that steer the user away from unseen physical obstacles with a higher success rate than existing state-of-the-art algorithms. Existing algorithms typically

do not consider the layout of the virtual environment when deciding where to steer the user in the physical space. We show that by considering the structure of both the physical *and* virtual environments together, one can develop a new kind of redirection algorithm that can avoid more collisions than algorithms that do not consider the structure of the virtual environment, in both static and dynamic scenes. Our algorithms leverage *alignment*, the concept of measuring the similarity of the user's state (in our case, proximity to objects) in the physical and virtual environments, to approximate the likelihood that the user will collide with a physical object. Once an alignment score has been computed, we use this metric to steer the user to a safer physical location that minimizes the discrepancy between the user's proximity to physical and virtual objects.

Next, we develop a third algorithm that formalizes the usage of *distractors*, any element of the virtual environment that aims to capture the user's attention [148], for improved collision-avoidance during VR locomotion. We describe a framework for how distractors can be implemented into any natural-walking locomotion interface as long as the VR system can compute regions of the physical space that the user can safely navigate towards and a feasible distractor behavior can be generated that guides the user to that safe physical region. We demonstrate the viability of this framework through a simple implementation and study the effects of distractor behavior on the effectiveness of this framework.

2. **A novel, rigorous formulation of the redirected walking problem:** Many redirection algorithms have been based on simple, hand-designed heuristics based on intuitions about the situations in VR walking that are typically challenging (e.g., small physical spaces). This decision to use heuristics limits an algorithm's ability to perform well across a wide

range of physical and virtual environments since it is likely that the chosen heuristics do not cover the vast range of possible environments and configurations the user may be in. Taking inspiration from robot motion planning, we reformulated the redirected walking problem in terms of the user's configuration in either environment and their trajectory through the virtual environment. We then show how this framework highlights geometric and perceptual constraints that tend to make collision-free navigation difficult. Two of the redirection algorithms introduced in this thesis are based on this mathematical framework.

3. **A new metric to quantify navigability for virtual reality:** A large challenge with VR locomotion is that it is difficult to know how much collision-free navigation is possible for a given physical and virtual environment without conducting a user study. This is because quantifying the navigability of a pair of environments usually requires gaining an understanding of the types of virtual paths the user is likely to travel on, which determines how feasible it is for the user to safely navigate through their corresponding physical environment. We introduce, for the first time, a metric that approximates the navigability of a given pair of physical and virtual environments *without* using any user locomotion data (real or simulated). Our metric is based purely on the geometric layout of the two environments. Our metric is built on the observation that locomotion is a primarily *local* problem and that by quantifying the similarity of the local structure of uniformly-sampled points across the two environments, we can approximate the likelihood that a user will incur a collision for any given starting positions in the physical and virtual environments. We validate our metric using large-scale simulations and find that our metric is correlated with the navigability of environment pairs.

4. **A new study of the relationships between redirection and physiological signals:** Redirected walking works by injecting small, subtle rotations and translations into the user's virtual camera trajectory as they move around in their physical space. An important component of successfully deploying a redirected walking system is ensuring that these rotations and translations are never too large that the user consciously notices them. If the user notices the applied redirection, it is likely that it will interfere with the quality of their virtual experience and may make them experience simulator sickness [104]. Traditional methods for estimating a user's sensitivity to redirection employ psychophysical threshold estimation techniques which are not scalable since they require large amounts of time, can cause users to feel fatigued or bored, and do not generalize well to the broader population (i.e., there are individual differences in sensitivity). We conduct a study that measures perceptual thresholds for RDW rotation gains and examines the relationship between the strength of these gains and physiological signals generated by the user. In particular, we show that the strength of the gain is correlated with the stability of the user's gaze and posture, which opens the door for new methods of RDW sensitivity estimation that do not require long calibration times like traditional psychophysical methods do.

1.2 Overview of Dissertation

This thesis presents new methods for understanding locomotion in virtual reality and for developing natural walking-based locomotion interfaces. We organize the thesis as follows:

- **Chapter 2—Background**: We provide a high-level overview of the human perceptual system and locomotion interfaces for virtual reality. In particular, we detail the basic

mechanics of how visual perception works and how it contributes to a user's perception of self-motion through their environment. We also provide details on the basics of locomotion interfaces for virtual reality, with a focus on methods that enable users to explore their virtual surroundings using natural walking.

- **Chapter 3—Alignment-Based Redirection:** The first contribution of this thesis is based on the observation that collision-free locomotion in VR does *not* require the physical and virtual environments to have globally similar layouts. That is, regions of local similarity can yield collision-free paths as long as the user's proximity to objects is similar between the physical and virtual environments. Furthermore, this similarity in proximity is only necessary in the direction of travel. Based on this observation, we designed a redirection algorithm that first quantifies the *alignment* of the user's state in each environment (i.e., the similarity of the user's physical and virtual positions in terms of proximity to objects) and then applies redirection to steer the user towards a physical location that is more similar to their virtual location. Interestingly, we find that by applying redirection only proportional to the magnitude of the differences in proximity to physical and virtual objects, we are able to achieve improved collision avoidance while also steering the user with *weaker* redirection gains compared to state-of-the-art RDW algorithms, which may decrease the chance that the user feels symptoms of simulator sickness. This finding goes against a common rule-of-thumb that stronger redirection gains yield better collision-avoidance performance.

- **Chapter 4—Visibility-Based Redirection:** Next, we present a novel, mathematical formulation of the RDW problem based on concepts *robot motion planning*. In particular, we detail how

the redirection problem can be described using the notions of configuration spaces (which describe the user's position and orientation in an environment) and trajectories (which are represented as an ordered set of user configurations in an environment). Building upon this framework, we use *visibility polygons* to represent the local structure of the environment around the user's current position. We show that this representation is useful for computing regions of space where a collision is possible *without* needing to know what the user's future trajectory is, meaning no trajectory prediction is required. Using our mathematical framework and visibility polygons, we develop a new redirection algorithm that achieves improved results over state-of-the-art algorithms (including our alignment-based algorithm described in Chapter 3) in both static and dynamic environments.

- **Chapter 5—Distractor-Based Redirection:** In this chapter, we present a natural walking interface for VR that integrates *distractors*, elements of the virtual environment that capture the user's attention [148], to help guide the user away from collisions with physical objects. Since VR is an interactive technology, users often interact with elements of their virtual surroundings (usually things like virtual agents or objects) as part of their virtual experience. Based on this observation, we designed a locomotion framework that directly uses distractors as a guiding agent to steer the user away from imminent physical collisions in a naturalistic way that does not interfere with their virtual experience (either through overt reorientation interventions or motions injected into the virtual camera). Our framework functions by computing *safe zones* of the physical environment, which are regions of the physical space that the user is able to safely navigate to and are not likely to lead to a collision in the near future. Once an appropriate safe zone has been computed and the user interacts with

a virtual distractor, our system generates a distractor behavior that is natural (within the context of the virtual experience) and will guide the user towards the safe zone as long as the user continues to interact with the virtual distractor. We implement a simplified version of our framework and demonstrate its effectiveness compared to a redirection system with naïve distractor behavior. Furthermore, we study how changes to the distractors' behavior impact our ability to guide the user away from collisions with physical objects.

- **Chapter 6—Quantifying Environment Navigability for Natural Walking in Virtual Reality:** One challenge for researchers who study virtual reality locomotion navigation is that it is difficult to predict how easily a user will be able to explore a given virtual environment without conducting a user study. In this chapter, we develop, for the first time, a metric that approximates the navigability of a pair of physical and virtual environments purely based on the geometric layout of the environments (i.e., our metric does not require users' navigation trajectories as an input). Our metric is based on the observation that natural walking behavior is largely determined by the structure of the user's local surroundings. Therefore, a method that can sample and quantify the similarity of locations in the physical and virtual environments will likely be correlated with the ease of collision-free locomotion in those environments. We present details on how such a metric can be computed, taking inspiration from geometric shape similarity metrics and robot navigation, and show through extensive user studies and simulations that our metric is correlated with how far a user is able to walk in an environment before they incur a collision with an unseen physical obstacle.

- **Chapter 7—Perceptual Sensitivity and Physiological Signals of Tolerance to Redirection:**

When studying redirected walking systems it is important to consider not only the design of redirection algorithms but also the user's subjective perceptual sensitivity to the rotations and translations that RDW introduces into their virtual camera motion. A crucial factor to consider when deploying a RDW algorithm is to make sure that the algorithm does not apply redirection gains that are too strong for the user such that they notice the injected motions and feel symptoms of simulator sickness, which will detract from their virtual experience. Traditional methods to estimate this sensitivity to injected motions are based on psychophysics and often entail long measurement processes that are tiring for the user and cannot be employed while the user is in a virtual experience (e.g., while the user is experience a virtual job simulator). In this chapter, we conduct a study that measures sensitivity to RDW rotation gains under light and dark conditions and correlates users' sensitivity to patterns in their physiological signals. In particular, we show that increased rotation gains are positively correlated with postural and gaze instability. This finding opens the door to the possibility of using physiological signals as a measure of user comfort during redirected locomotion, potentially bypassing the long measurement process required by traditional psychophysical methods.

- **Chapter 8—Conclusion, Limitations, and Future Work:** In this chapter, we provide a summary of the results, discuss its limitations, and discuss avenues for future work in this area. The overall goal with this thesis is to improve our understanding of the dynamics of natural walking in virtual reality and develop algorithms and metrics that improve users' ability to explore virtual environments using natural walking. In an effort to do this, we introduced new algorithms for natural walking in VR, developed new mathematical tools to

think about and analyze virtual locomotion, and provided new insight into the relationship between perception and physiology during virtual locomotion. However, our work has limitations relating to the usage of simulation-based methods and how likely some of our results are to generalize to real users, the computational costs of our navigability metric, and the generalizable of the results on physiology to more representative natural walking scenarios and tasks. Future work in this area should aim to improve upon these limitations by building more complex models of human locomotion and perception, developing data-driven methods for estimating environment navigability, and conducting large-scale user studies to better understand the success and failure cases of our locomotion interfaces.

Chapter 2: Background

In this section, we provide a high-level overview of the three main areas of research that this dissertation builds upon. We discuss how human perception contributes to a person's ability to understand and interact with their environment (Section 2.1), the interfaces that have been developed to enable users to explore VEs (Section 2.2), and the mathematical framework that roboticists have created to allow them to develop rigorous robot navigation algorithms (Section 2.3).

2.1 Human Perception

The human perceptual system is responsible for organizing, identifying, and interpreting the sensory information that is received by an observer's sensory system [237]. Virtual reality experiences are highly influenced by the user's perceptual system. When in VR, the user perceives stimuli that mostly come from the VR system. That is, VR system developers have direct control over a large portion of the information perceived by users. Because of this, it is important for us to understand how users react to different perceived stimuli so that we can create enjoyable and effective virtual experiences for our users.

Some examples of different types of stimuli include visual, auditory, haptic, proprioceptive, and vestibular signals, each of which is processed by a respective perceptual system of an observer. During the process of perception, an observer's brain must integrate the information from all

of these different perceptual systems so that they can come to a conclusion about the state of their surroundings. This is a process known as multisensory integration [186]. In VR, it is not uncommon for the information about the user's surroundings from different perceptual systems to disagree. For example, a user playing a game in VR might see stimuli that correspond to a jungle environment but might simultaneously be overhearing sounds of a cooking appliance in the kitchen nearby, which provides auditory information that contradicts the visual indicator that the user is in a jungle. This is a situation known as sensory conflict, and it can decrease a user's feeling of presence within a virtual experience.

2.1.1 Visual Perception

Although human perception is a multisensory experience, the rest of this dissertation focuses primarily on visual perception and its intersection with locomotion since visual stimuli are usually the primary channel through which a user experiences VR. Indeed, one of the primary techniques that this work uses, called redirected walking (RDW), only works because humans tend to respond more often to visual stimuli than non-visual stimuli (a phenomenon known as visual dominance [152]). However, as we will briefly discuss in Section 7.5, a multisensory view of locomotion in VR will likely be necessary in order to make notable progress in understanding human locomotion in VR.

Visual perception refers to the brain's interpretation of an environment through the eyes. It is an important part of how observers understand their surroundings. Within VR, the visual stimuli a user perceives come from the head-mounted display (HMD). The quality of the stimuli will depend on HMD factors including refresh rate, display resolution, and field of view (FOV).

FOV is the observable space an observer can see through their eyes or viewing device. FOV is of particular interest to us in this thesis, since differences in FOV have been shown to influence observers' locomotion patterns. Visual perception is crucial to virtual experiences, so we will now discuss some of the important facets of visual perception and how they interact with locomotion.

2.1.1.1 Optical Flow

Optical flow refers to the pattern of perceived motion of the surrounding environment that is projected onto the human observer's retina. Optical flow patterns serve as a visual signal of self-motion for the human observer. Numerous studies have shown that optical flow influences the observer's locomotion control depending on the speed and direction of optical flow [13, 145, 221]. When the observer's non-visual movement signals conflict with their visual movement signals (namely optical flow), the brain prioritizes the visual signals. That is, when the observer determines their current motion, they are more likely to believe visual information than non-visual information if the two provide conflicting cues of self-motion [16, 114].

2.1.1.2 Vection

Vection is the illusory impression of self-movement provided by visual stimulation [78, 188]. It is typically felt when the observer visually perceives a moving environment, but their body moves in a manner that would not produce the perceived optical flow patterns. Because vection is most often induced by visual stimuli, it is closely tied to the perceived optical flow. A common example of vection is the feeling of movement when an observer sits stationary in a train and watches a neighboring train move.

It is known that peripheral stimulation plays an important role in perceiving optical flow patterns [145]. Thus, we can infer that peripheral stimulation, which VR provides a considerable amount of, plays an important role in the degree of vection felt in the observer. In fact, many studies have demonstrated that optical flow perceived in the periphery increases feelings of vection [24, 89, 224]. However, it should be noted that there is evidence of feelings of vection when foveal, and not peripheral, stimulation is present [220].

2.1.2 Simulator Sickness

Simulator sickness is the feeling of motion sickness experienced when using a VR system. When they experience vection, it is common for users to also experience simulator sickness. It is also possible for users to experience simulator sickness when using VR applications. Simulator sickness decreases the usability of VR and can potentially deter people from wanting to experience VR more than once. The exact cause of simulator sickness is not known, but the main theory argues that conflict between visual, proprioceptive, and vestibular stimuli is the cause [108]. Hettinger et al. [78] strengthened this theory when they provided data suggesting that simulator sickness is a product of vection.

It has been noted that FOV influences simulator sickness—specifically, a smaller FOV has been shown to reduce the amount of simulator sickness users experience [53, 123]. A study by Fernandes et al. [62] further explored how FOV influences simulator sickness. In their study, they dynamically changed the FOV in VR using what they refer to as FOV restrictors. They concluded that changing the FOV based on visually perceived motion makes users feel more comfortable during their VR experiences [62].

2.2 Virtual Reality Locomotion Interfaces

Navigation consists of two processes: wayfinding and locomotion. Wayfinding is the process of determining the route through an environment that an agent (in our case, a human) must travel on to go from their starting location to their goal destination [48]. Locomotion refers to the low-level, mechanical process of how an agent travels along the route determined in the wayfinding step in order to reach its destination (e.g. walking, flying, driving). Locomotion in VR is essential for exploring VEs and delivering an interactive experience. A lack of support for locomotion within the VE may reduce feelings of presence and, in turn, make VR less effective [179].

Human gait features a wide range of movements like walking, running, skipping, and waddling. A good locomotion interface must support these motions, while also accounting for a variety of physical space shapes and user dimensions. Supporting such a variety of movements is a challenge for VR systems. In this section, we will discuss the advantages and disadvantages of different locomotion interfaces.

A locomotion interface is a device and/or software that allows a user to travel in a virtual environment. Ideally, a locomotion interface should allow the user to naturally walk[1] (or *perfectly* mimic the sensations felt when one really walks), be easy to understand, and require minimal extra hardware or setup. A number of different locomotion interfaces have been proposed, prototyped, and evaluated. Some well-known interfaces include joystick controls, omnidirectional treadmills [97], powered shoes [99], moveable tiles [98], and *redirected walking* [158]. Different

[1] "Naturally walk" refers to step-driven locomotion that does not use treadmills or other mechanical devices, makes use of the entire gait cycle, and, ideally, is perceived by the user as identical to how they walk in the real world while not in VR [116, 188].

locomotion interfaces may be undesirable in different situations because they do not meet all the criteria of an ideal locomotion interface. Suboptimal locomotion interfaces are usually unsatisfactory because they involve unwieldy hardware or lack vestibular or proprioceptive feedback that is present during real walking, e.g. a treadmill.

Of the locomotion interfaces that have been studied, interfaces that utilize *redirection techniques* (RTs) are especially appealing since they allow users to walk naturally while exploring a VE. RTs allow users to explore VEs that are larger than the tracked workspace by manipulating the user's path in the virtual environment [140]. It has been shown that natural walking is the most intuitive and beneficial locomotion technique in VR, as it improves users' sense of presence [209], memory, and performance [85, 149, 162]. As a result of the numerous benefits real walking offers, researchers have invested considerable effort into developing and understanding locomotion interfaces that support real walking.

2.2.1 Natural Walking in Virtual Reality

Standard VR systems do allow users to walk around during a virtual experience, but users are only able to walk within the tracked space. Movement outside the workspace borders will not be tracked by the system's sensors, so the visual scene displayed on the HMD will not update according to the user's movements. Thus, the size of the VE that a user can explore is limited to the size of the tracked space.

To support real walking and increase the size of the explorable VE, we can employ RTs. A multitude of redirection techniques have been developed [27, 94, 158, 194], which has prompted researchers to classify RTs based on their implementation-specific characteristics. Suma et al.

distinguished between redirection techniques based on the conspicuousness (overt or subtle) and continuity (discrete or continuous) of their implementations [195]. Subtle and continuous techniques are preferred because they have been reported to create fewer breaks in presence. However, depending on the user's projected path and position in the workspace, we cannot always rely on such techniques to keep users in the tracked workspace. In these situations, redirection systems may sometimes be required to fall back on more overt techniques to ensure the user's safety [140, 195].

2.2.2 Redirected Walking

One popular subtle and continuous RT that enables natural walking in VR is *redirected walking* (RDW) [158]. RDW involves imperceptibly manipulating the VE via rotations and translations so that a user subconsciously adjusts their real-world position to remain on their intended virtual path. Using this technique, we can steer users away from the tracked-space edges while still giving users the benefits of real walking in the VE. This reduces the amount of breaks in presence caused by reaching the bounds of the tracked space.

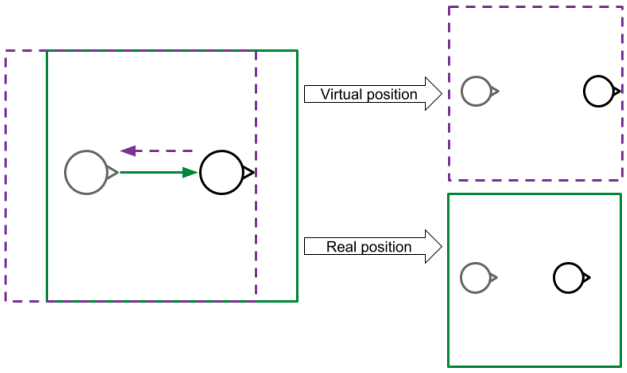
For example, a user will physically rotate by 180° when he or she wants to turn 180° in the VE, if no redirection is applied. If redirection is applied such that some real-world rotation results in a larger rotation in the VE, the user will turn until their position in the VE has rotated 180°, but the physical rotation will be less than 180°. We can also redirect such that a physical rotation results in a smaller virtual rotation. When implemented carefully, this discrepancy between the physical and virtual movements is imperceptible to the user if it is small enough. Similar transformations can be applied to a user's walking path. When walking on a straight path, we

can translate the VE in the direction opposite to the user's walking direction, which results in a virtual displacement that is larger than the user's physical displacement. We can also rotate the VE while the user walks to force the user to follow a curved path in the real world. Depending on the strength and direction of the rotation, this will force the user's real path to steer away from the edges of the tracked space. See Figure 2.1 for a diagram that explains how RDW manipulates the VE. This thesis is only concerned with rotations of the VE when the user is standing in place.

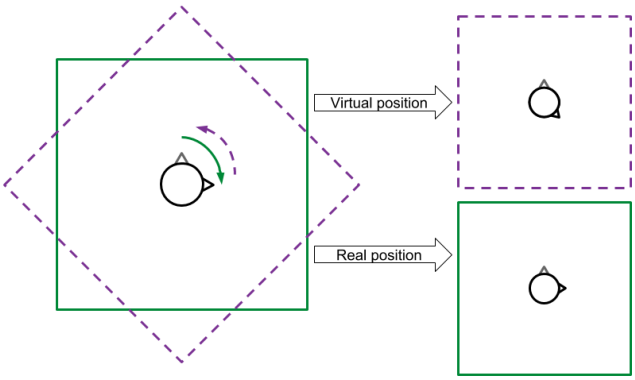
2.2.2.1 Limits of Redirection

By applying RDW, users are able to walk naturally and explore VEs larger than the tracked workspace. However, we cannot simply amplify users' movements by a large, constant factor to maximize the size of the explorable VE without incurring negative repercussions such as disorientation or increased simulator sickness. The scaling of a user's movements must be small enough to maintain the VR application's usability and ensure the user's comfort. Thus, there exists a trade-off between redirection intensity and user experience [158]. Ideally, enough redirection is applied to maximize the explorable size of the VE and minimize discomfort and breaks in presence caused by manipulating the VE.

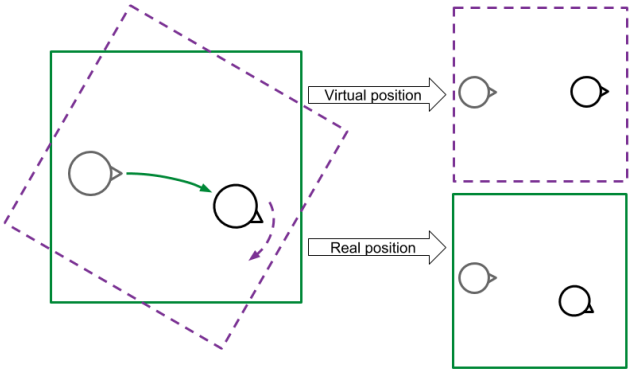
The intensity of scaling applied to the VE is controlled by parameters called *gains*. Rotation gains increase or decrease a user's rotation in the VE relative to their real-world rotation, while translation gains increase or decrease a user's displacement in the VE relative to their real-world displacement. Curvature gains, on the other hand, cause users to walk along a curved physical path while walking on a straight virtual path. Both rotation and translation gains are expressed as a ratio of virtual motion to physical motion. A gain of 1 is applied when virtual motion to



(a) A translation gain allows the user to walk distances in the VE that are greater than the distance walked in the real world.



(b) A rotation rotation gain allows the user to turn a greater virtual distance compared to their physical rotation.



(c) A curvature gain forces the user to walk on a curved physical path in order to walk in a straight path in the VE.

Figure 2.1: Diagrams that illustrate how different RDW gains can be used to increase the size of the explorable VE. The green borders represent the real-world tracked space borders, and the purple borders represent the borders of the VE that correspond to the size of the tracked space. Arrows indicate the user (green) or VE (purple) movement.

physical motion is mapped 1:1. When a gain is greater than 1, the virtual movement (rotation or translation) is increased, and the resulting real-world movement is smaller than the virtual movement. Similarly, when a gain is less than 1, the virtual movement is decreased, and the resulting real-world movement is larger than the virtual movement. A *threshold* refers to the point at which the applied gain becomes noticeable to the user, and each threshold has an associated gain. A threshold t corresponds to a gain g. A t threshold of g means that $t\%$ of the population will believe that their virtual movements are larger than their physical movements when the gain g is applied. For example, if the 50% threshold has a gain of 1.02, then half the population will believe that their physical and virtual movements are the same when we apply a gain of 1.02 while the other half will believe that their virtual movements are larger than their physical movements. In previous work by Steinicke et al. the threshold values of interest are users' 25% and 75% thresholds, which correspond to *decreased* and *increased* virtual rotations respectively [187].

VE rotation is often discussed in relation to the user's physical rotation. VE rotation *with* the user's physical rotation direction corresponds to a real-world rotation that is larger than the virtual rotation, and VE rotation *against* the user's physical rotation direction corresponds to a real-world rotation that is smaller than the virtual rotation.

2.3 Motion Planning

In the field of robotics, motion planning is the problem of moving a robot from an initial state to a goal state through a series of valid configurations that avoid collisions with obstacles [115]. For a robot with n degrees of freedom, its configuration space (denoted $\mathcal{C}$) is an n-dimensional manifold, where each point in the manifold corresponds to a configuration of the

robot. The configuration space describes the set of all states that a robot can be in. In order to successfully navigate from a starting position to a goal position, the robot must find a set of configurations that takes it from the starting position to the goal position. This can be formulated as finding a continuous path of valid configurations through $\mathcal{C}$. Some common desirable traits of such a path are that it yields the shortest path and that the robot trajectory is smooth, without many oscillations as it travels along this path. Motion planning has seen great success in allowing researchers to create navigation algorithms that are rigorously defined, can provide guarantees on navigation completion, and are robust to unknown, unpredictable, and dynamic environments.

There is considerable work on developing motion planning algorithms for static and dynamic environments. Search-based planners discretize the state space (the set of all possible states) and employ search algorithms to find a path from the start to the goal. An example of a search-based motion planning algorithm is the A* algorithm [76]. Sampling-based planners operate by randomly sampling the configuration space in order to build a valid path. Such algorithms can usually quickly find valid solutions, but their solutions are usually not the most efficient [60]. Potential field methods uses attractive and repulsive forces to guide the robot through the environment [105]. These planners are easy to implement but are susceptible to getting the robot trapped in local minima of the potential function. Planning algorithms may also use geometric representations, such as visibility graphs and cell decomposition, to reason about the environment, detect collisions, and compute collision-free paths [49]. Motion planning algorithms may also use optimization to to handle dynamic obstacles and compute smooth trajectories [144, 156]. Optimization-based approaches are advantageous in that they can more easily handle complex, high-dimensional state spaces. Dynamic motion planning is the problem of computing a collision-free path in an environment with moving obstacles. A popular approach to dynamic

motion planning is the use of velocity obstacles to reason about collision-free paths in terms of velocity [64, 211].

In this dissertation we show how VR locomotion can be reframed as a special kind of motion planning (Chapter 4). We use motion planning as a kind of "language" which we can use to precisely define constraints on VR locomotion and reason about the dynamics of VR locomotion. As we will show, this allows us to develop new redirection algorithms and environment complexity metrics that would be difficult to create using purely heuristic-based approaches like much of prior work in the VR locomotion community has done.

Part II

Locomotion Interfaces and Metrics for Natural Walking in Virtual Reality

Chapter 3: Alignment-Based Redirection

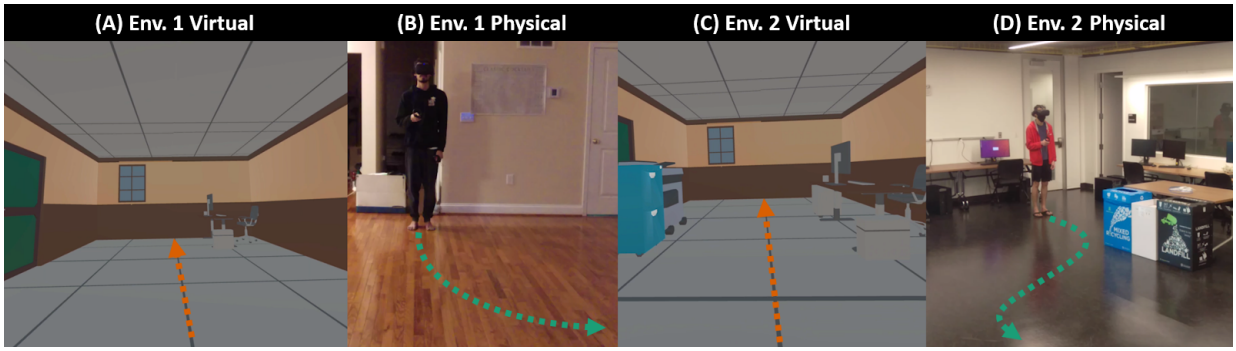


Figure 3.1: A user being steered with our alignment-based redirection controller (ARC) in two different environments. In Environment 1, the virtual environment (VE) is larger than the physical environment (PE), and there is an obstacle in the northeast corner of the VE. The PE has no obstacles. In Environment 2, the VE is larger than the PE, and both have obstacles in different positions. **(A)** The user walks in a straight line forward in the VE. **(B)** In the PE, the user is steered on a curved path away from the edge of the tracked space, in order to minimize the differences in proximity to obstacles in PE and VE. **(C)** The user walks in a straight line forward in the VE, with obstacles on either side of the path. **(D)** The user is steered on a path with multiple curves in the physical space. The user avoids a collision with the obstacle in front of them, and is also steered to minimize the differences in proximity to obstacles in the PE and VE. We are able to steer the user along smooth, collision-free trajectories in the PE. Our extensive experiments in real-wold and simulation-based experiments show that in simple and complex environments, our approach results in fewer collisions with obstacles and lower steering rate than current state-of-the-art algorithms for redirected walking.

In this chapter, we present a novel redirected walking controller based on alignment that allows the user to explore large and complex virtual environments, while minimizing the number of collisions with obstacles in the physical environment. Our alignment-based redirection controller, ARC, steers the user such that their proximity to obstacles in the physical environment matches the proximity to obstacles in the virtual environment as closely as possible. To quantify a

controller's performance in complex environments, we introduce a new metric, Complexity Ratio (CR), to measure the relative environment complexity and characterize the difference in navigational complexity between the physical and virtual environments. Through extensive simulation-based experiments, we show that ARC significantly outperforms current state-of-the-art controllers in its ability to steer the user on a collision-free path. We also show through quantitative and qualitative measures of performance that our controller is robust in complex environments with many obstacles. Our method is applicable to arbitrary environments and operates without any user input or parameter tweaking, aside from the layout of the environments. We have implemented our algorithm on the Oculus Quest head-mounted display and evaluated its performance in environments with varying complexity.

3.1 Introduction

Exploring virtual environments (VEs) is an integral part of immersive virtual experiences. Real walking is known to provide benefits to sense of presence [209] and task performance [162] that other locomotion interfaces cannot provide. Using an intuitive locomotion interface like real walking has benefits to all virtual experiences for which travel is crucial, such as virtual house tours and training applications. Redirected walking (RDW) is a locomotion interface that allows users to naturally explore VEs that are larger than or different from the physical tracked space, while minimizing how often the user collides with obstacles in the physical environment (PE) [157]. RDW works by slowly transforming the VE with rotations or translations such that these transformations are imperceptible while still creating a subtle discrepancy between the user's physical and virtual trajectories. This discrepancy causes the user to adjust their physical

trajectory to counteract the virtual camera motion induced by the virtual transformations, in order to stay on their intended virtual trajectory. RDW is an appealing locomotion interface because it allows users to explore the VE using real, natural walking.

Although RDW is a good locomotion interface for immersive virtual experiences, it has two main limitations. First, the amount of redirection that can be applied to steer the user is dependent on how easily the user can consciously perceive the virtual transformations induced by RDW. The amount of redirection applied is controlled by *gains*, which determine the magnitude (intensity) of rotations and translations applied to the VE. The gains that transform the VE the most, while still remaining imperceptible to the user, are the *perceptual threshold* gains [187]. The intensity of a gain corresponds to the amount of deviation between the user's physical and virtual paths, and thus the amount that the user is steered in the PE. Gains with high intensity result in more redirection of the physical user at the cost of larger VE transformations, which can be more easily detected and can cause simulator sickness [140].

The second limitation is that the effectiveness of RDW at minimizing the number of collisions with physical obstacles depends on the *relative complexity* of the physical and virtual environments. Navigation through an environment becomes more difficult when the environment is populated with more obstacles because the user has fewer options for possible collision-free routes that allow them to avoid collisions with obstacles when making any movements. Thus, the complexity of an environment can be described in terms of the density of the obstacles in the environment. For example, the empty VE used by Bachmann et al. [11] would be considered low complexity, while the maze-like VE with multiple branching paths used by Nescher et al. [136] would be considered high complexity. In virtual reality (VR), the user navigates through a virtual and physical environment *at the same time*. A movement in one environment is paired with a

movement in the other. Therefore, the navigation problem becomes harder, since all movements must consider the obstacles in the VE as well as the PE. If the complexities (density and layout of obstacles) of the PE and VE are similar, avoiding collisions is easier, since a movement that yields a particular result (collision or no collision) in one environment is likely to result in the same outcome in the other environment. However, if the complexities of the PE and VE are very different, navigation is harder, because a movement in one environment will likely lead to a movement in the other environment with a different outcome.

In the context of VR, a *redirection controller* determines the amount of redirection to apply, given the user's position in the physical and virtual environments [140]. At each frame, the controller decides the level (intensity) of gains to apply in order to rotate or translate the VE and alter the user's physical trajectory. The controller uses heuristics or optimization in conjunction with information about the PE and/or VE in order to determine the level of gains to apply. *Reactive* controllers make decisions on how to steer the user based on the instantaneous state of the system, while *predictive* controllers make decisions based on predictions about the user's future trajectory. Reactive controllers typically do not consider the VE when setting gains, and the VE is often abstracted away by using an unbounded, empty environment. This design decision allows reactive controllers to be simpler and remain relatively effective without requiring much additional information, at the cost of worse performance than predictive controllers in some environments. On the other hand, predictive algorithms often use information from the VE to make predictions about the user's virtual trajectory. From these predictions, these algorithms are able to perform better than reactive algorithms by applying gains that are more suited to the user's environments and trajectory [136, 243]. Predictive algorithms rely on accurate predictions however, so they usually do not perform well if it is difficult to forecast the user's movement.

At a high level, *alignment* can be defined as a state in which the user's physical and virtual configurations match. It was first formally studied by Thomas et al. [206, 207] with the goal of allowing the user to interact with the PE when they are within some predefined region of the VE, to enable haptics. Prior to the work of Thomas et al., Zmuda et al. [243] developed a controller that used a core idea of alignment; their controller was allowed to place the user near a physical obstacle if its position relative to the physical user matched the position of a virtual obstacle relative to the virtual user. Alignment provides a simple way to consider both the physical *and* virtual environment when steering the user, which is important for developing effective controllers.

Main Contributions: We present a novel alignment-based redirection controller (ARC) for locomotion in VR. ARC is a redirection controller that applies RDW gains to steer the user such that the user's proximity to obstacles in the PE matches their proximity to obstacles in the VE as closely as possible. Our controller is able to steer users through physical and virtual environments that have different relative complexities and makes no assumptions about the distribution of the obstacles in each environment. In these complex environments, ARC achieves a lower number of collisions with physical obstacles when compared to current state-of-the-art controllers. Furthermore, ARC achieves this lower number of collisions while also redirecting the user using *less intense* gain than other controllers, which reduces the likelihood that users experience simulator sickness during locomotion [140]. We conduct extensive experiments in varied environments, using many different performance metrics, and find that ARC consistently outperforms existing state-of-the-art algorithms. The main contributions included are:

- A novel alignment-based redirection controller that can function in arbitrary environments,

without requiring any information from the application except for a map of the PE and VE and the obstacles in each environment. Benefits of our algorithm include:

  - Significantly fewer collisions in PE/VE pairs with similar complexities *and* in PE/VE pairs with very different complexities.
  - A lower steering rate, which helps avoid simulator sickness and increases the usability of the system for users with high sensitivity to redirection.

- A novel metric, Complexity Ratio (CR), for measuring and comparing the complexity of PE/VE pairs in the context of VR navigation. Using CR, we can directly assess a controller's performance in different environments, which allows us to compare redirection controllers easily.

- Extensive simulation-based evaluation of ARC compared to current state-of-the-art controllers. From our experiments, we conclude that alignment is an effective tool for minimizing collisions with physical obstacles when steering a user with RDW in simple and complex environments. We also show our controller working in a proof of concept implementation on the Oculus Quest.

3.2 Background

Redirected walking works by imperceptibly transforming the VE around a user such that they adjust their physical trajectory to compensate for the VE transformations and remain on their intended virtual trajectory [157]. The magnitude (or intensity) of the VE transformations is determined by *gains*. Razzaque et al. [157] defined three gains, rotation, translation, and

curvature, for rotating or translating the VE depending on the user's movement. Rotation gains rotate the VE around the user as they turn in place, which results in virtual rotations that are larger or smaller than the corresponding physical rotations, depending on the direction of the VE rotation relative to the physical rotation. Translation gains translate the VE forward or backward as the user walks in a straight line, which results in their virtual displacement being different from their physical displacement, depending on the direction of the VE translation. Curvature gains steer the user on a curved physical path by slowly rotating the VE around the user as they walk on a straight virtual path. The direction in which the user is steered is determined by the direction that the VE is rotated. Most research in redirected walking aims to either understand the perceptual limits of redirection or develop RDW controllers that minimize the number of collisions a user experiences during locomotion. An overview of different RDW methods is given in [140].

3.2.1 Perceptual Thresholds

The amount of redirection that can be applied before users notice the redirection is determined by perceptual thresholds. Perceptual thresholds are important to consider since strong redirection can induce simulator sickness [140] and break the user's feeling of presence in a virtual experience [195]. There has been considerable research into measuring the perceptual thresholds of each RDW gain, but there is no general consensus when it comes to selecting these thresholds [140].

The first comprehensive study of RDW thresholds was performed by Steinicke et al. [187]. Many researchers have since expanded on threshold estimation by reproducing results and measuring thresholds under different conditions. A study by Grechkin et al. [71] determined that translation and curvature gains can be applied simultaneously without altering either gain's perceptual thresholds.

Neth et al. [137] showed that a user's curvature gain detection thresholds are dependent on his or her walking speed. Williams et al. [230] reproduced the results found by Steinicke et al. [187] and demonstrated that users' perceptual thresholds could vary depending on their gender, the field of view, and the presence of distractors in the VE. Hutton et al. [92] suggest that perceptual thresholds can differ greatly between different people, which may explain the different threshold values reported in prior literature. Thus, a system using some commonly-accepted threshold values (such as those measured in [187]) may apply appropriate gains for most users. However, gains could still be too high for some users, which could induce sickness and make the user uncomfortable. All this suggests that there are still many open problems with respect to accurately measuring a person's perceptual thresholds. A recent review of studies that measured perceptual thresholds can be found in [111].

3.2.2 Redirected Walking Controllers

A redirection controller is an algorithm that decides which gains to apply at each frame in order to minimize the number of collisions the user incurs in the PE [140]. While a controller's goal is to minimize the number of collisions, it is important to note that a controller cannot guarantee a collision-free trajectory in all circumstances. A controller's effectiveness depends on the configuration of the physical and virtual environments (environment dimensions and size and location of obstacles), the virtual path traveled, and the user's perceptual thresholds.

Controllers fall into three categories: scripted, reactive, and predictive [140]. *Scripted controllers* steer the user as they follow a virtual path pre-determined by the system developers. Scripted controllers are effective at reducing the number of collisions but impose tight restrictions

on the VE. These controllers can perform very poorly if the user deviates from the pre-determined virtual path. Work studying scripted controllers includes the development of steering algorithms based on change blindness [194] and overlapping virtual spaces [196].

Reactive controllers steer the user based on information available from the user's previous movements and current state. These controllers are designed to work in a wide variety of PEs and VEs since they do not make assumptions about the user's future path. Reactive controllers fall short at achieving maximal collision avoidance since they do not use all the information available to the system. Razzaque [157] proposed three reactive algorithms for RDW: steer-to-center (S2C), steer-to-orbit, and steer-to-multiple-targets. Steer-to-center constantly redirects the user towards the center of the physical environment. Steer-to-orbit steers the user along a circular path that orbits the center of the PE. Steer-to-multiple-targets steers the user to one of multiple pre-determined physical goal positions, depending on the user's position in the PE. Despite being one of the first controllers ever, S2C has regularly outperformed other algorithms in a variety of environments [7, 83]. Additionally, steer-to-orbit performs well when the user walks on long, straight virtual paths [83]. However, more recent algorithms have performed as well as, or better than, S2C. Strauss et al. introduced a controller trained by reinforcement learning that outperformed S2C in simulated trials and performed as well as S2C in user trials [192]. Chang et al. [29] and Lee et al. [117] have also recently used reinforcement learning to train RDW controllers. Thomas et al. [205] and Bachmann et al. [11] simultaneously introduced controllers based on artificial potential fields that outperformed S2C in non-convex and multi-user environments.

Predictive controllers make predictions about the user's intended virtual path and steer them accordingly. Predictive controllers can be effective since they tend to use most of the

information available to the system. However, their performance relies partially on the accuracy of their predictions. Nescher et al. [136] and Zmuda et al. [243] developed predictive controllers that were outperformed S2C, while Dong et al. improved upon the potential field-based controllers by incorporating trajectory prediction into the controller [54].

In addition to determining the gains to apply at each frame, redirection controllers have a resetter component. When the user gets too close to a physical obstacle, the system initiates a reset maneuver in order to reorient the user away from the nearby obstacle. This reset maneuver is counted as a collision. The specific reset policy employed depends on the controller, but one popular resetting technique is the 2:1 reset [229], wherein the magnitude of a user's physical rotations is doubled, so a 180° physical turn yields a 360° virtual turn. Another effective reset technique is distractors, which are elements in the VE that capture the user's attention to reorient them [147, 148, 149]. Other reset techniques may be specific to the RDW controller, such as the reset-to-gradient technique seen in potential field controllers [11, 205].

Evaluation metrics for RDW controllers can depend on the experimental setup. The majority of all studies use the number of collisions as one performance metric. Other common metrics include the average virtual distance walked between collisions, the mean steering rate, and user performance at a virtual task. Since the success of an RDW controller depends on the environments and the path traveled, it can be difficult to compare algorithms using only performance metrics. Thus, it is common for researchers to test not only their new controller but also the state-of-the-art controllers in the same environments.

3.2.3 Environment Complexity Metrics

Measuring the effect of environment complexity on task performance is useful for understanding the interactions between an agent and its environment. This measurement enables us to understand and predict how an agent will perform at a task in an environment, which allows us to change the environment design or improve our algorithms accordingly.

VR researchers have studied how a user's ability to complete a task in an environment depends on the environment's complexity [21, 22, 154]. While those studies are useful for understanding the interactions between environment complexity and task performance, they did not quantify the environment complexity with precise metrics. This makes it difficult to generalize their results and makes it harder to predict how users will perform in unstudied environments. Researchers in robot navigation have developed metrics to quantify environment complexity [4, 45, 169]. These metrics allow researchers to group and classify environments by complexity and directly compare the performance of different algorithms in different groups of environments.

3.3 Redirection by Alignment

The concept of alignment in the context of redirected walking was first formally studied by Thomas et al. [206, 207]. However, Zmuda et al. [243] used key elements of alignment prior to the work of Thomas et al. Additionally, Simeone et al. [174] recently introduced a locomotion technique that is similar to alignment in that it aims to match the VE and the PE, but it does so by overtly manipulating the VE in real time. In this section, we define our notion of alignment and provide the details of our general alignment-based redirection controller.

3.3.1 Definitions and Background

3.3.1.1 Alignment

A redirection controller takes as input the current position and orientation of the user in the PE and the VE. We define the configuration of the VR system as the user's position and orientation in the PE and VE. *Alignment* is a configuration in which the user's physical state matches their virtual state. When this configuration is achieved, we say that the system (or user) is *aligned*. In this work, we are concerned with steering users on collision-free paths in the PE and VE at the same time. How close a user is to incurring a collision can be described by the distance to obstacles around them, i.e. their proximity to obstacles. Thus, we describe the user's *state* in an environment by their proximity to obstacles in the environment.

We assume that the user travels on a collision-free path in the VE. We associate a proximity function along each point on this path. This proximity function tells us how close the point is to obstacles in the environment. We want to define a proximity function that can be formulated for paths in the PE and VE, to be used by our redirection controller to compute collision-free paths to steer the user on.

Let $d(p, \theta)$ be the distance to the closest obstacle in the direction θ originating from a location $p = (x, y)$ in an environment. This distance can be computed using simple ray-intersection queries. Let $S = \{\theta_1, \theta_2, ..., \theta_k\}$ be a set of k discrete directions in the range $[0, 2\pi)$. We define the proximity function, $Prox(p)$, at a point p to be the sum of distances to obstacles in each direction $\theta_i \in S$:

$$Prox(p) = \sum_{i=1}^{k} d(p, \theta_i). \tag{3.3.1.1}$$

To compare proximity values $Prox(p_{phys})$ and $Prox(p_{virt})$ for a point p_{phys} in the PE and a point p_{virt} in the VE, we *cannot* simply compute $Prox(p_{phys}) - Prox(p_{virt})$. If we did, it would be possible to get a value of 0, implying perfect alignment, for positions in which the physical and virtual user are *not* actually perfectly aligned. For example, this can happen when $d(p_{phys}, \theta) - d(p_{virt}, \theta) = -1 \cdot (d(p_{phys}, \theta + \pi) - d(p_{virt}, \theta + \pi))$ for all $\theta \in [0, 2\pi)$.

To resolve this problem and have a meaningful notion of what it means to compare values of $Prox(p)$, we can instead sum the absolute value of the differences in distance over all θ_i:

$$dist(Prox(p_{phys}), Prox(p_{virt})) = \sum_{i=1}^{k} |d(p_{phys}, \theta_i) - d(p_{virt}, \theta_i)|. \tag{3.3.1.2}$$

Note that we use the same set of directions S for the PE and the VE. Computing this difference is too computationally expensive for large values of k, so in our implementation we approximate this value by computing the difference in distances in three directions around the point ($k = 3$).

Now we can define the physical and virtual states at time t which we use in our rediretion controller:

$$\begin{aligned} q_t^{phys} &= \{d(p_{phys}, \theta_{phys}), d(p_{phys}, \theta_{phys} + 90°), d(p_{phys}, \theta_{phys} - 90°)\}, \\ q_t^{virt} &= \{d(p_{virt}, \theta_{virt}), d(p_{virt}, \theta_{virt} + 90°), d(p_{virt}, \theta_{virt} - 90°)\}. \end{aligned} \tag{3.3.1.3}$$

Here, p_{phys} and p_{virt} are the user's positions in the physical and virtual environments, respectively. Similarly, θ_{phys} and θ_{virt} are the user's headings in the physical and virtual environments, respectively. Given the user's physical and virtual states, we define the state of the system at time t as the union

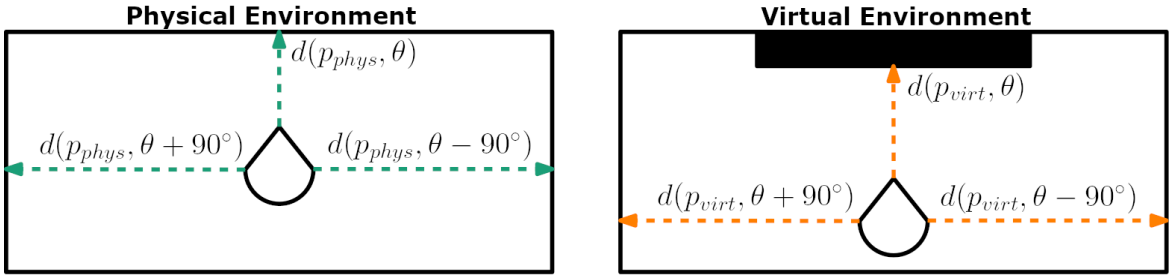


Figure 3.2: Visualization of the three values from the PE and three values from the VE that constitute a user's state.

of their physical and virtual states at time t:

$$Q_t = \{q_t^{phys}, q_t^{virt}\}. \tag{3.3.1.4}$$

This definition of state is illustrated in Figure 3.2.

Given Q_t, we can measure the alignment of the state, $A(Q_t)$ by computing the discretized version of Equation 3.3.1.2:

$$A(Q_t) = dist(q_t^{phys}, q_t^{virt}), \tag{3.3.1.5}$$

where $dist(q_t^{phys}, q_t^{virt})$ is defined as the sum of the absolute values of the differences between the distances to obstacles in the PE and VE:

$$\begin{aligned} dist(q_t^{phys}, q_t^{virt}) = &|d(p_{phys}, \theta_{phys}) - d(p_{virt}, \theta_{virt})| \\ &+ |d(p_{phys}, \theta_{phys} + 90°) - d(p_{virt}, \theta_{virt} + 90°)| \\ &+ |d(p_{phys}, \theta_{phys} - 90°) - d(p_{virt}, \theta_{virt} - 90°)|. \end{aligned} \tag{3.3.1.6}$$

The more similar a user's physical and virtual states are, the closer $A(Q_t)$ will be to 0. Conversely, a physical and virtual state that are very different will yield a larger value for $A(Q_t)$. We reiterate that one can define $A(Q_t)$ and $dist(q_t^{phys}, q_t^{virt})$ differently from how we have defined

them. A different definition corresponds to a different notion of what it means for a system to be aligned. Our controller is concerned with avoiding collisions in the PE, so proximity to obstacles was an appropriate way to define $A(Q_t)$ and $dist(q_t^{phys}, q_t^{virt})$.

With traditional RDW controllers, the goal of the system is to steer the user away from obstacles in the PE. With an alignment-based controller, the goal is to steer the user to a physical state that most closely matches the virtual state. In general, the VE will differ greatly from the PE, so it is common that a particular virtual state does not have a corresponding physical state with which it aligns perfectly. Thus, at any given instance, an alignment-based redirection controller aims to minimize the difference between the physical and virtual states and does *not* necessarily aim to perfectly align the two. If the global minimum yields perfect alignment, then an alignment-based controller should eventually reach this configuration. If it were possible to keep a user aligned at all times, the user would never encounter collisions while exploring a VE, and an alignment-based RDW controller could provide an optimal solution to the problem of RDW.

3.3.1.2 Environment Complexity

The complexity of an environment is dependent on the task to be completed in the environment [45]. For the purposes of locomotion, we define complexity as the ease with which a user can reach a goal destination without colliding with any obstacles in the environment. As the environment becomes populated with more obstacles, this becomes more difficult, and so the complexity of the environment increases.

Let $C(p)$ be the shortest distance between a point p and the closest obstacle in an environment

E. We define the complexity of E as the average value of $C(p)$ over all points in E:

$$C(E) = \frac{1}{|P|} \sum_{p \in P} C(p), \tag{3.3.1.7}$$

where P is the set of all points p in E. When there is a lot of open space in the environment (low obstacle density), $C(E)$ will be large; $C(E)$ approaches 0 as the amount of open space in the environment decreases (high obstacle density).

Since VR locomotion depends on a physical *and* virtual environment, we must relate the environments' complexity measures together. To do this, we compute the *complexity ratio* (CR) of the virtual environment E_{virt} and physical environment E_{phys}:

$$\text{CR} = \frac{E_{phys}}{E_{virt}}. \tag{3.3.1.8}$$

This definition of environment complexity gives us an intuitive way to describe how easy it is to locomote through an environment, and CR tells us how similar the complexities of two environments are. In VR, it is common for the PE to have a lower obstacle density than the VE, *i.e.* $C(E_{phys}) > C(E_{virt})$. A higher value for CR corresponds to a greater disparity in the complexity of the PE and VE, which implies that collision-free VR navigation is more difficult. This definition for CR is formulated on a continuous domain, which makes it difficult to compute exactly. To simplify the computation, we discretize the equation by sampling a point every 0.5 meters in the environment.

3.3.2 Alignment-based Redirection Controller

3.3.2.1 Redirection Heuristic

In this section, we provide details on how our alignment-based redirection controller (ARC) uses alignment to determine the RDW gains to apply at each frame. Note that ARC assumes that the user travels on a collision-free virtual trajectory in the direction of their heading (they do not walk backwards or side-to-side). Since the user's positions and orientations in the environments are known at all times through the tracking information, ARC can compute the $A(Q_t)$ on every frame according to the equations defined in subsubsection 3.3.1.1. If $A(Q_t) = 0$, no redirection is applied. If $A(Q_t) \neq 0$, ARC uses the following heuristics to set the redirection gains, depending on the user's current movement.

If the user is translating, the translation gain g_t is set to be:

$$g_t = \text{clamp}\left(\frac{d(p_{phys}, \theta_{phys})}{d(p_{virt}, \theta_{virt})},\ minTransGain,\ maxTransGain\right), \tag{3.3.2.1}$$

where $minTransGain = 0.86$ and $maxTransGain = 1.26$. The clamp$(x, minVal, maxVal)$ function returns x if $minVal \leq x \leq maxVal$, and returns $minVal$ if $x < minVal$ or $maxVal$ if $x > maxVal$. This heuristic for the translation gain speeds up the user's physical walking speed relative to their virtual walking speed if there is more open space in front of the physical user than there is in front of the virtual user. If there is more space in front of the virtual user than there is in front of the physical user, the user's physical walking speed decreased relative to their virtual walking speed. We set g_t equal to the ratio of the distances, bounded by previous measured perceptual thresholds, so that the translation gain changes gradually, which increases

user comfort.

If the user is undergoing a translation motion, we need to set the curvature gain g_c. First, we determine which of the spaces to the left and right of the physical user is more dissimilar to its virtual counterpart:

$$\begin{aligned} misalignLeft &= d(p_{phys}, \theta_{phys} + 90^\circ) - d(p_{virt}, \theta_{virt} + 90^\circ), \\ misalignRight &= d(p_{phys}, \theta_{phys} - 90^\circ) - d(p_{virt}, \theta_{virt} - 90^\circ). \end{aligned} \tag{3.3.2.2}$$

If $misalignLeft > misalignRight$, we want to steer the user to the left in order to minimize the misalignment. To do this, we set g_c as follows:

$$\begin{aligned} scalingFactor &= min(1, misalignLeft), \\ g_c &= min(1, scalingFactor \times maxCurvatureRadius), \end{aligned} \tag{3.3.2.3}$$

where $maxCurvatureRadius = 7.5m$. If we instead want to steer the user to the right in order to minimize the misalignment (i.e. when $misalignRight > misalignLeft$), we set g_c in a manner similar to Equation 3.3.2.3, but we exchange $misalignLeft$ for $misalignRight$ in the $scalingFactor$ computation. The sign of the curvature gain must also be set appropriately to steer the user in the desired direction. With this heuristic, we set the curvature gain proportional to the misalignment on the left or right of the user, depending on which is larger. The gain is bounded by the maximum curvature gain of radius $7.5m$ to reduce the chance that the user feels simulator sickness during redirection.

If the user is rotating, we set the gain according to the user's distance to objects in front of *and* on both sides of the user. Our heuristic for g_t only considers the distance to objects in

front of the user since translation gains only alter the forward and backwards displacement of the user. The heuristic for g_c only considers the distances to objects on either side of the user since curvature gains only steer the user to the left or right when walking on a straight virtual path. However, our heuristic for the rotation gain considers all three distances in order to accurately describe the user's orientation. Orientation is a function of all 360° around the observer, so the most accurate measurement of orientational alignment would compare distances in all 360° directions. That degree of detail is not necessary, since $d(p, \theta)$ and $d(p, \theta + \Delta\theta)$ will produce very similar values for most positions p in an environment for small $\Delta\theta$. Furthermore, sampling distances in all directions around the observer is too computationally expensive to run in real-time, which is a requirement for reactive RDW controllers.

To set the rotation gain g_r, we check if the direction that the user is turning increases or decreases their rotational alignment. To compute this, we first compute the user's rotational alignment for the current and previous frames:

$$\begin{aligned} curRotaAlignment &= dist(q_t^{physical}, q_t^{virtual}), \\ prevRotaAlignment &= dist(q_{t-1}^{physical}, q_{t-1}^{virtual}). \end{aligned} \tag{3.3.2.4}$$

Then, we set g_r based on whether the current rotational alignment is better or worse than the previous rotational alignment:

$$g_r = \begin{cases} minRotaGain & prevRotaAlignment < curRotaAlignment, \\ maxRotaGain & prevRotaAlignment > curRotaAlignment, \\ 1 & \text{otherwise.} \end{cases} \tag{3.3.2.5}$$

Here, $minRotaGain = 0.67$ and $maxRotaGain = 1.24$. The rotation gain is smoothed by linearly interpolating g_r between frames with a weighting of 0.125 on the previous frame's gain. The idea behind this heuristic is that we want to speed up the user's rotation when they are turning in a direction that improves their rotational alignment, and slow down their rotation when they are turning in a direction that worsens their rotational alignment.

3.3.2.2 Resetting Heuristic

Since we cannot guarantee that the user will travel on a collision-free physical path, our alignment-based controller needs a resetting policy to reorient the user when they are about to collide with a physical obstacle. To ensure that the user does not actually walk into any obstacles, a reset is triggered when the user comes within $0.7m$ of an obstacle. Our reset policy reorients the user such that they face the direction in the PE for which the distance to the closest physical obstacle in the user's physical heading direction most closely matches the distance to the closest virtual obstacle in the user's virtual heading direction.

When a reset is triggered, let p_{phys} and p_{virt} be the user's physical and virtual positions, respectively, and let θ_{phys} and θ_{virt} be their physical and virtual headings, respectively. First, we sample 20 equally-spaced directions $\{\theta_1, \theta_2, ..., \theta_{20}\}$ on the unit circle centered at p_{phys}. For each θ_i, we compute the distance to the closest physical obstacle in that direction as $d(p_{phys}, \theta_i)$ to produce 20 distances $\{d_1, d_2, ..., d_{20}\}$. The direction the user will face after the reset is complete, denoted θ_{reset}, is the θ_i for which the corresponding distance d_i most closely matches the distance $d(p_{virt}, \theta_{virt})$. This value θ_{reset} is subject to two constraints. First, θ_{reset} must face away from the

obstacle the user is about to walk into:

$$dot(\theta_{reset}, obstacleNormal) > 0, \tag{3.3.2.6}$$

where $obstacleNormal$ is the normal of the closest face of the obstacle that triggered the reset. The second constraint is that $d(p_{phys}, \theta_{reset}) \geq d(p_{virt}, \theta_{virt})$. If this second constraint cannot be satisfied by any of the θ_i that also satisfy the first constraint, θ_{reset} is set to the direction that minimizes the difference between $d(p_{phys}, \theta_{reset})$ and $d(p_{virt}, \theta_{virt})$ and satisfies the first constraint. Please refer to Figure 3.3 for a visual explanation of our resetting policy.

Once θ_{reset} is computed, we turn the user to face that direction. The user is instructed to turn in place until their heading is the same as θ_{reset}, while the virtual turn is scaled up to be a $360°$ turn. In order to minimize the amount of rotational distortion required for the reset, the user turns in the direction of the larger of the two angles between θ_{phys} and θ_{reset}. This method for resetting is inspired by the reset method used by Bachmann et al. [11].

3.4 Experiments

We conducted three experiments in simulation, each with a different pair of physical and virtual environments (see Subsection 3.4.3). For each experiment, we compared our controller with two reactive controllers: an artificial potential function-based algorithm (APF) and steer-to-center (S2C). The APF controller is implemented as described by Thomas et al. [205]. We compared our method with APF because it is currently the state-of-the-art reactive RDW controller, having been shown to perform well in empty environments, environments with obstacles, and environments with multiple users [11, 54, 132, 205]. We chose to also compare our method

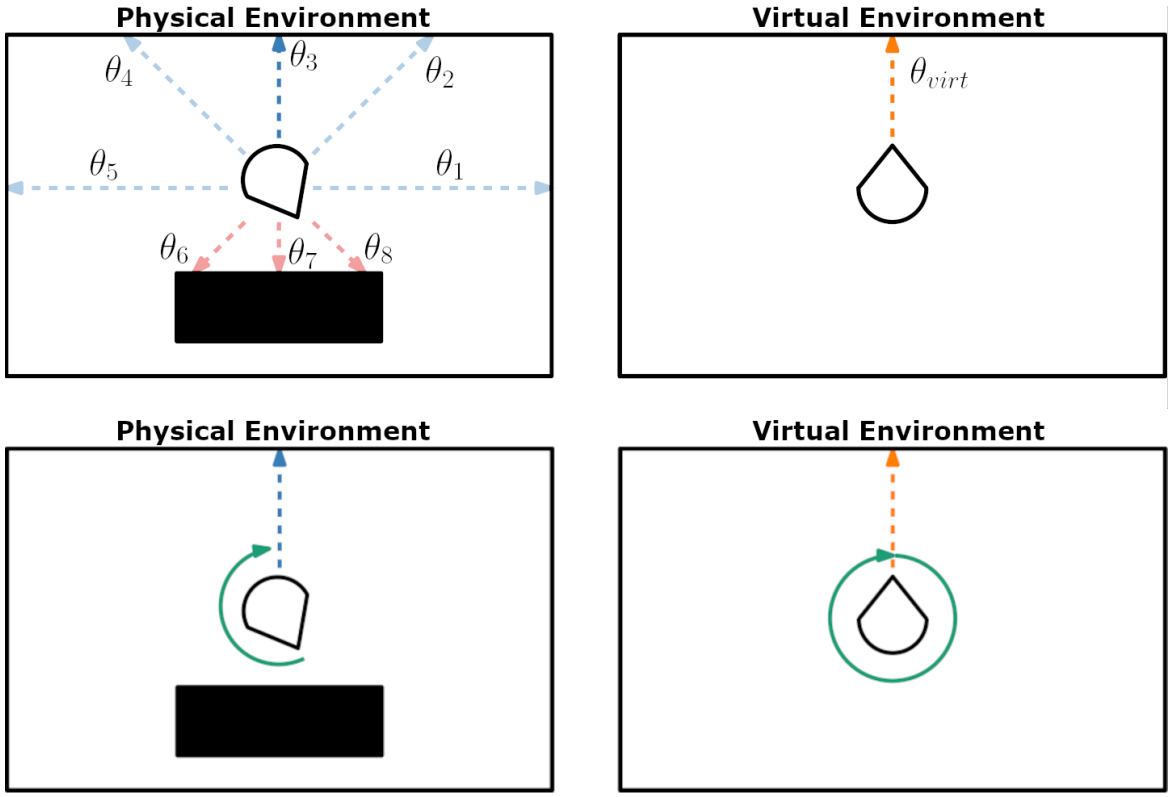


Figure 3.3: A visualization of the two steps involved in resetting. The top row shows the process of selecting the best direction for resetting. In this example, θ_{reset} is chosen to be θ_3. To reduce visual clutter, we only show eight of the twenty sampled directions. The bottom row shows the user to turning to face the best direction.

with S2C because it is a commonly-used benchmark in the RDW controller literature. Our implementation of S2C is the same as the one developed by Hodgson et al. [84] since it has many improvements over the original S2C algorithm proposed by Razzaque [157]. We note that S2C is expected to perform very poorly in some of our environments (Subsection 3.4.3) due to obstacles near the center of the PE, and that it is fairer to compare ARC against APF in these environments, but we still evaluated S2C in these conditions for the sake of completeness. The reset policy used by APF and S2C was the modified reset-to-center policy described in [205]. We also informally tested a proof of concept VR implementation to evaluate ARC in real PE/VE pairs, for which we implemented ARC in the Unity 2019.4.8f1 game engine, and ran tests using an Oculus Quest head-mounted device. The participant in the proof of concept was one of the authors.

There are existing controllers that either have similar features to our alignment-based controller or were tested in similar environments. Zmuda et al.'s FORCE controller [243] makes use of the core assumption of alignment, that users will not walk into obstacles in the VE, to get performance gains. Nescher et al.'s MPCRed controller [136] uses information about the VE to inform the decisions about gain selection. Although those controllers are similar to our controller in some aspects, we did not compare our work against them in this work because they are predictive controllers, while our algorithm is purely reactive. Since predictive and reactive controllers have fundamental differences by definition [140], it would not be a fair comparison.

It should be noted that while it is unfair to compare ARC to predictive controllers, it may not be completely fair to compare with reactive controllers, either. ARC is not predictive in the sense that it does not explicitly predict the user's future trajectory. However, by computing the user's proximity, ARC does implicitly "predict" where a user will travel, since the algorithm assumes

that users will avoid virtual obstacles. ARC is reactive in the sense that all information used to set redirection gains is computed on a frame-by-frame basis, and no future or past information is used in the steering process. Thus, one can consider ARC to fall somewhere between predictive and reactive controllers, which suggests that the traditional taxonomy of redirection controllers [140] may need to be updated to include newer algorithms.

3.4.1 Performance Metrics

For each experiment, we compared the performance of the controllers using three quantitative performance metrics. The metrics we used are:

- **Number of resets:** The number of times the user collided with a physical obstacle. This is a standard metric in RDW literature.
- **Average distance walked between resets:** The average of the physical distance walked on a path before incurring a collision.
- **Average alignment:** The average alignment $A(Q_t)$ for a path.

We also include qualitative evaluations showing the amount of space in the physical environment that was used, showing the effects of CR on controller performance, and showing the distribution of curvature gains applied by each controller.

The number of resets and the average distance walked between resets both provide a measurement of how many collisions the user incurs during locomotion. The more collisions a user experiences, the shorter the distance between resets will be. We also use heat maps of the user's location in the physical environment as a qualitative metric for the number of collisions.

When a user collides with an obstacle, they start a reset maneuver (subsubsection 3.3.2.2) which involves turning in place where they stand. This is manifested as a large amount of time spent in one spot, which will be highlighted on the heat map. To measure and compare the intensity of gains applied by each controller, we computed the average curvature gain for each path walked, and present histograms of the frequency of each average curvature gain across all paths. This metric is the same as the average steering rate metric that commonly appears in RDW literature [11, 117, 192]. The average alignment metric is used to show that our algorithm does indeed optimize for a low alignment score, and that other algorithms do not.

3.4.2 Simulated Framework

Properly evaluating an RDW controller requires testing the controller on a large number of paths, ideally in varied environments. It is common to use simulations to evaluate RDW controllers to avoid the high cost of running user studies [11, 29, 54, 117, 132, 192, 205, 206, 207]. Our simulated experiments were conducted on a computer with an AMD Ryzen 7 3700X 8-Core processor (3.60 GHz), 16 GB of RAM, a GeForce RTX 2080 SUPER GPU, and 64-bit Windows 10 OS.

In an effort to make it easier to compare out work to prior research, our simulated user representation is similar to the one used by Thomas et al. [205]. The user was represented as a circle with radius $0.5m$. If the boundary of this circle came within $0.2m$ of an obstacle, this was counted as a collision and a reset was initiated. The user's walking velocity was $1m/s$, and their angular velocity was $90°/s$. The path model used to generate user trajectories is the same as the one developed by Azmandian et al. [7]. In this model, a waypoint is generated at a random

distance ranging from $2m$ to $6m$ away from the previous waypoint. The waypoint was placed at a random angle between π and $-\pi$ relative to the previous waypoint. To follow a series of waypoints, the user turned to face the next waypoint, and then walked in a straight line towards it. Our simulation ran with a timestep of 0.05. To compute distances to obstacles, we represent the PE, VE, and obstacles as polygons (sets of vertices).

3.4.3 Environment Layouts

Each of our three simulation experiments had a unique pair of physical and virtual environment configurations. Diagrams for each environment are shown in Figure 3.4.

Environment A includes an empty $10m \times 10m$ physical environment and an empty $10m \times 10m$ virtual environment. This simple environment is used as a sanity check and to show that our algorithm can guide users on collision-free paths if perfect alignment is achievable. The CR of Environment A is 1. Exact vertex coordinates for Environment A are listed in Table 3.1.

Environment A (physical)	
Boundary	$(-5,-5),(5,-5),(5,5),(-5,5)$

Environment A (virtual)	
Boundary	$(-5,-5),(5,-5),(5,5),(-5,5)$

Table 3.1: Coordinates of vertices of boundaries and obstacles in each environment.

Environment B is a moderately complex environment. The physical environment is a $12m \times 12m$ physical room with $2m$-wide corridors. These corridors are created by four $3m \times 3m$ square obstacles placed in the four quadrants of the room. The virtual space for Environment B is a $17m \times 12m$ room with $2m$-wide corridors, created by six $3m \times 3m$ square obstacles. Environment B was used to show that ARC can handle environment pairs that have locally similar

features (regular corridors of the same width) but globally different dimensions. The CR of Environment B is 1.170. Exact vertex coordinates for Environment B are listed in Table 3.2.

Environment B (physical)

Boundary	$(-6,-6),(6,-6),(6,6),(-6,6)$
Obstacle 1	$(-4,-4),(-1,-4),(-1,-1),(-4,-1)$
Obstacle 2	$(1,-4),(4,-4),(4,-1),(1,-1)$
Obstacle 3	$(1,1),(4,1),(4,4),(1,4)$
Obstacle 4	$(-4,1),(-1,1),(-1,4),(-4,4)$

Environment B (virtual)

Boundary	$(-11,-6),(6,-6),(6,6),(-11,6)$
Obstacle 1	$(-4,-4),(-1,-4),(-1,-1),(-4,-1)$
Obstacle 2	$(1,-4),(4,-4),(4,-1),(1,-1)$
Obstacle 3	$(1,1),(4,1),(4,4),(1,4)$
Obstacle 4	$(-4,1),(-1,1),(-1,4),(-4,4)$
Obstacle 5	$(-9,1),(-6,1),(-6,4),(-9,4)$
Obstacle 6	$(-9,-4),(-6,-4),(-6,-1),(-9,-1)$

Table 3.2: Coordinates of vertices of boundaries and obstacles in each environment.

Environment C is a highly complex environment. The physical environment is a $10m \times 10m$ physical room with three rectangular obstacles. In the center of the space is a $2m \times 4m$ obstacle. The bottom-left corner of the room features a $2m \times 2m$ square obstacle. Along the top boundary of the room is a $1m \times 4m$ obstacle. This PE was designed to represent a plausible layout for a room in a house (such as a living room). The virtual space used in Environment C is a $20m \times 20m$ room with regular and irregular polygonal obstacles scattered throughout the room. Environment C was used to show that our algorithm is able to steer users through environments that are different in both local and global features. The CR of Environment C is 1.625. Exact vertex coordinates for Environment C are listed in Table 3.3.

We also used two different PE/VE pairs in our proof of concept implementation. The first environment pair included a roughly $3.8m \times 6.9m$ PE and a roughly $5.65m \times 8.7m$ VE. The PE

was empty, and the VE had a roughly $1.7m \times 2.3m$ obstacle in the northeast corner of the room. The second PE/VE pair featured a roughly $4.87m \times 7.62m$ PE and the same virtual room from the first environment pair. The PE had a $1.1m \times 1.5m$ obstacle along the west wall, and the VE had a $1.7m \times 2.3m$ obstacle on the east wall, and a $1.7m \times 1.8m$ obstacle along the west wall.

Environment C (physical)

Boundary	$(-5,-5),(5,-5),(5,5),(-5,5)$
Obstacle 1	$(-4.5,-4.5),(-2.5,-4.5),$ $(-2.5,-2.5),(-4.5,-2.5)$
Obstacle 2	$(-2,-1),(2,-1),(2,1),(-2,1)$
Obstacle 3	$(-2,4),(2,4),(2,5),(-2,5)$

Environment C (virtual)

Boundary	$(10,-10),(10,10),(-10,10),(-10,-10)$
Obstacle 1	$(-4.5,-4.5),(-2.5,-4.5),(-3.5,-2.5)$
Obstacle 2	$(0,2),(2,1),(1,-2),(-1,-2),(-2,1)$
Obstacle 3	$(-2,4),(2,4),(2,5),(-2,5)$
Obstacle 4	$(-8.5,8.5),(-8.5,2.5),(-6.5,2.5),$ $(-7,7),(-2.5,6.5),(-2.5,8.5)$
Obstacle 5	$(-8,-1),(-8,-2),(-7,-2),(-7,-1)$
Obstacle 6	$(-7,-3),(-7,-4),(-6,-4),(-6,-3)$
Obstacle 7	$(-9,-5),(-9,-7),(-8,-7),(-8,-5)$
Obstacle 8	$(-6,-9),(-3,-7),(-3,-6),(-7,-8)$
Obstacle 9	$(3,-4),(3,-8),(7,-8),(7,-4)$
Obstacle 10	$(5,9),(4,8),(8,4),(8,8)$

Table 3.3: Coordinates of vertices of boundaries and obstacles in each environment.

3.4.4 Experiment Design

For each experiment, we generated 100 random, collision-free virtual paths with 100 waypoints. The user travelled along each path three times, using either S2C, APF, or ARC for redirection. Note that the *same* 100 paths were used for each redirection controller within a particular environment. Although the virtual paths were random, they all had the same starting location and direction within an environment. In Experiment 1, the virtual user started in the center of Environment

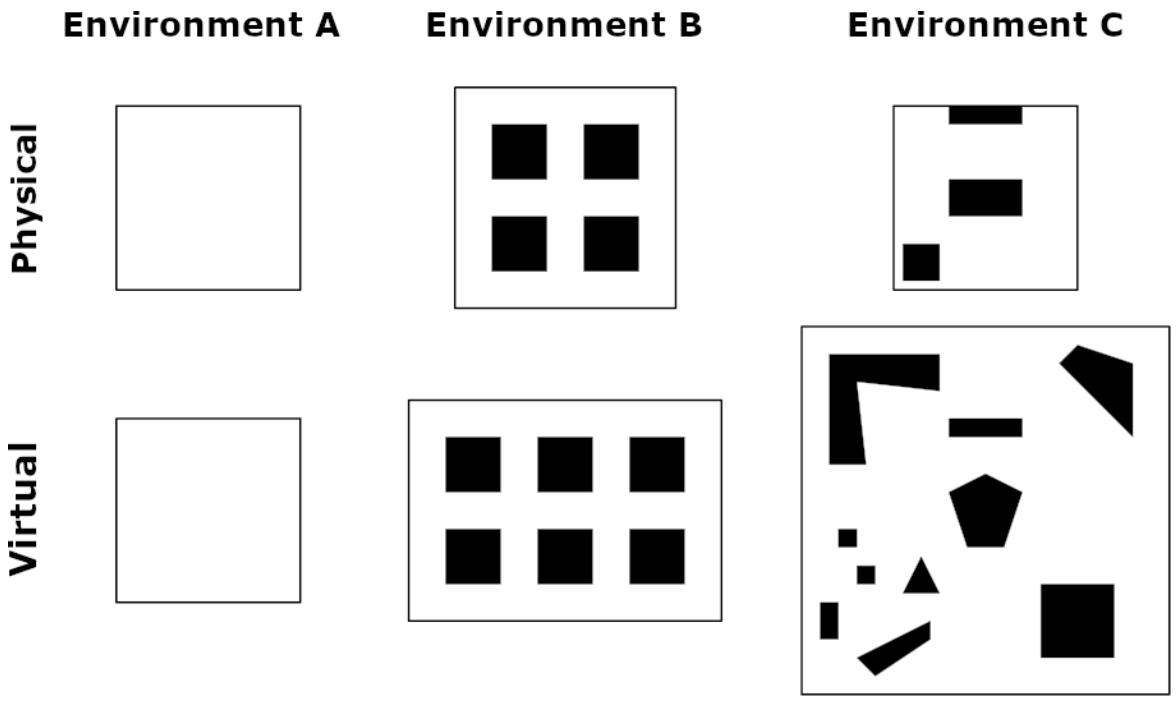


Figure 3.4: Diagrams of the physical and virtual environment pairs tested in our experiments.

A, facing north. In Experiment 2, the virtual user started in the center of Environment B, facing north. In Experiment 3, the virtual user started $3.5m$ below the center of Environment C (south of the pentagon), facing north. For every path, the physical user had a random starting location, but their heading direction matched that of the virtual user's at the start of the path. The physical starting location for a given path was the same regardless of the controller being evaluated. Having the user start in a random physical location increases the dissimilarity between the user's physical and virtual states, which makes it harder for a redirection controller to avoid collisions. We used these random starting positions to show that ARC is still able to achieve a low number of collisions, even in this difficult setting.

3.5 Results

We performed evaluations of the performance of S2C, APF, and ARC using three quantitative metrics and three qualitative metrics (Subsection 3.4.1). Each metric was computed for all 100 paths in each condition. RDW controller algorithms can sometimes encounter "unlucky" virtual paths that make it particularly difficult to avoid collisions, and the user will end up incurring many collisions in a short time frame. To make our comparisons robust to these unlucky paths, outliers in the data that were 1.5 times larger than the interquartile range of the data were replaced with the median of the data. We evaluated the normality of the data using visual inspection of Q-Q plots and histograms as well as measures of the distributions' skew and kurtosis. Homoscedasticity was evaluated using Levene's test. For each metric measured, the assumption of normality or homoscedasticity was violated, so we conducted all of our tests using a robust one-way repeated measures 20% trimmed means ANOVA (using the WRS2 package for R). Pairwise post-hoc comparisons were computed using linear contrasts.

Due to the large sample size in our experiments, we report 95% confidence interval instead of *p*-values. As the sample size grows, statistical tests become sensitive to small differences between samples, and the *p*-value becomes unreliable as it approaches 0. Confidence intervals, however, become narrower as the sample size grows, which means they are able to scale with the sample size and are still reliable for experiments with large samples. Furthermore, reporting confidence intervals allows for easier comparisons with future work if they also report confidence intervals, since the interval provides numerical bounds on the differences between conditions [124].

The results are shown in Table 3.4, with discussions in subsequent sections. Since the

distance walked between resets depends on the number of resets, we do not present the full analyses of the distance walked. Instead, we only include the average differences in distances walked (see Table 3.4). Most of the discussion is limited to comparing ARC to APF, since it is already well-established that APF outperforms S2C [11, 205]. A confidence interval that does not contain 0 in its range [CI lower, CI upper] is considered to be significant. Thus, we found significant differences between all groups in our post-hoc tests. The common pattern seen in the results is that ARC outperforms APF and S2C, while APF outperforms S2C. One condition for which this was not the case is the alignment metric in Environment C.

Environment A

	Number of resets			Distance walked between resets			Average alignment score		
Redirection Controller	$\hat{\psi}$	CI lower	CI upper	$\hat{\psi}$	CI lower	CI upper	$\hat{\psi}$	CI lower	CI upper
S2C [83] vs. ARC	16.983	14.066	19.901	-14.163	-17.857	-10.470	1.290	1.266	1.312
APF [205] vs ARC	5.750	2.686	8.814	-8.809	-12.967	-4.651	0.492	0.465	0.519
S2C [83] vs APF [205]	11.617	9.148	14.086	-5.822	-6.808	-4.836	0.801	0.789	0.813

Environment B

	Number of resets			Distance walked between resets			Average alignment score		
Redirection Controller	$\hat{\psi}$	CI lower	CI upper	$\hat{\psi}$	CI lower	CI upper	$\hat{\psi}$	CI lower	CI upper
S2C [83] vs. ARC	46.867	32.784	60.950	-0.507	-0.630	-0.385	0.808	0.778	0.838
APF [205] vs ARC	11.317	3.449	19.184	-0.130	-0.237	-0.023	0.761	0.747	0.774
S2C [83] vs APF [205]	36.408	23.560	49.256	-0.382	-0.502	-0.263	0.033	0.002	0.065

Environment C

	Number of resets			Distance walked between resets			Average alignment score		
Redirection Controller	$\hat{\psi}$	CI lower	CI upper	$\hat{\psi}$	CI lower	CI upper	$\hat{\psi}$	CI lower	CI upper
S2C [83] vs. ARC	2139.983	2062.215	2217.752	-3.719	-3.792	-3.645	-1.036	-1.060	-1.012
APF [205] vs ARC	137.108	124.065	150.152	-2.296	-2.420	-2.171	-0.109	-0.131	-0.087
S2C [83] vs APF [205]	2005.500	1928.841	2082.159	-1.414	-1.488	-1.339	-0.928	-0.961	-0.894

Table 3.4: The results of pairwise post-hoc comparisons between controllers, computed using linear contrasts and reported using confidence intervals due to the large sample size [124]. For each metric, $\hat{\psi}$ is the difference in estimated means between the two groups (estimate of the true mean). CI lower is the lower bound of the confidence interval on this difference, and CI upper is the upper bound. Narrower intervals indicate a more precise estimate of the true mean. We can interpret a cell as the estimated difference between the group means ($\hat{\psi}$), and CI lower and CI upper to represent that on 95% of samples, the true difference in means between the groups will fall in the range $[\hat{\psi} - \text{CI lower}, \hat{\psi} + \text{CI upper}]$. For a given row that compares Algorithm X vs. Algorithm Y, a positive $\hat{\psi}$ value indicates that Algorithm X scored more than Algorithm Y by that $\hat{\psi}$, while negative a value indicates that Algorithm X scored lower than Algorithm Y by that $\hat{\psi}$, bounded by CI lower and CI upper.

3.5.1 Experiment 1 (Environment A)

3.5.1.1 Number of resets

The robust repeated-measures ANOVA revealed a significant effect of redirection controller on the number of resets in Environment A $F(1.68, 98.95) = 126.1711, p < .0001$. ARC outperformed both S2C and APF because it was usually able to steer the user on the same virtual paths as S2C and APF, but with fewer collisions. We noticed that ARC sometimes performed worse than APF. However, ARC was also sometimes able to achieve perfect alignment and steer users along paths with no collisions, which APF and S2C were not able to do.

In this experiment, APF outperformed S2C. However, in the implementation of APF by Thomas et al. [205], they did not find significant differences in the number of resets between APF and S2C in Environment A. Since APF steers the user towards the center of the room, it is expected that APF and S2C will have similar results, as they did in [205]. The difference in performance between APF and S2C is possibly explained by the stronger curvature gains applied by APF, since our implementation of S2C [83] includes gain smoothing, whereby curvature gains transition gradually between values instead of instantly applying the strongest gain, as is done in APF [205].

3.5.1.2 Average alignment

The robust trimmed means ANOVA revealed a significant effect of steering controller on the user's average alignment $F(1.39, 82.19) = 10870.26, p < .0001$. The user's average alignment score was usually lower when they were redirected with ARC than when they were redirected with either APF or S2C. Neither APF nor S2C is designed using concepts of alignment, so they should not be expected to achieve higher alignment scores than ARC does. We also

observed that user's alignment was consistently low regardless of the path, which signals the reliability of ARC in making the state at least close to aligned.

3.5.1.3 Average Distance Walked Between Resets

There was a significant effect of controller on the distance walked $F(1.09, 64.09) = 57.9766, p < .0001$. A boxplot of the average distance walked between resets for each controller is shown in Figure 3.11, and the precise difference between controllers is shown in Table 3.4. When navigating with ARC, the user was able to walk further without colliding with a physical obstacle when compared with APF and S2C. The long upper whisker and the dots representing outlier paths indicate that for some paths, the simulated user was able to walk over $45m$ before colliding with an obstacle, which shows that ARC is able to deliver VR experiences with very few resets.

3.5.1.4 Qualitative Evaluations

The heat map of the physical space for Environment A is shown in Figure 3.5. For all conditions, the user spent most of their time near the center of the environment. This is what S2C is designed to do, and APF reduces to S2C in empty environments, so this is no surprise. One interesting thing to note, however, is that when the user is steered using ARC, they spend less time at the center of the room than with S2C and APF.

We observed differences in the average curvature gain applied by the controllers (Figure 3.6). The implementation of APF we used [205] always applies maximum curvature gain. However, S2C and ARC do not always apply the maximum curvature gain. The average curvature gain for S2C is always above $6°/s$, which is still fairly high considering the perceptual limit is $\approx 7.6°/s$. On the other hand, ARC is able to apply curvature gains much lower than the perceptual limit and with high consistency, mostly ranging from $3°/s$ to $6°/s$. All of the average gains applied by

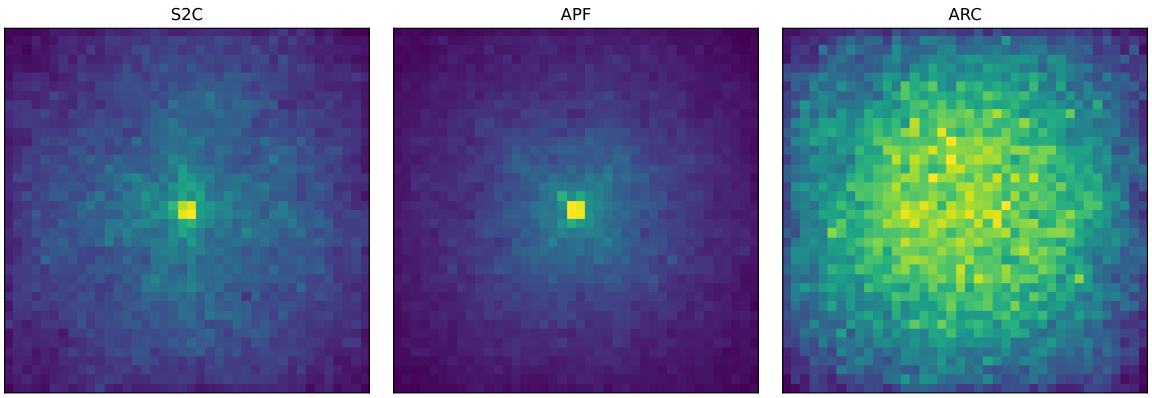


Figure 3.5: A heat map of the user's physical position across all paths for each controller in Environment A. Yellow tiles indicate the most time spent at that location, while purple tiles indicate the least amount of time. S2C and APF steer the user such that they spent the large majority of their time in the center of the room, while ARC allows the user to visit each region of the room more evenly.

ARC are below $7°/s$.

3.5.2 Experiment 2 (Environment B)

3.5.2.1 Number of resets

The robust ANOVA revealed a significant effect of controller on the number of resets $F(1.6, 94.42) = 56.0129, p < .0001$. ARC and APF have somewhat similar performances, although ARC still resulted in significantly fewer resets than APF. Furthermore, the interquartile range for the number of resets is lower for ARC than it is for APF, supporting the notion that ARC delivers a consistent locomotion experience that is robust to different virtual paths.

3.5.2.2 Average alignment

A significant effect of redirection controller on the user's average alignment was found $F(1.44, 84.85) = 3484.467, p < .0001$. The same trend seen in Environment A for the average alignment score is also seen in Environment B. ARC achieves a noticeably lower alignment score, which shows that ARC is able to successfully steer the user to a more aligned state.

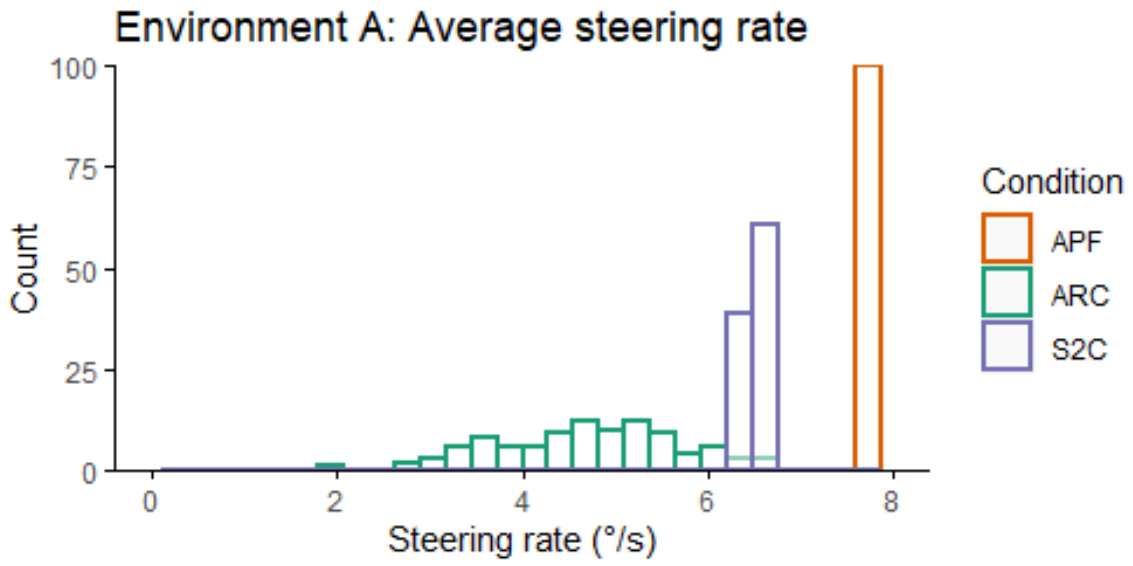


Figure 3.6: A histogram of the average curvature gain applied by each controller for each path in Environment A. The implementation of APF we used always applies the same gain, while S2C and ARC apply lower gains on average. S2C still applies gains fairly close to the perceptual threshold ($\approx 7.6°/s$), but ARC is able to steer the user on paths with fewer collisions and significantly reduced curvature gains. Most of the gains applied by ARC fall in the $3°/s - 5°/s$ range, showing that ARC only applies the gains necessary to avoid collisions and maintain alignment.

3.5.2.3 Average Distance Walked Between Resets

There was a significant effect of steering algorithm on the average distance walked between resets $F(1.9, 112.05) = 58.9188, p < .0001$. A plot of the average distances walked between resets for all paths with all controllers is shown in Figure 3.11. The results of post-hoc tests to determine the differences between controllers is shown in Table 3.4. The boxplot for the physical distances walked in Environment B shows that ARC achieves a higher median distance than APF and S2C, but the largest average distances afforded by ARC are not as big as the longest distances walked with APF. This suggests that APF may be more suited than ARC for navigation in environments with corridors, like Environment B, but additional studies should be conducted to confirm this.

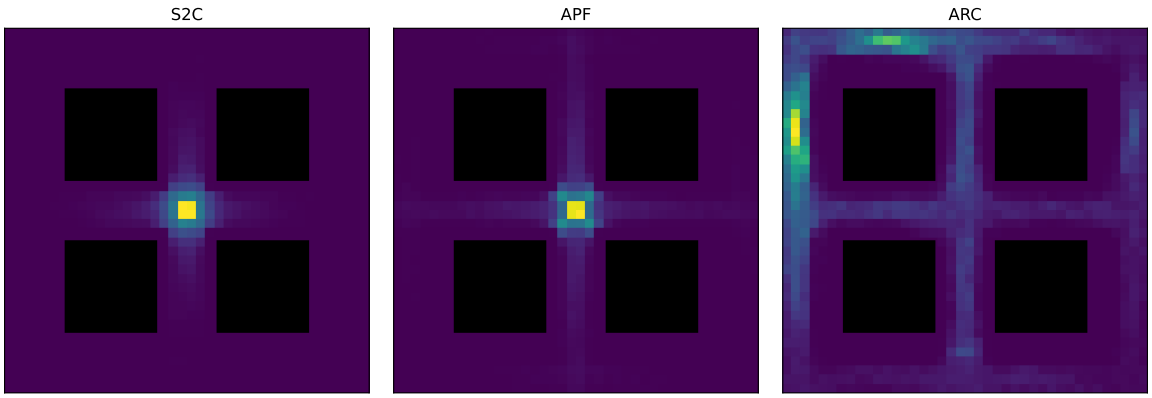


Figure 3.7: A heat map of the physical locations visited by the user in Environment B when steered with each controller. Yellow tiles indicate more visits to a region, while purple tiles indicate less time spent in a region. Obstacles are shown in black. S2C and APF keep the user concentrated near the center of the room since it is the most open space in all directions, while ARC is able to utilize more of the space and steer the user along all corridors in the room. ARC has some tendency to keep the user near the north wall of the room, which we suspect is due to the user getting stuck in between obstacles, but the exact cause is not clear.

3.5.2.4 Qualitative Evaluations

The physical position data showed differences between algorithms in where they steered the user (see Figure 3.7 for heat map visualizations). S2C and APF have very similar heat maps since both algorithms steer the user towards the center of the space. Interestingly, the user is able to visit most areas of the room using ARC, but there is a tendency to keep the user in the upper-left corner of the room. It is possible that ARC is getting stuck between obstacles. If that is the case, however, we would expect that the user gets stuck uniformly across the room due to their random starting locations, rather than getting stuck in one corner.

The average curvature gains showed a similar pattern as they did in Environment A (see Figure 3.8). S2C and ARC apply weaker curvature gains than APF. One difference between Environment A and B is that the distribution of gains applied by ARC in Environment B is much smaller than it was in Environment A, which is likely because ARC was not able to achieve

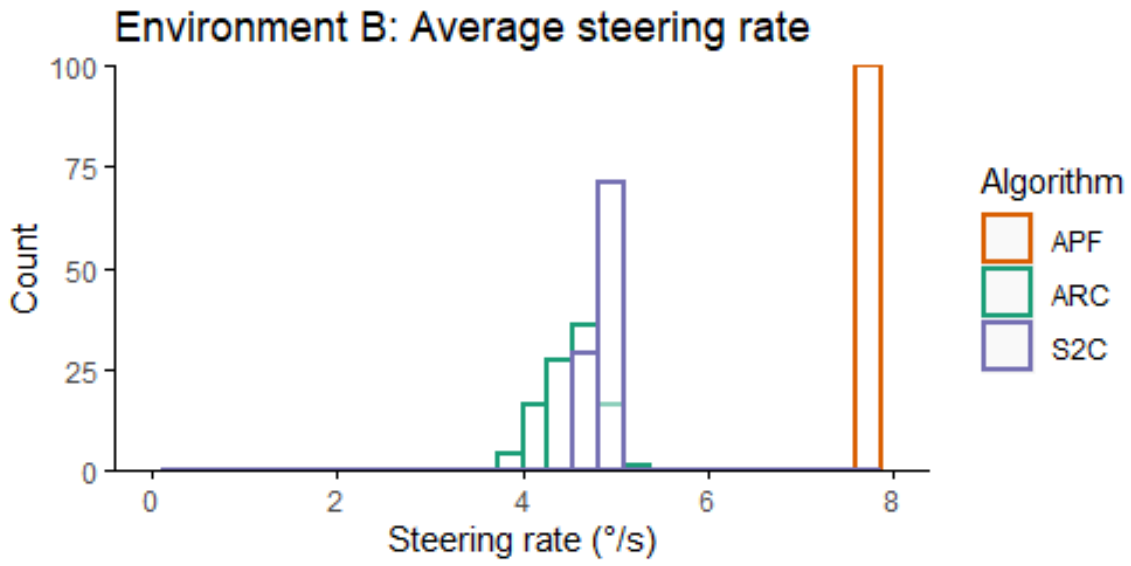


Figure 3.8: A histogram of the average curvature gain applied for each path with each controller in Environment B. As in Environment A, APF applies a constant curvature gain when the user is walking. S2C and ARC apply gains with an average in the range of $4°/s - 6°/s$, with ARC applying gains all gains at a lower intensity than about half of the gains applied by S2C. Note that the lowest gains applied by S2C are lower than those of ARC.

perfect alignment and thus was not able to apply small gains while also lowering the user's alignment score.

3.5.3 Experiment 3 (Environment C)

3.5.3.1 Number of resets

There was a significant effect of controller on the number of resets $F(1.04, 61.34) = 4186.948, p < .0001$. ARC performs dramatically better than APF and S2C, with a much smaller spread in the number of collisions. Paths steered by APF all have at least as many collisions as paths steered by ARC and are sometimes more than twice as bad as the worst path for ARC.

3.5.3.2 Average alignment

We found a significant effect of steering controller on the user's average alignment $F(1.04, 61.34) = 4186.948, p < .0001$. An interesting pattern seen in the alignment scores for Environment C is

that S2C scores the best (lowest) average alignment of all the controllers, and ARC has the highest alignment score. This is surprising because neither S2C nor APF is designed to work based on alignment, but ARC is. Even though ARC has the worst average alignment score of all the controllers in Environment C, it undoubtedly has the best performance in terms of number of collisions and physical distance travelled between resets. This disagreement in the metrics suggests that the alignment metric we used in this study may not be a good representation of an alignment-based controller's ability to keep the system aligned and avoid collisions. We stress that we *do not* believe this means the alignment-based methods ARC uses to steer the user are flawed, since all other results in this section indicate that ARC *does* work well compared to other controllers. It may simply be the case that reporting the averaged sum of the user's forward and lateral alignment is not a good way to measure a controller's alignment capabilities since the results from Environment C show that it is possible for a controller that does not use alignment to have a better alignment score. Another possible explanation for the differences in average alignment seen for Environment C is that the amount of distances we sample to compute distance to obstacles ($k = 3$, see subsubsection 3.3.1.1) may not be enough to accurately capture proximity in this environment. Since this work is only the second to formally study concepts of alignment, our alignment metric can likely be improved.

3.5.3.3 Average Distance Walked Between Resets

A robust trimmed-means ANOVA revealed a significant effect of controller on the average physical distance walked by the user between resets $F(1.52, 89.44) = 5855.824, p < .0001$. Boxplots showing the distributions of average distances walked between resets for all controllers are Environment C is shown in Figure 3.11, and the results from post-hoc significance tests are in

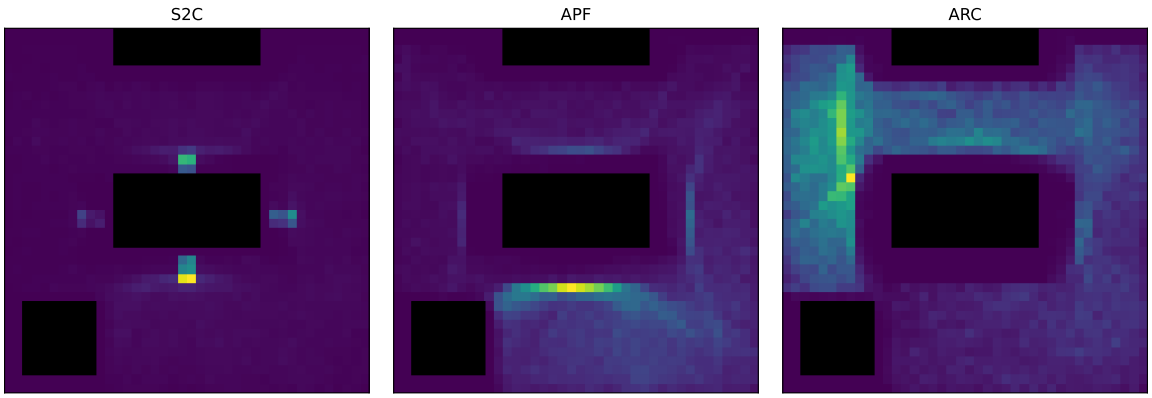


Figure 3.9: A heat map of the simulated user's location in the physical environment when exploring a virtual environment using three different redirection controllers. Yellow tiles represent a large amount of time spent in that region, and purple tiles represent a small amount of time spent in that region. The Alignment-based Redirection Controller (ARC) allows the user to utilize more of the physical space while exploring the virtual world compared to S2C and APF. This means that users spends less time being reset and more time walking through the physical environment, when steered with ARC than with S2C or APF. This is supported by the results for the number of collisions and distance walked.

Table 3.4. ARC outperforms APF and S2C, and the results for ARC are more consistent than they are for APF, though there is not as dramatic a difference as there was for the number of resets. The number of resets for Environment C shows that ARC performs much more consistently than APF, but the average distance between resets for Environment C, while it shows the same overall trend, suggests that the difference in consistency is not as large as it seemed from the number of resets, highlighting the importance of using multiple performance metrics.

3.5.3.4 Qualitative Evaluations

Upon observing the physical position heat maps (Figure 3.9), we noticed that ARC utilizes more of the PE for navigation than do APF or S2C, but there is still a bias towards the leftmost region of the room. Visual inspection of the random starting positions in the PE confirmed that the bias was not due to the starting position, so more work should be done to get a better understanding of the biases of ARC.

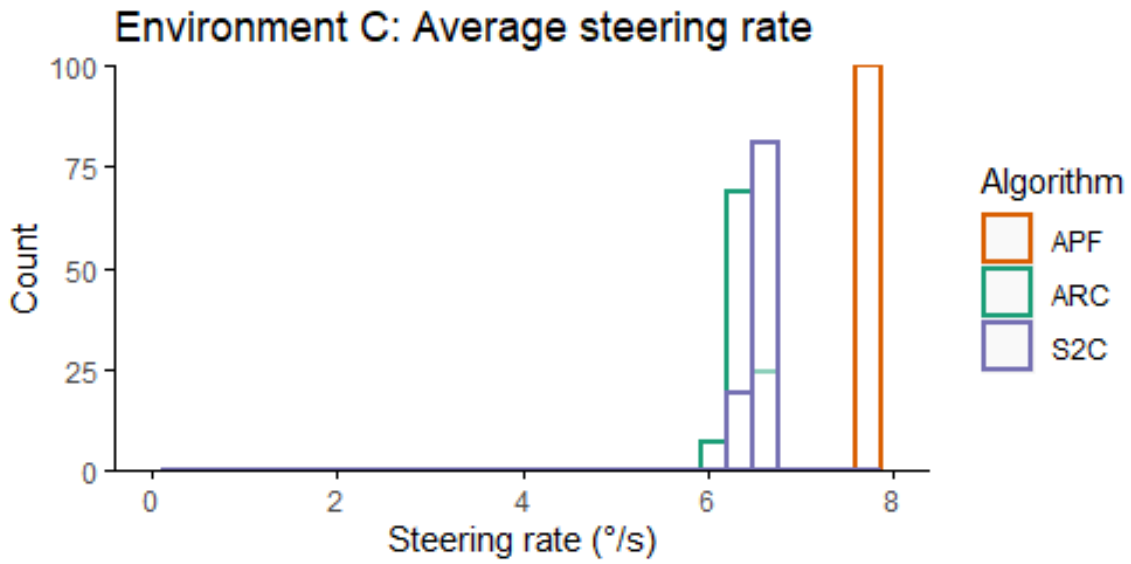


Figure 3.10: The average curvature gain applied by each controller for all paths in Environment C. The same trend as in Environment B is seen here, where APF has a higher steering rate than S2C and ARC. One difference between the steering rates in Environment B and C is that the gains applied by S2C and ARC are in a higher range ($6°/s - 7°/s$) in Environment C than they were in Environment B ($4°/s - 6°/s$).

We inspected the frequency plot of gains applied by each controller across all paths in Environment C (Figure 3.10). All gains applied by both ARC and S2C are lower than those of APF, with ARC applying the lowest gains of all, while applying stronger gains less frequently than S2C.

3.5.4 Proof of Concept Implementation

We implemented ARC in a VR system using an Oculus Quest and the Unity 2019.4.8f1 game engine. Our proof of concept implementation shows that ARC works as intended in real VR systems, but we note that a full user study should be conducted before drawing more conclusions about ARC in VR systems. In both environments that we tested, the user's virtual position started centered along the south wall of the VE, and the user was instructed to walk in a straight line forward in the VE.

The first PE/VE pair was intentionally designed to be fairly simple in order to verify that ARC steers the user as it should. The user's physical position started in the southeast corner of the PE. Once the user started walking forward, they were steered to the left, away from the east wall of the PE, using curvature gains. The user was steered away from the east wall in order to improve their alignment with the virtual state, since the virtual user was not beside any walls.

In the second PE/VE pair, the user's physical location started in the southwest corner of the physical room. Similarly to the first environment, the user was steered away from the nearby physical wall. Once their virtual position was between the two obstacles in the VE, ARC continued to steer the user to the right with curvature gains, in an effort to minimize the misalignment between the user's physical and virtual states. When the user walked past the virtual obstacle on the left, ARC steered the user slightly towards the left to improve their alignment.

The proof of concept implementation shows that our algorithm is able to run in real time without interfering with the user's ability to travel on an intended virtual path, which is a crucial requirement for redirection controllers. Additionally, ARC is simple enough such that it can be used on each frame without negatively impacting the frame rate of the system.

3.6 Discussion

We found that a redirection controller based on alignment can be a very effective alternative to traditional controllers that always try to steer users away from physical obstacles. Our novel alignment-based redirection controller, ARC, outperformed current state-of-the-art methods in all environments that we tested for all metrics except for average alignment in Environment C. In addition to being able to deliver a locomotion experience with fewer collisions and further distances walked between collisions, ARC steers the user with curvature gains that are less intense

than those applied by other controllers. ARC achieves this high performance using just three distance calculations from the VE and three from the PE, which allows it to easily run in real time. Using information from the VE has usually only been done by predictive controllers, but we were able to develop a reactive controller that leverages instantaneous information from the VE for large performance benefits, and does not require complex predictions about the user's behavior.

We also presented Complexity Ratio (CR), a new metric to measure the relative complexity of a pair of physical and virtual environments by describing the density of obstacles in the environments. The relative complexity of environments is an important factor in a controller's ability to steer the user, but we are unaware of any RDW studies that have explicitly defined and discussed any notions of relative complexity and how it affects controllers' performance. Our work presents the first step in this direction. We showed that traditional controllers tend to perform worse as the difference in complexity between the PE and VE grows. This agrees with prior observations that the shape of the environment affects a controller's performance [7, 83].

ARC comes with many advantages over traditional steering policies. First, ARC decreases the likelihood that a user experiences simulator sickness due to strong redirection. While the exact cause of simulator sickness is not known, one of the main theories is that simulator sickness arises when there is a conflict between visual, vestibular, and proprioceptive stimuli [108]. RDW creates this exact perceptual conflict, so it is not uncommon for users to feel simulator sickness when being redirected. Although we know it is safe to apply redirection within the perceptual thresholds, these thresholds will vary from user to user. Thus, the we cannot assume that commonly purported threshold values will be suitable for all users. By only applying redirection when the user is misaligned, and only applying gains at the intensity necessary to achieve alignment,

ARC redirects the user less than a traditional controller does, which decreases the likelihood of simulator sickness and creates a more comfortable experience.

Steering by alignment provides passive haptics by enabling the user to interact with the physical environment [206, 207]. Passive haptics are physical objects that provide feedback to the user through their shape and texture [125]. Passive haptics can significantly increase a user's feelings of presence and spatial knowledge transfer [93]. Passive haptics and RDW have typically been considered mutually exclusive due to their conflicting requirements. However, Kohli et al. [107] demonstrated that it is possible to combine the two if we have the appropriate environment configurations. Their demonstration was in a carefully crafted environment designed specifically to enable passive haptics. Alignment can enable passive haptics in arbitrary environments, which may allow for more immersive experiences that combine comfortable locomotion through redirection with realistic sensations through passive haptics. The efficacy of using alignment to combine passive haptics and RDW should be studied through formal user studies, since alignment currently does not consider the shape and orientation of obstacles, which are important factors for effective passive haptics [107].

3.7 Conclusions and Future Work

In this work we presented ARC, a novel controller based on alignment. Through extensive simulation-based experiments, we showed that our controller was able to outperform state-of-the-art algorithms in both simple and complex environments. Furthermore, our algorithm applied redirection gains at a lower intensity than other controllers, which reduces the chances of inducing simulator sickness and improves the usability of RDW systems for people with low RDW perceptual thresholds. We also formalized the notion of relative environment complexity between the physical

and virtual environments, which to the best of our knowledge had not yet been done. To this end, we introduced Complexity Ratio (CR), a novel metric to measure the difference in navigation complexity between the physical and virtual environments, and showed how CR influences controller efficacy.

There are many avenues for future work. The heuristics that ARC uses are fairly simple, so it is likely that a more complex algorithm will yield a better performance. For example, a finer approximation of the user state using more distance samples may yield better results. Additional work should also be done to get a better understanding of the biases that ARC exhibited, so we can better predict how a controller will perform in an environment. Extending ARC to dynamic scenes with moving obstacles or multiple users is also an interesting area for future work. Furthermore, ARC should also be evaluated with full user studies now that we know that alignment can be an effective method for redirection. Future work should also investigate ways to use concepts of alignment to combine passive haptics with redirected walking.

It is currently quite difficult to compare controllers from different researchers without implementing them oneself, since experiments are often conducted under very different conditions. The ability to compare controllers may help the community to develop new controllers more effectively, since direct comparisons will highlight the strengths and weaknesses of controllers. To enable comparisons between RDW controllers, work should be done to develop accurate performance metrics and standard benchmarks. It is likely that development of good metrics and benchmarks will require a deep understanding of the complicated interactions between the PE, the VE, the virtual path, and the controller, since any good metrics and benchmarks will need to encapsulate these interactions.

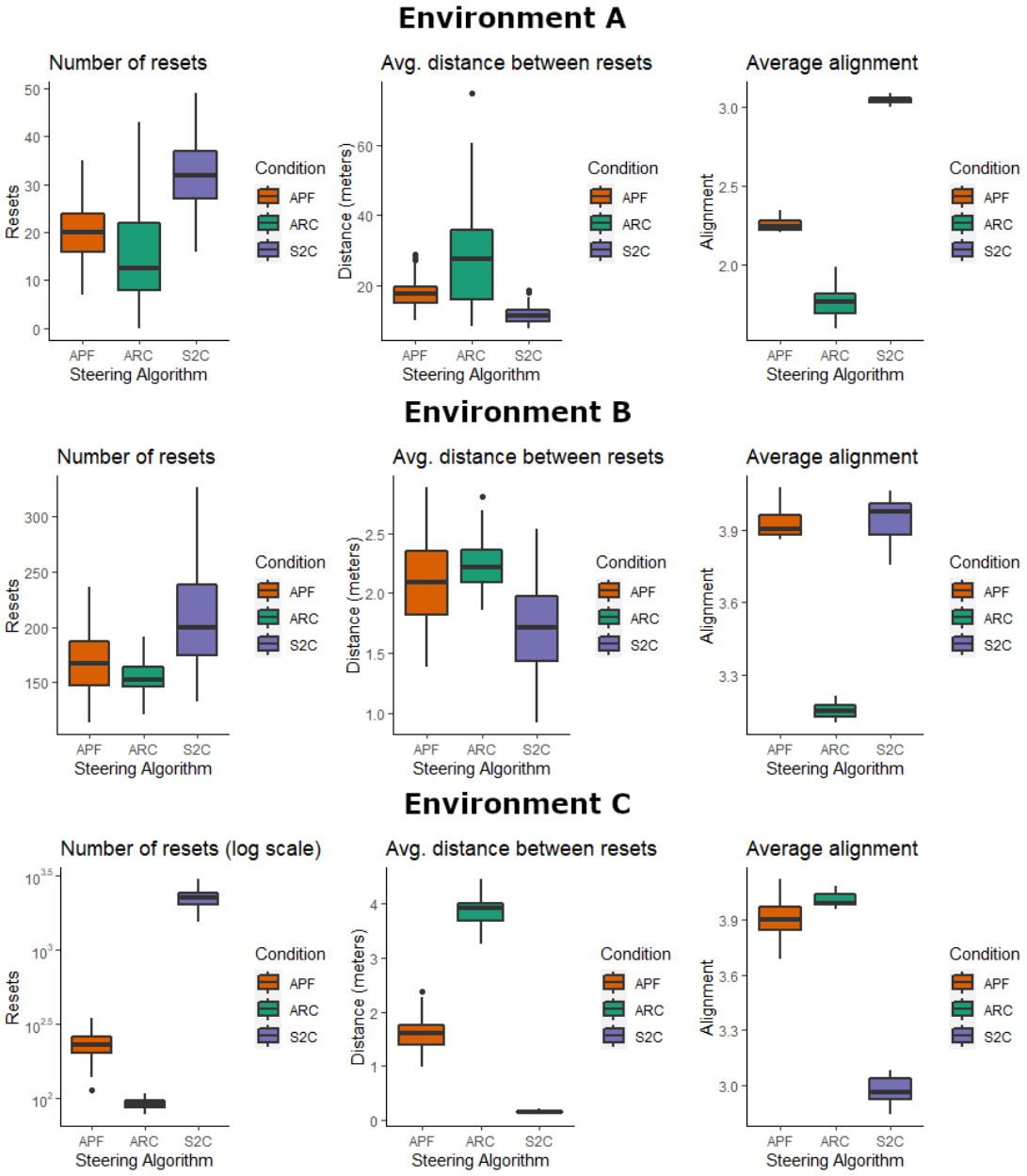


Figure 3.11: Boxplots of performance metrics for each controller in each environment. The boxplots show the median and IQR for the data. A significant difference was found between all algorithms in all environments. ARC outperformed APF and S2C for all metrics in all environments except for average alignment in Environment C.

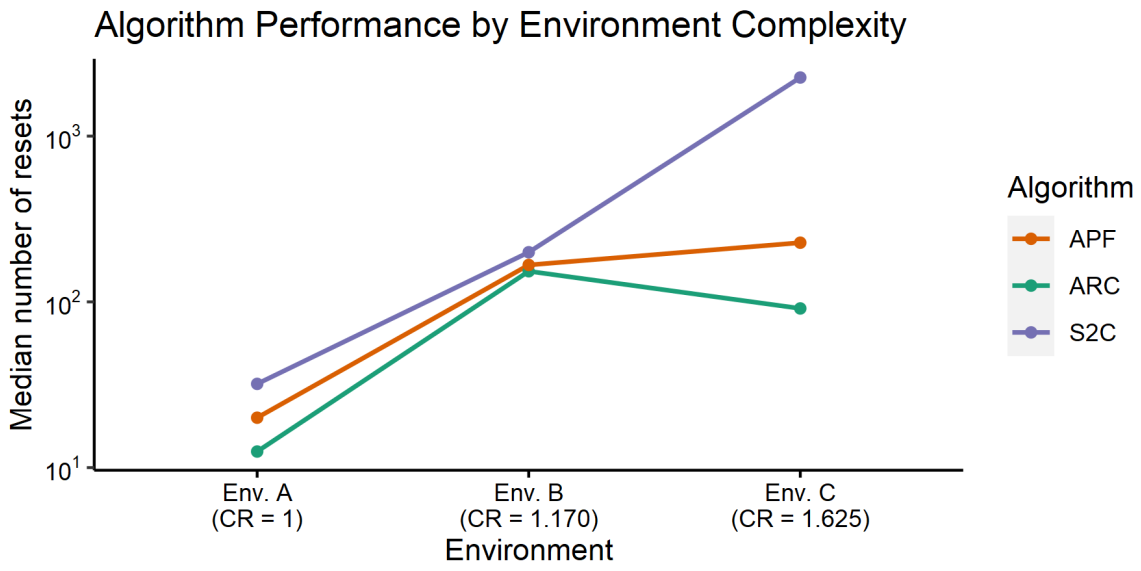


Figure 3.12: The relationship between the environment complexity and the number of resets incurred by a redirection controller. ARC consistently has a better performance than S2C and APF for all environment complexities. The performance difference between ARC and the other algorithms is quite large for environments A and C, but the difference decreases drastically for Environment B. It is not clear why Environment B causes the controllers to have a more similar performance, but it may be due to the relatively few pathing options afforded by the narrow hallways of Environment B. Environments A and C both include regions with a fairly large amount of open space, unlike Environment B (see Figure 3.4).

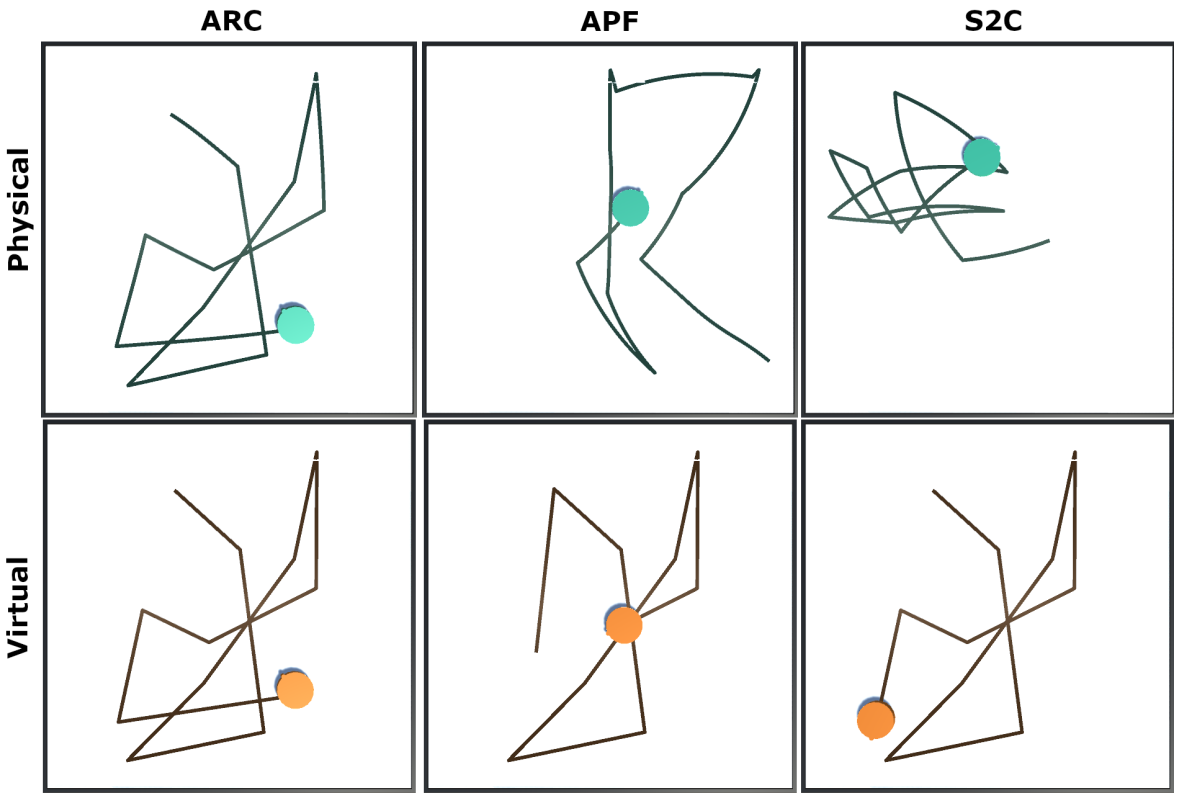


Figure 3.13: A screenshot of the user's state and recent path in Environment A for each controller. Each simulated user travelled on the same virtual path in this figure, and the screenshot was taken at the same time in the simulation. When steered with ARC, the system is able to achieve perfect alignment, and the user's physical state and recent path matches the virtual counterpart. APF and S2C are not able to achieve alignment, and their paths and states are very dissimilar to the virtual counterparts. The state of the virtual user is not the same across all conditions because the virtual user pauses while the physical user reorients after a collision, and each controller incurred a different number of collisions.

Chapter 4: Visibility-Based Redirection

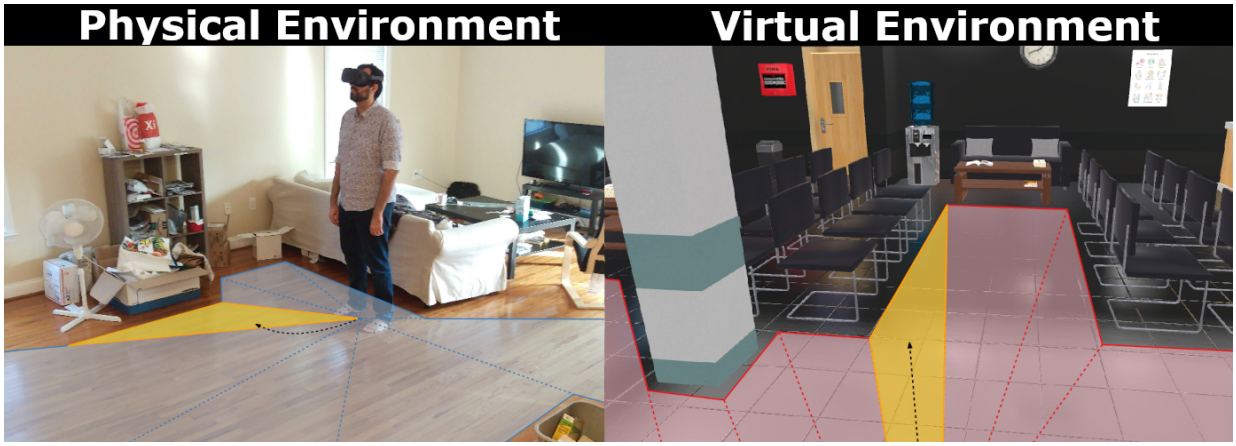


Figure 4.1: A visualization of the geometric reasoning that our redirection controller performs on every frame in order to steer the user in the physical space. First, the controller computes the visibility polygon for the user's physical and virtual locations (the regions bounded by the blue and red edges, respectively). Next, the controller computes the region of space (part of the red visibility polygon) in front of the user that the user is walking towards in the virtual environment (yellow region in the right image). By comparing the areas of the regions, our controller computes the region in the physical space (yellow region in the left image) that is most similar to the virtual region the user is heading towards. Finally, the controller applies redirected walking gains to steer the user to walk towards the highlighted region in the physical space. Black dashed arrows indicate the user's trajectory in the environment. Our algorithm yields significantly fewer resets with physical obstacles than prior algorithms.

In this chapter, we present a new approach for redirected walking in static and dynamic scenes that uses techniques from robot motion planning to compute the redirection gains that steer the user on collision-free paths in the physical space. Our first contribution is a mathematical framework for redirected walking using concepts from motion planning and configuration spaces. This framework highlights various geometric and perceptual constraints that tend to make collision-free redirected walking difficult. We use our framework to propose an efficient solution to the

redirection problem that uses the notion of visibility polygons to compute the free spaces in the physical environment and the virtual environment. The visibility polygon provides a concise representation of the entire space that is visible, and therefore walkable, to the user from their position within an environment. Using this representation of walkable space, we apply redirected walking to steer the user to regions of the visibility polygon in the physical environment that closely match the region that the user occupies in the visibility polygon in the virtual environment. We show that our algorithm is able to steer the user along paths that result in significantly fewer resets than existing state-of-the-art algorithms in both static and dynamic scenes.

4.1 Introduction

Natural walking as a means to explore virtual environments (VEs) is generally preferred over other artificial locomotion interfaces such as flying [209] or teleportation since natural walking improves the user's sense of presence and task performance in the VE. Redirected walking (RDW) [157] is a locomotion interface that affords natural walking in virtual reality by imperceptibly steering the user along physical paths that differ from their virtual counterparts. Over the years, many researchers have developed different algorithms (known as *RDW controllers*) that apply RDW to steer the user to avoid collisions with physical obstacles that they cannot see or detect. If a collision is imminent, the RDW system applies a "reset" to reorient the user away from the nearby obstacle. Although significant progress has been made, even the best controllers are unable to guarantee a reset-free experience in arbitrary static and dynamic environments.

Redirection controllers generally fall into one of three categories: reactive, predictive, or scripted [140]. Reactive controllers steer the user based only on information available at the current or previous frames. Predictive controllers steer the user based on the user's predicted

future movement in the VE, and scripted controllers steer the user as they walk along a predefined path in the VE. Reactive controllers are usually preferred since they can be deployed in arbitrary environments and often require little to no additional setup (such as environment preprocessing). However, their ability to steer the user away from obstacles is usually lower than that of predictive or scripted controllers since reactive controllers must operate using less information. Predictive controllers can outperform reactive controllers, but they rely on accurate predictions of the user's movements. Recently, a new paradigm of redirection controllers based on the concept of alignment has emerged. Instead of steering the user away from physical obstacles, alignment-based controllers steer the user in an attempt to minimize the difference between the physical and virtual environments. These controllers have been shown to be effective for lowering the number of resets [231] and enabling passive haptics for increased immersion [206, 207].

Much of the prior work on RDW controllers focuses on users exploring static physical and virtual environments. Some work that instead focuses on dynamic scenes mostly looks at multi-user RDW systems [8, 11, 54]. Chen et al. [33] proposed ideas on how redirection controllers can steer the user away from moving physical obstacles. Though there has been work on dynamic physical scenes as well as multi-user systems, there are no known solutions that work well in all dynamic environments with no assumptions on obstacles' motions.

Main Results: We provide a mathematical framework for the redirected walking problem, as well as a novel approach to redirected walking in static and dynamic environments. We frame the redirection problem as optimizing the user's path in the physical environment by applying the appropriate redirection at each frame. Our formulation makes it easy to develop effective controllers and conduct rigorous analyses to better understand a redirection controller's behavior. Using this formulation of the redirection problem, we also introduce a RDW controller that uses

a novel alignment metric based on visibility polygons. That is, given the layouts of the physical and virtual environments and the user's location in them, our controller computes the region of walkable space visible from the user's position in both environments. With these representations of the local space that is available to the user, our controller steers the user towards a region in the physical space that is most similar to the region they are approaching in the virtual space. We found that our controller is able to significantly reduce the number of times that the user has to reset their orientation when they come too close to physical obstacles. Our main contributions in this work include:

- A mathematical framework for the redirected walking problem based on concepts from robot motion planning. We formulate it as an optimization problem, where the aim is to transform a virtual path to an optimal physical path, subject to constraints depending on the user and the information available.
- A novel redirected walking controller that steers the user based on alignment, with heuristics derived from visibility polygons. Our algorithm achieves significantly fewer resets than the current state-of-the-art controllers.
- Comparisons of our algorithm to the state-of-the-art algorithms in both static and dynamic simulated, single-user scenes.

4.2 Prior Work and Background

Locomotion is a fundamental problem in virtual reality (VR) since the user wishes to explore a VE that is usually much larger than the physical environment (PE) they are in. Furthermore, the locations of obstacles in the PE are usually not the same as the locations of obstacles in the

VE. This creates a problem when the user wishes to travel along a collision-free virtual path but doing so may force them to walk into unseen physical obstacles. Many different techniques for locomotion have been developed [94, 149, 157, 188, 198, 209], but in this work we focus on the RDW technique developed by Razzaque et al. [158]. A recent review of VR locomotion interfaces can be found in [126].

By slowly rotating or translating the VE around the user while they walk, RDW allows users to explore virtual environments while located in smaller physical environments [157]. These rotations and translations are controlled by parameters called *gains*. In order to remain on their intended virtual path, users will subconsciously adjust their physical path to counteract the VE movements. The three main gains are translation, rotation, and curvature gains. Translation gains translate the VE around the user while they walk, causing their physical path to be longer or shorter than their virtual path, depending on the direction of the VE translation. Rotation gains rotate the VE around the user while they turn in place, which causes their physical rotation to be larger or smaller than their virtual rotation one, depending on the direction that the VE rotates. Similarly, curvature gains rotate the VE around the user while they are walking, which causes them to veer on a physical path with a different curvature than their virtual one. The faster the VE rotates or translates, the more the user's physical path will deviate from their virtual path, and the easier it will be to steer the user away from physical obstacles. However, it is important that the gains are not large enough that the user can perceive the rotations since this will make it harder to explore and can lead to simulator sickness [187]. Other gains such as bending gains [112] have been developed, but they are less well-understood than translation, curvature, and rotation gains so we do not consider them in this work.

4.2.1 Redirection Controllers

A redirection controller is an algorithm that applies RDW gains to steer a user along a physical path, with the intention of minimizing the number of times the user collides with a physical obstacle [140]. A controller consists of a *steering* component and a *resetting* component. As the name suggests, the steering component is responsible for applying gains to steer the user in the physical space while walking. The resetting component is responsible for initiating a "reset," wherein the user's virtual movements are disabled until they turn in place to reorient themself in the physical world. Resets are initiated when the user gets too close to any physical obstacle.

Many redirection controllers operate on different assumptions and with different amounts of information available to them. Consequently, controllers have traditionally been classified as reactive, predictive, or scripted [140]. Reactive controllers steer the user according to the information available at the current frame or any prior frames. These controllers are typically designed to function in arbitrary environments, with little to no pre-processing or setup required. Examples of reactive controllers include steer to center [157], steer to orbit [83, 157], steer to multiple targets [157], and controllers based on reinforcement learning [29, 117, 192] or artificial potential fields [11, 132, 205]. Predictive controllers steer the user according to the information available on the current/prior frames and a prediction of the user's future path in the VE. These types of controllers can perform better than reactive controllers, but their performance depends on the accuracy of the path prediction. Examples of predictive controllers include those developed by Zmuda et al. [243], Nescher et al. [136], and Dong et al. [54]. Scripted controllers are controllers that steer the user as they travel along designated paths in the VE [6, 241]. Scripted controllers usually result in the fewest resets, but they require the researchers to design and

plan the environments carefully. As a result, these controllers cannot be generalized to arbitrary environments.

Although the RDW community has developed this taxonomy for redirection controllers, more recent controllers that involve more complex algorithms or use new types of information are not easily categorized as reactive, predictive, or scripted. Thomas et al. [206, 207] and Williams et al. [231] introduced redirection controllers that leverage the concept of alignment. *Alignment* is the concept of comparing the physical and virtual environments according to some environment feature(s) (such as the location of a user relative to an object). In other words, alignment measures the similarity of the two environments according to these environment features. It should be noted that other researchers have used ideas similar to alignment by editing the VE to match the physical one [174] or by making assumptions about the user's motion based on the VE [243]. The alignment-based controllers developed by Thomas et al. and Williams et al. steer the user with RDW gains, but the steering decisions are guided by the degree of similarity (alignment) of the user in the physical and virtual environments. Furthermore, some controllers make implicit assumptions about the virtual reality system and incorporate these assumptions into the steering policy. Williams et al. [231] assume that the user walks along a collision-free path in the VE, while controllers based on reinforcement learning train an algorithm that implicitly learns a model of the user's locomotion behavior and steers the user according to this model.

4.2.2 Motion Planning and Visibility Polygons

Motion planning is the problem of computing a collision-free path through an environment that takes an agent (traditionally, a robot) from its initial configuration to a goal configuration (Section 2.3). In our approach, we use techniques from motion planning literature to compute

collision-free paths for the user in the physical space and the virtual space. In particular, we perform geometric reasoning in each of these 2D spaces by using *visibility polygons*. The visibility polygon for a point p is the set of all points in the plane that are visible from p. The visibility polygon thus encapsulates the entire region of space that has line-of-sight visibility from the point p. Depending on the layout of the space, the visibility polygon may be unbounded. If we consider a human (or robot) observer located at position p, the visibility polygon can be thought of as the entire region of space that the observer can see. The visibility polygon can be used to compute the free space corresponding to a point robot in an environment with polygonal obstacles [73]. Thus, the visibility polygon provides a well-defined region that we can use to compute a local path in the environment to move the user closer to their goal without any collisions.

Since human locomotion is largely dominated by the information that is immediately available to the person [133], we use the visibility polygon to perform geometric reasoning in a user's local surroundings. In the architectural design literature, researchers have used visibility polygons (which they refer to as "isovists") to describe an environment and then studied how people's locomotion patterns change as the environment structure changes [14]. Wiener et al. [228] showed that the complexity of the visibility polygon (characterized by its jaggedness) was correlated with an observer's task performance and locomotion speed in the environment. Christenson et al. [38] introduced occlusion maps, which are maps that define the regions visible over multiple points in an environment, to study how the layout of an environment changes as the observer moves to a new position. Within the RDW community, Zank et al. [242] used visibility polygons to predict the areas of the VE that the user might walk towards next.

4.3 Redirected Walking Using Visibility Polygons

To aid in the exposition of our redirection controller, we begin with formal definitions and a mathematical formulation of the RDW problem. A formal characterization of the RDW problem provides us with a framework that we can use to conduct more rigorous analyses of and reasoning about RDW steering algorithms. Note that the definitions we use are adapted from those used in the robot motion planning literature [115].

4.3.1 Definitions and Notation

In virtual reality, the user is simultaneously located in a PE, E_{phys}, and a VE, E_{virt}. For each environment, the user state describes their location p and heading direction θ in the environment. We denote the user's state as the pair $q = \{p, \theta\}$. The user's state at a particular time t is denoted by $q^t = \{p^t, \theta^t\}$. Thus, the user's state at time t in E_{phys} is denoted by $q^t_{phys} = \{p^t_{phys}, \theta^t_{phys}\}$. The path that a user walks along in an environment is represented by an ordered set of states $Q = \{q^0, q^1, ..., q^t\}$. For brevity, we denote the physical and virtual paths as $path_{phys} = \{q^0_{phys}, ..., q^t_{phys}\}$ and $path_{virt} = \{q^0_{virt}, ..., q^t_{virt}\}$, respectively.

The user's state in an environment is also referred to as their configuration. The configuration space $\mathcal{C}$ (or $\mathcal{C}$-space for short) is the set of all possible configurations of the user. Some configurations in the $\mathcal{C}$-space correspond to the user colliding with an obstacle in the environment. The obstacles in an environment occupy the obstacle region, $\mathcal{O} \subset E$. The set of all colliding configurations q is the obstacle space $\mathcal{C}_{obs}$:

$$\mathcal{C}_{obs} = \{q \in \mathcal{C} \mid q \cap \mathcal{O} \neq \emptyset\}. \tag{4.3.1.1}$$

That is, the obstacle space $\mathcal{C}_{obs}$ is the set of all configurations q for which the user intersects with an obstacle in the obstacle region $\mathcal{O}$. The free space $\mathcal{C}_{free}$ is all of the other configurations that

are not in the obstacle region. We define the free space as $\mathcal{C}_{free} = \mathcal{C} \setminus \mathcal{C}_{obs}$. The free space and obstacle space of the PE are denoted $Free_{phys}$ and Obs_{phys}, while the free and obstacle spaces of the VE are denoted $Free_{virt}$ and Obs_{virt}. Finally, note that $E_{phys} = Free_{phys} \cup Obs_{phys}$ and $E_{virt} = Free_{virt} \cup Obs_{virt}$.

4.3.2 Redirected Walking and Configuration Spaces

Redirected walking is typically implemented by rotating the VE around the user as they walk, so it is natural to think about a redirection controller as an algorithm that rotates the VE according to some criteria on every frame. Instead, our goal is to frame redirection controllers in terms of the simultaneous computation of collision-free trajectories in the physical and virtual spaces. We can visualize this as superimposing the user's virtual path onto their physical location and applying RDW gains that transform the superimposed path such that the user avoids any obstacles that the path intersects with (see Figure 4.2). In our figures, black shapes represent obstacles, white space is any walkable region in the environment, and the colored regions represent the free space (*i.e.* the subset of the walkable region visible from the user's position).

With our goal in mind, we can now formally describe the redirection problem using the definitions from Subsection 4.3.1. Given two environments E_{phys} and E_{virt}, the user's configuration in both environments are q_{phys} and q_{virt}, respectively. The user explores E_{virt} by following a collision-free path $path_{virt} \in Free_{virt}$. This path corresponds to some physical path $path_{phys} \in E_{phys}$, for which it is possible that $path_{phys} \cap Obs_{phys} \neq \emptyset$. If $path_{phys} \cap Obs_{phys} \neq \emptyset$, we wish to find some new optimal path $path^*_{phys}$ such that $path^*_{phys} \cap Obs_{phys} = \emptyset$, i.e. $path^*_{phys} \in Free_{phys}$. Let `RDW(`$path$`)` be a generic redirection function which applies translation (g_t), curvature (g_c), or rotation (g_r) gains at each timestep $q^t \in path$ to yield a new path $path^*$.

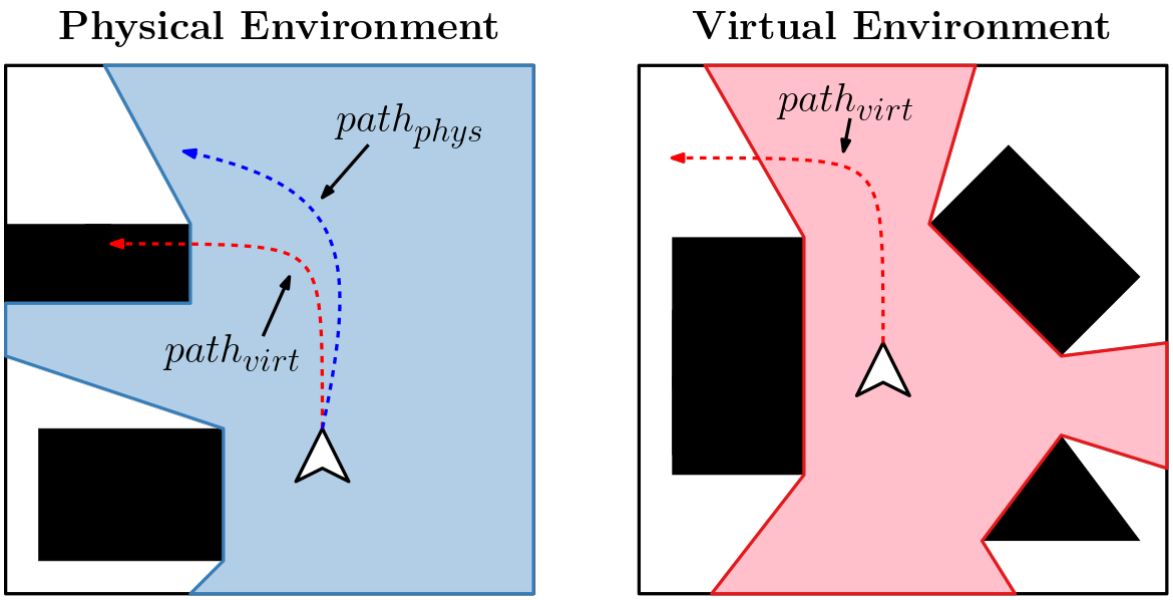


Figure 4.2: Visualization of the redirected walking problem. If the user tries to walk on the virtual path $path_{virt}$ with no redirection applied, they will collide with the obstacle to their left in the physical space. After applying redirection, the user instead walks along $path_{phys}$ and avoids any collisions. The free spaces $Free_{phys}$ and $Free_{virt}$ are shown in blue and red, respectively.

The redirection problem is thus to find some best function `RDW*()` such that `RDW*(`$path_{virt}$`)` $= path^*_{phys}$. We seek to develop a redirection controller that executes `RDW*()`. `RDW()` is subject to multiple constraints:

1. **Maximum redirection constraint:** Since RDW is limited by human perception, the gains applied by `RDW()` must be bounded by the user's empirically measured perceptual thresholds. This limits how much redirection can be done at any given moment and makes it more difficult to avoid Obs_{phys}.

2. **Geometric deviation constraint:** `RDW(`$path_{virt}$`)` results in a path $path_{phys}$ that is geometrically different from $path_{virt}$, where translation gains change the length and rotation and curvature gains change the curvature of $path_{phys}$ relative to $path_{virt}$. Stronger gains correspond to a greater deviation between $path_{phys}$ and $path_{virt}$. Stronger redirection gains have a higher

chance of inducing simulator sickness in the user, so we wish to compute $path_{phys} \in Free_{phys}$ with the weakest gains possible. Geometrically, this is defined as the path with the lowest deviation from $path_{virt}$.

3. **Information constraint:** Oftentimes, the user's future virtual path $path_{virt}$ is not known, which makes optimizing $path_{phys}$ difficult. Additionally, the user's PE may not be completely known, depending on the capabilities of the virtual reality system.

Due to the above constraints, developing the perfect redirection function `RDW*()` for all $path_{virt}$ is very difficult.

Since it is difficult to develop `RDW*()`, we must rely on our `RDW()` function in conjunction with a resetting function `reset()`. The resetting function is responsible for prompting the user to stop walking and reorienting them to a safe configuration in E_{phys}. Similarly to `RDW()`, `reset()` applies redirection gains to transform $path_{virt}$ such that the resulting physical path is valid, i.e. $path_{phys} \in Free_{phys}$. However, `reset()` differs from `RDW()` in that `reset()` alters $path_{virt}$ in order to yield a safe $path_{phys}$. In practice, this is performed by having the user turn $360°$ in place in E_{virt}, while reorienting by some smaller angle in E_{phys} due to rotation gains. Thus, `reset()` inserts configurations into $path_{virt}$ that cause the user to turn in place in E_{virt} while applying rotation gains g_r to alter the user's corresponding physical rotation.

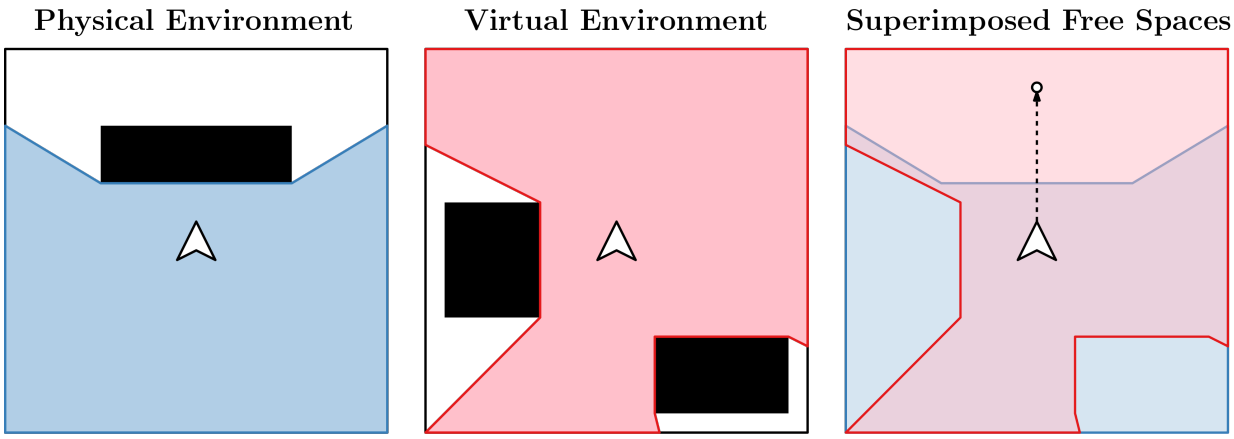
4.3.3 Finding `RDW*()` Using Visibility Polygons

In this section, we present a new redirection controller that uses the notion of visibility polygons to compute `RDW()`. Our goal is to steer the user along a path in $Free_{phys}$. The visibility polygon computed in E_{phys} at the user's position p_{phys} is a representation of $Free_{phys}$ that can be computed in $O(n\ logn)$ time [49, 199], where n is the number of segments that define the

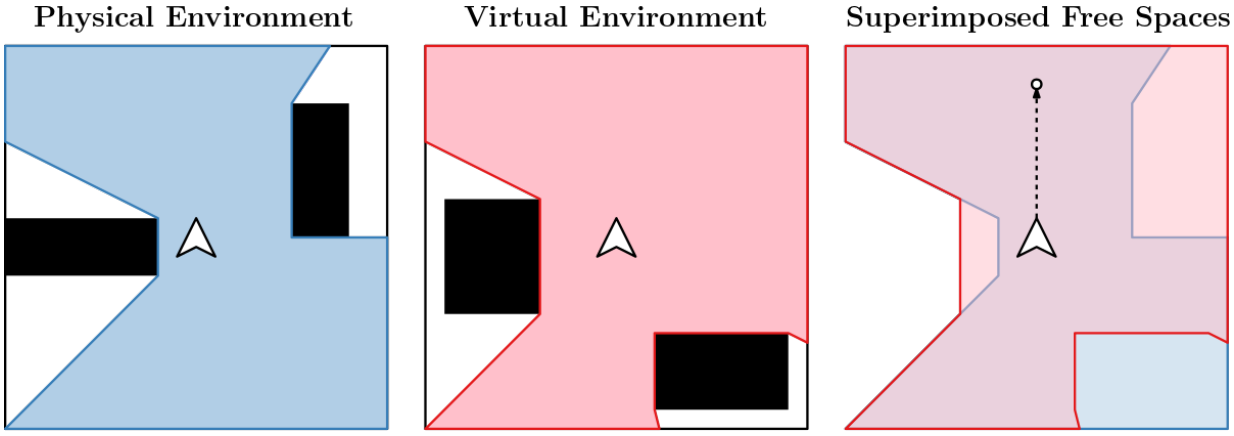
boundaries of obstacles in the environment. Thus, in order to take advantage of motion planning techniques, our controller is based on the visibility polygon.

Considering the problem statement in Subsection 4.3.2, our controller's goal is to apply a function `RDW(`$path_{virt}$`)` $= path_{phys}$ such that $path_{phys}$ is in the physical visibility polygon (which represents $Free_{phys}$). Our redirection controller (which executes `RDW()`) is not predictive, so it does not have access to the user's future virtual path $path_{virt}$. This makes it effectively impossible to compute the optimal physical path $path^{*}_{phys}$ as described in Subsection 4.3.2, since we cannot directly transform $path_{virt}$ to make it lie within $Free_{phys}$. To resolve this, we reframe the redirection problem slightly and use alignment in our redirection function `RDW()`. Instead of superimposing $path_{virt}$ onto E_{phys}, we superimpose $Free_{virt}$ onto $Free_{phys}$, centered on the user (both $Free_{virt}$ and $Free_{phys}$ are represented by visibility polygons). When the user walks in E_{virt}, their path in the superimposed $Free_{virt}$ will correspond to some path within E_{phys}, and hopefully within $Free_{phys}$. Our controller's goal is now to apply redirection such that $Free_{phys}$ is transformed to match $Free_{virt}$ as closely as possible. The intuition here is that the more similar $Free_{phys}$ is to $Free_{virt}$, the higher the chance that the user's next virtual configuration q^{t+1}_{virt} will correspond to a valid physical configuration $q^{t+1}_{phys} \in Free_{phys}$. This process of superimposing $Free_{phys}$ and $Free_{virt}$ is shown in Figure 6.3.

We guide this free space matching process using our alignment metric, which we define as the area of the free space in front of the user (see subsubsection 4.3.3.3). Note that while we framed the problem as transforming $Free_{phys}$, in implementation, we simply apply RDW gains to change the user's physical configuration such that their new $Free_{phys}$ is more similar to $Free_{virt}$. The pseudocode that our redirection controller executes on every frame is shown in algorithm 1. We now provide details of each step of the algorithm.



(a) Configurations in physical and virtual environments that lead to a collision. After the user walks forward (rightmost image), they are still in $Free_{virt}$ but they are no longer inside $Free_{phys}$, indicating that there was a collision with a physical obstacle.



(b) Free space configurations in physical and virtual environments that do not lead to a collision. After the user walks forward (rightmost image), they are still within $Free_{virt}$ *and* $Free_{phys}$, indicating that there was no collision along the physical path that the user travelled.

Figure 4.3: A visualization of the superimposition of the two free spaces, $Free_{phys}$ (blue) and $Free_{virt}$ (red). Regions of $Free_{virt}$ that do not overlap with $Free_{phys}$ signify regions of the virtual environment that the user cannot walk to without colliding with a physical obstacle. Our controller aims to steer the user in E_{phys} such that $Free_{phys}$ and $Free_{virt}$ overlap in the region that the user is walking towards.

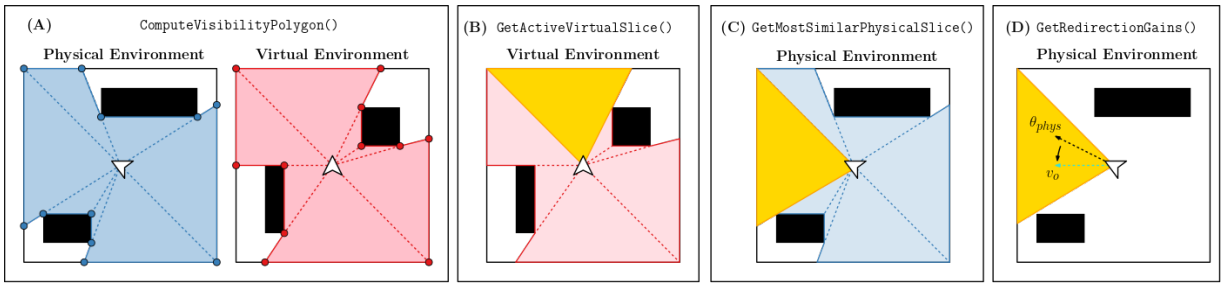


Figure 4.4: An overview of our redirection controller based on visibility polygons. **(A)** We compute the visibility polygon corresponding to the user's position in both the physical (blue) and virtual (red) environments. After the visibility polygons are computed, they are divided into regions called "slices" which we use later in our approach to measure the similarity of the two polygons. **(B)** The "active slice" in the virtual environment is computed. This is the slice of the virtual visibility polygon that the user is walking towards (shown in yellow). **(C)** The corresponding slice in the physical environment that is most similar to the active slice is computed. Similarity is measured using slice area. **(D)** Redirected walking gains are applied according to the user's heading to steer them in the direction of the most similar physical slice that was computed in step (C).

Algorithm 1 Redirection Gain Computation

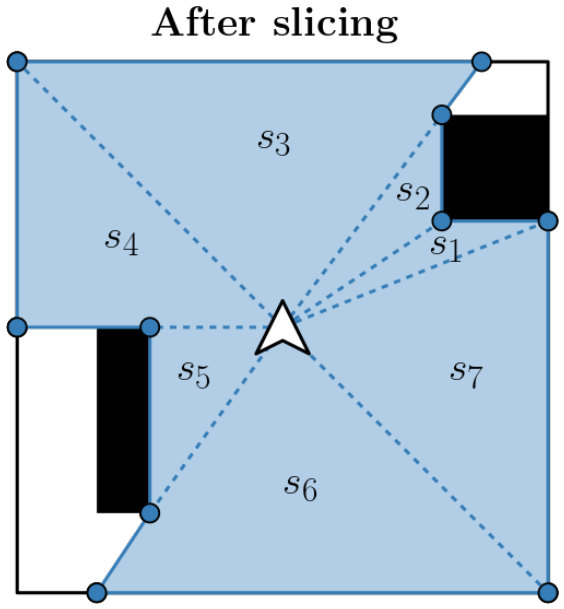
Result: Redirection gains g_r, g_t, g_c to apply on the current frame.

P_{phys} = `ComputeVisibilityPolygon` $(E_{phys}, p_{phys}, \theta_{phys})$
P_{virt} = `ComputeVisibilityPolygon` $(E_{virt}, p_{virt}, \theta_{virt})$
s^{virt} = `GetActiveVirtualSlice` $(P_{virt}, \theta_{virt})$
s^{phys} = `GetMostSimilarPhysicalSlice` (P_{phys}, s^{virt})
g_r, g_t, g_c = `GetRedirectionGains` $(s^{virt}, s^{phys}, \theta_{phys})$

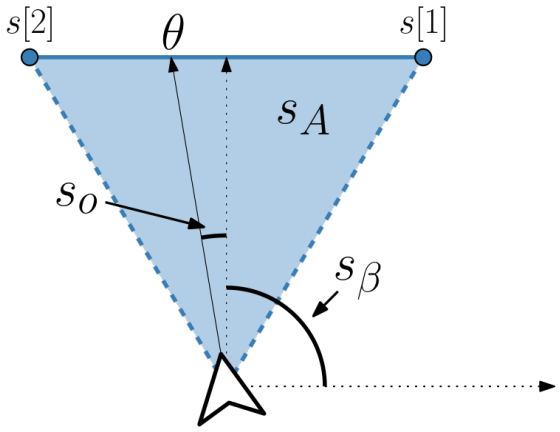
4.3.3.1 `ComputeVisibilityPolygon` (E, p, θ)

Given an environment E and a position p and heading θ in that environment, we compute the visibility polygon using the algorithm described by Suri et al. [199] (note that θ is not used in this initial computation). The visibility polygon P is defined by a kernel k and a set of vertices $\{v_0, v_1, ..., v_{n-1}\}$. Here, k is the position of the observer (i.e., the position p in the environment E). The set of vertices defines the edges of P, where consecutive vertices v_i and v_{i+1} form an edge.

Once P is computed, it is divided into "slices" s_i around k according to the order and position of the vertices of P. The vertices are sorted in counterclockwise order around k,



(a) A visibility polygon, its vertices, and its slices $\{s_1, s_2, ..., s_7\}$.



(b) A diagram showing some of the components that are computed for every slice s. Note that we do not show the average length s_l or the kernel position $s[0]$ in order to reduce visual clutter.

Figure 4.5: A visibility polygon after its slices are computed (Figure 4.5a) and the composition of one slice (Figure 4.5b).

and slices are defined as the triangles formed by triplets of points $\{k, v_i, v_{i+1}\}$. If the points $\{k, v_i, v_{i+1}\}$ are colinear, the slice is instead defined by the next vertex $v_{i+x} \in P$ such that $x > 1$ and $\{k, v_{i+1}, v_{i+x}\}$ are not colinear. A diagram of these slices is shown in Figure 4.5a.

For each slice s, the following attributes are computed:

- **Slice vertices:** The vertices $\{s[0], s[1], s[2]\}$ that define the slice. Note that $s[0] = k$ and $\{s[1], s[2]\} \subset P$.

- **Slice bisector (s_β):** The angle $\phi \in [0, 2\pi)$ that bisects angle $\angle s[1]s[0]s[2]$.

- **Average length (s_l):** The average length of the two line segments connecting $s[1]$ and $s[2]$ to $s[0]$:

$$s_l = \frac{||s[0] - s[1]|| + ||s[0] - s[2]||}{2}. \tag{4.3.3.1}$$

- **Angle offset (s_o):** The angular distance between the user's heading and the slice bisector:

$$s_o = |\theta - s_\beta|. \tag{4.3.3.2}$$

- **Slice area (s_A):** The area of the triangular slice s.

The composition of a slice is shown in Figure 4.5b. The polygon P, and all of its slices, are returned at the end of `ComputeVisibilityPolygon`(E, p, θ).

4.3.3.2 `GetActiveVirtualSlice`$(P_{virt}, \theta_{virt})$

We define the "active slice" as the slice that the user is walking towards in the VE. For this work, we assume that the user walks in the direction θ_{virt} that they are facing. Thus, the active slice s^{virt} is defined as:

$$s^{virt} = \arg\min_{s \in P_{virt}} |s_\beta - \theta_{virt}|. \tag{4.3.3.3}$$

We return s^{virt} at the end of `GetActiveVirtualSlice`$(P_{virt}, \theta_{virt})$.

4.3.3.3 `GetMostSimilarPhysicalSlice`(P_{phys}, s^{virt})

This function computes the slice $s^{phys} \in P_{phys}$ that is most similar to s^{virt}. Here, we leverage our similarity metric to measure the alignment of slices in P_{phys} to the active virtual slice s^{virt}. Our slices are triangles, so we wish to compute a similarity metric that accurately measures how similar two triangles are. Furthermore, the slices represent regions of free space that the user can walk in, so an ideal similarity metric would be able to compare two slices according to their similarity with regards to both shape *and* navigability. Measuring shape similarity is a very well-studied problem in geometric computing [218]. Likewise, the relationship between an environment's structure and navigation in that environment is well-studied [43, 80, 133]. Our goal is to design a metric that can compute the similarity of shapes with respect to the geometric

structure as well as navigability. Therefore, we use a geometric measure of similarity (slice area), but we constrain the set of physical slices that we consider according to perceptual heuristics that tend to guide human locomotion (we only consider slices in the user's field of view). We chose slice area over other shape similarity measures since we are primarily concerned with the user's proximity to Obs_{phys}, which is described by the slice's total area.

Given the physical visibility polygon P_{phys}, we first compute the set of eligible slices, which we will compare against s^{virt}. This set, denoted $S^{\dagger}$, is defined as all physical slices for which the slice's bisector is less than 90° away from the user's physical heading (this value is computed when P_{phys} is constructed):

$$S^{\dagger} = \{s \in P_{phys} \mid s_o < \frac{\pi}{2}\}. \tag{4.3.3.4}$$

Once $S^{\dagger}$ is computed, the physical slice that matches s^{virt} most closely is simply the slice with the area closest to the area of s^{virt}:

$$s^{phys} = \arg\min_{s \in S^{\dagger}} |s_A - s_A^{virt}|. \tag{4.3.3.5}$$

This slice s^{phys} is returned at the end of the function.

4.3.3.4 `GetRedirectionGains`(s^{virt}, s^{phys}, θ_{phys})

We have now computed the region $s^{phys} \in E_{phys}$ that is most similar to the region $s^{virt} \in E_{virt}$ that the user is heading towards in virtual reality. The final step is to set the rotation g_r, curvature g_c, and translation g_t gains to steer the user towards s^{phys}.

An optimal direction vector, v_o, is defined as:

$$v_o = \texttt{unit\_vector}(s_{\beta}^{phys}). \tag{4.3.3.6}$$

The `unit_vector(θ)` function returns the vector $[\cos\theta, \sin\theta]^T$. If the user is rotating in place, we apply a rotation gain g_r. We set g_r according to the following rule:

$$g_r = \begin{cases} minRotationGain & \text{user is turning away from } v_o, \\ maxRotationGain & \text{user is turning towards } v_o, \end{cases} \tag{4.3.3.7}$$

where $minRotationGain = 0.67$ and $maxRotationGain = 1.24$ [187].

If the user is walking, we apply translation and curvature gains to steer them in the direction of v_o. Specifically, we set the curvature gain g_c as:

$$\begin{aligned} \theta_\Delta &= \texttt{signed\_angle}(\texttt{unit\_vector}(\theta_{phys}), v_o) \\ g_c &= \texttt{sign}(\theta_\Delta) \times maxCurvatureRadius. \end{aligned} \tag{4.3.3.8}$$

Here, `signed_angle`(v_1, v_2) returns the positive angle between vectors v_1 and v_2 if the direction from v_1 to v_2 is counterclockwise. Otherwise, it returns the negative angle between v_1 and v_2. The `sign`(θ) function returns 1 or -1 depending on if θ is positive or negative. We set $maxCurvatureRadius$ to $7.5m$, which is a commonly-used curvature threshold value in the RDW literature [7, 83, 205].

We set the translation gain g_t according to the ratio between the average lengths of the physical and virtual slices, and we bound it by the perceptual thresholds for translation gains:

$$g_t = \texttt{clamp}(s_l^{phys}/s_l^{virt}, minTransGain, maxTransGain). \tag{4.3.3.9}$$

Here, the `clamp`(x, y, z) function returns x, but ensures that it is greater than or equal to the lower bound y and is less than or equal to the upper bound z. We set $minTransGain = 0.86$ and $maxTransGain = 1.26$ since these are commonly-accepted translation gain thresholds [187, 205].

4.4 Evaluation

We conducted four experiments to evaluate the performance of our algorithm. For each experiment, we compared our visibility-based algorithm against the alignment-based redirection controller (ARC) presented by Williams et al. [231], the artificial potential field (APF) controller by Thomas et al. [205], and the implementation of steer-to-center (S2C) by Hodgson et al. [83]. Our algorithm's reset function is the same as the one used by ARC [231]. Both APF and S2C use the modified reset-to-center algorithm introduced by Thomas et al. [205]. ARC is currently the best-performing reactive controller, while controllers based on potential-fields also perform fairly well. We compare against S2C since it is a very common benchmark to compare against in the RDW literature; however, we note that S2C is not expected to do well in any of our experiments due to the obstacles present in the PE. The first three experiments involve only static scenes with no dynamic obstacles. Our fourth experiment investigates controller performance in dynamic scenes.

It is known that the structure of the environment affects a redirection controller's performance [132], so it is important to consider the environments' layouts when assessing a controller's efficacy. There is currently no standard suite of test environments with which we can evaluate our algorithm, so we opted to test in environments used before to move towards a more standard set of environments. Furthermore, we wish to test our algorithm in cluttered environments with many obstacles and narrow passageways, which are challenging scenarios where avoiding obstacles is non-trivial. Thus, we evaluated our algorithm using two of the environment pairs from Williams et al. [231]. When choosing our physical environments, we wished to test environments that did not represent traditional physical tracked spaces which often have very few obstacles. Our goal is

to test the viability of redirection in unstructured, irregular environments that may be commonly encountered in the real world, such as an office building with cubicles or a living room with tables and couches.

The performance metric we use is the number of resets incurred over the entire duration of the walked path since it is a fairly standard performance metric in the RDW community. Another common metric is the average virtual distance walked between resets, but this metric is dependent on the number of resets, so it is slightly redundant to include both metrics in our evaluation. Before conducting our experiments, we developed three hypotheses:

H1 Our visibility-based steering algorithm will result in fewer collisions than the current state-of-the-art controllers in static scenes.

H2 Our visibility-based steering algorithm will result in fewer collisions than the current state-of-the-art controllers in dynamic scenes.

H3 Redirection controllers that use alignment will perform better in physical-virtual environment pairs that have more local similarity than they will in environment pairs that have less local similarity.

4.4.1 Environment Pairs

Each experiment had a different pair of physical and virtual environments. The environments are shown in Figure 5.4. Experiment 1 includes a $12m \times 12m$ physical and $17m \times 12m$ VE consisting of narrow corridors (see Table 4.1 for exact layout details). Experiment 2 has a $10m \times 10m$ PE with three rectangular obstacles and a $20m \times 20m$ VE with many convex and non-convex obstacles (see Table 4.2). Using these two environments makes it easier to draw

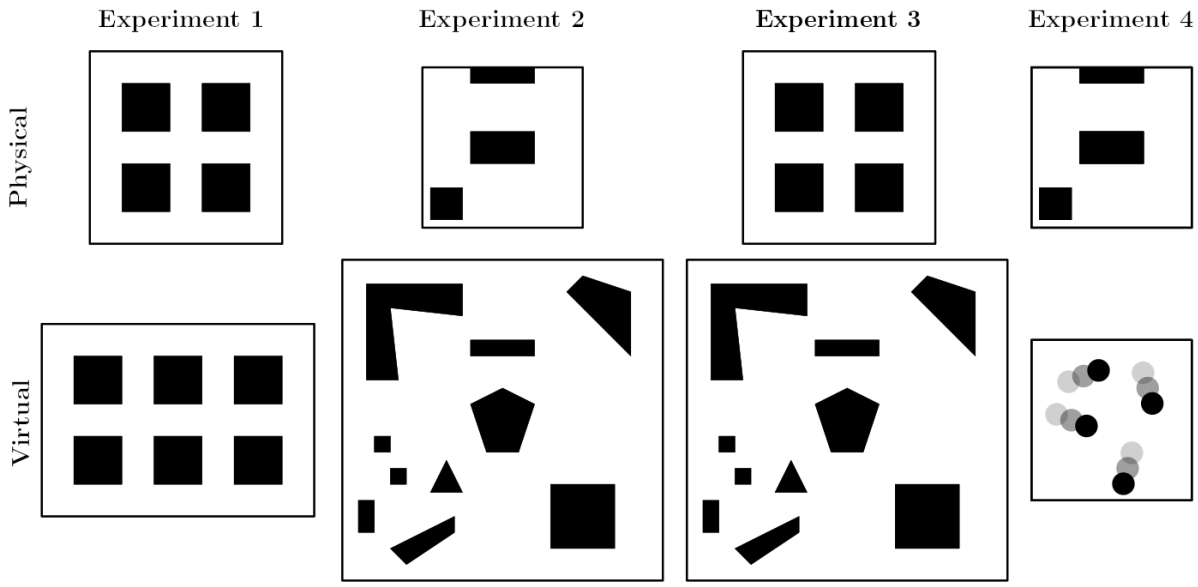


Figure 4.6: The layouts of the different environment pairs we tested in our experiments. The faded circles in the virtual environment for Experiment 4 indicate that the circles change position over time.

comparisons between our work and that of Williams et al. [231]. Experiment 3 uses the PE from Experiment 1 and the VE from Experiment 2. We noticed that the pairs of environments used in Environment B and C in [231] (Experiments 1 and 2 here) appear to have a high degree of local similarity. That is, the user's proximity to obstacles will be roughly similar between the physical and virtual environments. In Environment B, both environments feature only narrow corridors and angular turns. In Environment C, both physical and virtual environments have irregularity in the sizes and shapes of obstacles in both environments. Therefore, for our third experiment, we opted to evaluate the controllers on a mixture between the environments from Experiments 1 and 2 since this would lower the degree of local similarity between the physical and virtual environments and allow us to test H3.

Finally, in Experiment 4, we used the physical environment from Experiment 2 and an empty $10m \times 10m$ environment with four dynamic, circular obstacles as the virtual environment (see Table 4.3). We chose to use a VE with no static obstacles so that we could more easily study

Experiment 1 (physical)

Boundary	$(-6,-6),(6,-6),(6,6),(-6,6)$
Obstacle 1	$(-4,-4),(-1,-4),(-1,-1),(-4,-1)$
Obstacle 2	$(1,-4),(4,-4),(4,-1),(1,-1)$
Obstacle 3	$(1,1),(4,1),(4,4),(1,4)$
Obstacle 4	$(-4,1),(-1,1),(-1,4),(-4,4)$

Experiment 1 (virtual)

Boundary	$(-11,-6),(6,-6),(6,6),(-11,6)$
Obstacle 1	$(-4,-4),(-1,-4),(-1,-1),(-4,-1)$
Obstacle 2	$(1,-4),(4,-4),(4,-1),(1,-1)$
Obstacle 3	$(1,1),(4,1),(4,4),(1,4)$
Obstacle 4	$(-4,1),(-1,1),(-1,4),(-4,4)$
Obstacle 5	$(-9,1),(-6,1),(-6,4),(-9,4)$
Obstacle 6	$(-9,-4),(-6,-4),(-6,-1),(-9,-1)$

Table 4.1: Coordinates of vertices of boundaries and obstacles in both environments used in Experiment 1.

the influence of dynamic obstacles on the controller's performance. That is, in an environment with both static and dynamic obstacles, it may be difficult to determine to what degree either type of obstacle influences the controller's performance.

4.4.2 Simulated Environment

To evaluate the effectiveness of our algorithm, we conducted extensive experiments with a simulated user walking in virtual reality. Simulation has become a popular method of evaluation for redirection controllers since it allows researchers to quickly iterate on their algorithms and run large-scale experiments in a variety of environments [11, 29, 54, 117, 132, 192, 205, 206, 207, 231]. Our simulated user is represented as a circle with radius $0.5m$, and a reset is incurred whenever they came within $0.2m$ of any obstacle in the PE. The user walked with a speed of $1m/s$ and turned with a speed of $90°/s$. Our simulation timestep size was 0.05.

In our simulation, the model for generating the paths of the user is slightly different for static and dynamic scenes. For static scenes, we used the motion model that was introduced

Experiment 2 (physical)

Boundary	$(-5,-5),(5,-5),(5,5),(-5,5)$
Obstacle 1	$(-4.5,-4.5),(-2.5,-4.5),$ $(-2.5,-2.5),(-4.5,-2.5)$
Obstacle 2	$(-2,-1),(2,-1),(2,1),(-2,1)$
Obstacle 3	$(-2,4),(2,4),(2,5),(-2,5)$

Experiment 2 (virtual)

Boundary	$(10,-10),(10,10),(-10,10),(-10,-10)$
Obstacle 1	$(-4.5,-4.5),(-2.5,-4.5),(-3.5,-2.5)$
Obstacle 2	$(0,2),(2,1),(1,-2),(-1,-2),(-2,1)$
Obstacle 3	$(-2,4),(2,4),(2,5),(-2,5)$
Obstacle 4	$(-8.5,8.5),(-8.5,2.5),(-6.5,2.5),$ $(-7,7),(-2.5,6.5),(-2.5,8.5)$
Obstacle 5	$(-8,-1),(-8,-2),(-7,-2),(-7,-1)$
Obstacle 6	$(-7,-3),(-7,-4),(-6,-4),(-6,-3)$
Obstacle 7	$(-9,-5),(-9,-7),(-8,-7),(-8,-5)$
Obstacle 8	$(-6,-9),(-3,-7),(-3,-6),(-7,-8)$
Obstacle 9	$(3,-4),(3,-8),(7,-8),(7,-4)$
Obstacle 10	$(5,9),(4,8),(8,4),(8,8)$

Table 4.2: Coordinates of vertices of boundaries and obstacles in both environments used in Experiment 2.

by Azmandian et al. [7] and has been used by others [207, 231]. For dynamic scenes, we used ORCA [211] to generate trajectories for the user and the dynamic obstacles since ORCA generates smooth, collision-free paths for multiple objects in the same environment.

4.4.3 Experiment Design

For each of the static environment experiments, we generated 100 paths using the path model developed by Azmandian et al. [7]. The average path length in these experiments was roughly $350m$. We ran our simulation on these 100 paths once with each of the redirection controllers we evaluated (our visibility-based controller, ARC [231], APF [205], and S2C [83]). For each path, the user starts in a random location in the physical and virtual environments. The user also has a random heading in both environments. It is important to clarify that these random

Experiment 4 (physical)	
Boundary	$(-5,-5),(5,-5),(5,5),(-5,5)$
Obstacle 1	$(-4.5,-4.5),(-2.5,-4.5),$ $(-2.5,-2.5),(-4.5,-2.5)$
Obstacle 2	$(-2,-1),(2,-1),(2,1),(-2,1)$
Obstacle 3	$(-2,4),(2,4),(2,5),(-2,5)$

Experiment 4 (virtual)	
Boundary	$(-11,-6),(6,-6),(6,6),(-11,6)$

Table 4.3: Coordinates of vertices of boundaries and obstacles in both environments used in Experiment 4.

starting configurations were different between the 100 paths but were the same each time we simulated a particular path.

For the dynamic scene, we generated 100 collision-free paths for the user and the four dynamic obstacles in the VE using the ORCA [211]. These paths had an average length of $136m$. These paths were shorter than those used in static environments because we generated the dynamic paths to take roughly the same amount of timesteps to complete as in the static experiments, *before* considering time taken for resets. As we did in the static scenes, all redirection controllers were evaluated on the same 100 paths that we generated with ORCA.

4.5 Results

We compared the number of resets across all 100 paths for the four algorithms that we tested in our experiments. Some of our data violated assumptions of homoscedasticity or normality. To account for these violated assumptions, we compared the controllers' performance with a robust one-way repeated measures ANOVA with 20% trimmed means. We used the WRS2 package in R to conduct our analyses [128]. For all of the results presented in this section, we include the test statistic (F) and the significance level (p-value). We also include the results of the post-

Number of Resets												
	Experiment 1			Experiment 2			Experiment 3			Experiment 4		
Redirection Controller	$\hat{\psi}$	CI	p	$\hat{\psi}$	CI	p	$\hat{\psi}$	CI	p	$\hat{\psi}$	CI	p
Vis. Poly. vs ARC [231]	-32.7	[28.9, 36.5]	$< .0001$	-5.68	[-2.73, -8.64]	$< .0001$	-24.8	[-20.6, -29.0]	$< .0001$	-5.68	[-3.70, -7.67]	$< .0001$
Vis. Poly. vs APF [205]	-76.4	[-101, -51.7]	$< .0001$	-105	[-117, -92.1]	$< .0001$	-49.6	[-71.1, -28.0]	$< .0001$	-64.3	[-82.7, -45.9]	$< .0001$
Vis. Poly. vs S2C [83]	-114	[-143, -85.1]	$< .0001$	-342	[-362, -321]	$< .0001$	-56.9	[-71.7, -42.1]	$< .0001$	-183	[-203, -162]	$< .0001$
ARC [231] vs APF [205]	-42.3	[-66.6, -18.1]	$< .0001$	-99.0	[-112, -86.4]	$< .0001$	-25.2	[-46.8, -3.56]	$< .01$	-59.1	[-77.4, -40.7]	$< .0001$
ARC [231] vs S2C [83]	-82.4	[-114, -51.3]	$< .0001$	-335	[-356, -313]	$< .0001$	-32.0	[-47.6, -16.3]	$< .0001$	-177	[-197, -157]	$< .0001$
APF [205] vs S2C [83]	-35.2	[-71.3, 0.882]	$< .01$	-233	[-254, -212]	$< .0001$	-8.02	[-31.3, 15.2]	$= .350$	-112	[-133, -91.5]	$< .0001$

Table 4.4: The results of post-hoc pairwise comparisons of average number of resets for the redirection algorithms tested in our experiments. The post-hoc tests are computed using linear contrasts. The $\hat{\psi}$ value is the average difference in means between the first algorithm and the second algorithm listed in the "Redirection Conotrller" column. A negative $\hat{\psi}$ value indicates that the first algorithm has a lower average number of resets across all 100 paths. The CI column presents the lower and upper bounds of the confidence interval, while the p column presents the significance level of the difference between the algorithms. The $\hat{\psi}$ and CI values are rounded to three significant figures.

hoc comparisons in Table 4.4 and Table 4.5, which are computed using linear contrasts. For the post-hoc tests, we report the difference between the means ($\hat{\psi}$), the upper and lower confidence intervals ([CI lower, CI upper]), and the significance level (p-value). Note that we use confidence intervals as a measure of effect size [124]. The main focus of this work is to study the efficacy of our visibility-based algorithm, so we do not include the results of the post-hoc tests for any comparisons that do not include our visibility-based steering controller in the main text (e.g. ARC compared to APF).

Resets per Meter Walked												
	Experiment 1			Experiment 2			Experiment 3			Experiment 4		
Redirection Controller	$\hat{\psi}$	CI	p	$\hat{\psi}$	CI	p	$\hat{\psi}$	CI	p	$\hat{\psi}$	CI	p
Vis. Poly. vs ARC [231]	-.0929	[-.0811, -.105]	$< .0001$	-.0157	[-.00746, -.0239]	$< .0001$	-.0688	[-.0570, -.0806]	$< .0001$	-.0428	[-.0278, -.0577]	$< .0001$
Vis. Poly. vs APF [205]	-.220	[-.291, -.149]	$< .0001$	-.299	[-.324, -.255]	$< .0001$	-.137	[-.198, -.0767]	$< .0001$	-.471	[-.598, -.345]	$< .0001$
Vis. Poly. vs S2C [83]	-.327	[-.411, -.242]	$< .0001$	-.940	[-.995, -.885]	$< .0001$	-.158	[-.200, -.116]	$< .0001$	-1.34	[-1.47, -1.21]	$< .0001$
ARC [231] vs APF [205]	-.123	[-.195, -.0515]	$< .0001$	-.274	[-.309, -.239]	$< .0001$	-.0697	[-.131, -.00858]	$< .01$	-.431	[-.556, -.306]	$< .0001$
ARC [231] vs S2C [83]	-.237	[-.325, -.149]	$< .0001$	-.920	[-.975, -.864]	$< .0001$	-.0886	[-.132, -.0456]	$< .0001$	-1.30	[-1.43, -1.17]	$< .0001$
APF [205] vs S2C [83]	-.101	[-.207, .00495]	$< .05$	-.642	[-.700, -.587]	$< .0001$	-.0220	[-.0851, .0412]	$= .346$	-.829	[-.975, -.683]	$< .0001$

Table 4.5: The results of post-hoc pairwise comparisons of average distanced walked between resets for the redirection algorithms tested in our experiments. The post-hoc tests are computed using linear contrasts. The $\hat{\psi}$ value is the average difference in means between the first algorithm and the second algorithm listed in the "Redirection Conotrller" column. A negative $\hat{\psi}$ value indicates that the first algorithm has a lower average number of resets across all 100 paths. The CI column presents the lower and upper bounds of the confidence interval, while the p column presents the significance level of the difference between the algorithms. The $\hat{\psi}$ and CI values are rounded to three significant figures.

4.5.1 Experiment 1

The robust ANOVA indicated a significant difference in the number of resets incurred when exploring with the different redirection controllers $F(1.88, 111.03) = 46.92,\ p < .0001$. We see in Figure 4.7 that our visibility-based algorithm achieves a median of 121 resets, which is 32 lower than the median of 153 achieved by ARC. The median number of resets achieved by APF and S2C is slightly higher (188.5 and 217.5), and these two controllers have a much larger variance in the number of resets across all 100 paths.

4.5.2 Experiment 2

Our results showed a significant difference in performance between the RDW controllers $F(1.66, 97.65) = 1266.94,\ p < .0001$. Figure 4.8 shows us that the median number of resets incurred by our algorithm is 87, that the median for ARC is 95, and that APF and S2C achieve upwards of 150 median resets.

4.5.3 Experiment 3

There was a significant difference between the four steering algorithms in terms of the number of resets incurred $F(1.84, 108.75) = 29.90,\ p < .0001$. In Figure 4.9 we see that this significant difference favors our new algorithm. Similar to the pattern in first experiment, our visibility-based redirection controller achieves a median of 147.15 resets, which is about 25 fewer than that the 173 achieved by ARC, and APF and S2C have higher median resets (180 and 197) with a larger variance, too.

4.5.4 Experiment 4

In the dynamic scene, we found that the number of resets incurred was significantly different between the four controllers $F(2.04, 120.24) = 319.03,\ p < .0001$. Once again, the significant difference favors the new visibility-based algorithm we presented in this work (see Table 3.4). Our algorithm had a median of 31 resets, while ARC had a median of 37 resets, APF had a median of 90 resets, and S2C had a median of 213.5 resets.

4.6 Discussion

4.6.1 Static Scenes

Our first three experiments were conducted in static scenes. The results showed that our novel algorithm based on visibility polygons resulted in significantly fewer resets with obstacles in the environments we tested. Overall, the results in the static scenes supported our first hypothesis (H1) that our visibility-based controller will outperform existing redirection controllers in static scenes. In the first experiment, the physical and virtual environments had a considerable similarity since both environments featured regularly spaced narrow corridors, though the VE was larger than the physical one. The simulated user incurred more resets in this PE than in the one used for Experiment 2. This is likely because the narrow corridors make it difficult for the user to consistently walk without resets.

Experiment 3 used the PE from Experiment 1 and the virtual environment from Experiment 2. We chose this combination in order to study the steering algorithms' performance in environment pairs that have little similarity on a local scale. The PE is very regular and has only 90° turns. However, the VE contains many differently-shaped obstacles, creating irregular turns and regions of differing amounts of free space. We found that all redirection controllers except APF performed

worse in this experiment than they did in the others, which we believe is due to the more significant mismatch between the physical and virtual environments. It suggests that controllers based on alignment will experience more difficulty avoiding resets in environment pairs with low local similarity than they will in pairs with high local similarity. Note that APF, which does not steer the user with alignment, was able to perform *better* in Experiment 3 than in Experiment 1. The result of Experiment 3 supports our third hypothesis (H3) that controllers that steer based on alignment will perform better in environment pairs with high local similarity than in environment pairs with low local similarity.

4.6.2 Dynamic Scenes

In addition to testing our controller in static scenes, we also evaluated its performance in a scene with four dynamic virtual obstacles. We found that our steering algorithm based on visibility polygons performed significantly better than ARC, APF, and S2C. This result supports our second hypothesis (H2) that the additional information captured by the visibility polygon will lead to fewer resets in dynamic scenes. One interesting result from the fourth experiment is that overall, the number of resets was much lower than in any of the other experiments. This is initially surprising since dynamic environments intuitively seem to be more challenging to navigate. However, considering that the paths for this experiment were generated using ORCA [211], this is less surprising. The paths in Experiments 1-3 consist only of straight-line segments, while the paths in Experiment 4 include curved path segments. It may be the case that the curved segments are more amenable to redirection, which may be the cause of the lower number of resets seen in Experiment 4. Another possible explanation for the lower number of resets is the fact that the user's speed varied in Experiment 4 but was constant in Experiments 1-3. Finally, the paths

in Experiment 4 were shorter than those in Experiments 1-3 since they were designed to take the same amount of time as the paths in the static scenes, and the user sometimes stopped walking in order to avoid collisions with the moving obstacles.

4.6.3 Other Considerations

In our algorithm, we used triangle area as our metric for measuring the similarity between slices of the physical and virtual visibility polygons (subsubsection 4.3.3.3). Area is a relatively crude similarity metric, since two polygons can have very different shapes but still have the same area. Metrics that can more accurately capture how similar two triangles are may lead to improvements in avoiding resets, since the improved metric will guide users towards free space regions that are more similar. Shape similarity is a well-studied problem in the geometry and vision communities, so we believe that it is very likely that there exists a better triangle similarity metric. There is likely room for more sophisticated similarity metrics that go beyond triangular slices. We compared visibility polygons using the triangular slices that divide the polygons since it was a very natural way to segment the polygons, but it is possible that there exist better decompositions for comparing visibility polygons.

4.7 Conclusion, Limitations, and Future Work

In this work, we presented a novel formulation of the redirected walking problem and developed a visibility-based redirection controller that takes advantage of this formulation to yield fewer resets during walking. Our formal description of the redirection problem frames it in terms of optimizing the redirection gains such that the user follows a collision-free path in the physical free space while simultaneously traveling on a collision-free path in the virtual free space. We hope that this new formulation will allow researchers to further improve redirection

controllers by leveraging motion planning techniques that have been well-studied by the robotics community.

The novel redirection controller that we also presented in this work is based on the key insight that the visibility polygon provides a reliable representation of the user's free space at their location. By dividing the physical and virtual visibility polygons into "slices" and steering the user towards a physical slice that is similar to the current virtual slice, we were able to avoid significantly more resets than the state-of-the-art controllers were able to. As evidenced by our simulation-based results, our algorithm proved to be more effective in both static and dynamic scenes where the physical and virtual environments had locally dissimilar layouts.

Although our results were positive, it is important to discuss the limitations of this work, too. In our experiments, we only evaluated the algorithms in simulated environments. Simulation can be effective for quickly getting an understanding of a RDW algorithm's strengths and weaknesses, but full user studies should be conducted to gain a more complete evaluation of a controller's efficacy and to highlight shortcomings that may not arise in simulations. Another limitation of this work is that we only tested scenarios involving one user in the PE and VE. Many use-cases for VR involve multiple users, either in the same PE or different PEs. Thus, it is important to continue developing RDW controllers that can improve the locomotion experience for multiple users [8, 11, 54]. Finally, it is important to consider the challenges involved in extending our algorithm to commodity VR hardware and uncontrolled, irregular physical and virtual environments. Our algorithm requires knowledge of the locations of obstacles around the user in the PE and VE. Obtaining this data for the VE is generally not problematic, since the virtual environment data is already being used for rendering purposes. However, collecting layout data of the PE is considerably more difficult, since this involves object detection and tracking in real time. This

problem can be sidestepped by providing the RDW controller with a map of the PE if it is known beforehand, but this cannot be guaranteed in a commodity VR setting where the application will be used in a variety of PEs. In addition to collecting environment data, it is important to consider the run-time of the algorithm. Visibility computations are very well-studied problems in computer graphics and robotics, so a considerable amount of effort has been invested into developing efficient and robust algorithms for computing visibility. Nevertheless, since many other computations need to be done on each timestep in VR applications, computing visibility polygons in real-time may be difficult if the environment contains a large number of obstacles.

Future work should investigate different ways to use the visibility polygon for more sophisticated steering, such as more accurate measures of shape similarity. Our results showed that for some environments, the performance improvement afforded by visibility polygons was not large (though it was statistically significant). It may be the case that our algorithm does not use visibility polygons to their full potential. There is not a lot of research studying the relationship between the shape of the virtual path and a controller's performance, so the impact of the path models on a controller's performance is an open question that warrants more research. Another interesting area for future work is to extend our visibility-based controller to real-world scenarios where the exact geometry of the environments is not known. Our experiments were conducted using reliable simulations to show that our algorithm is effective, but calculating visibility polygons in real PEs will likely introduce new challenges. Finally, extensive in-person user studies should be conducted now that we know that our visibility-based controller can yield significant benefits.

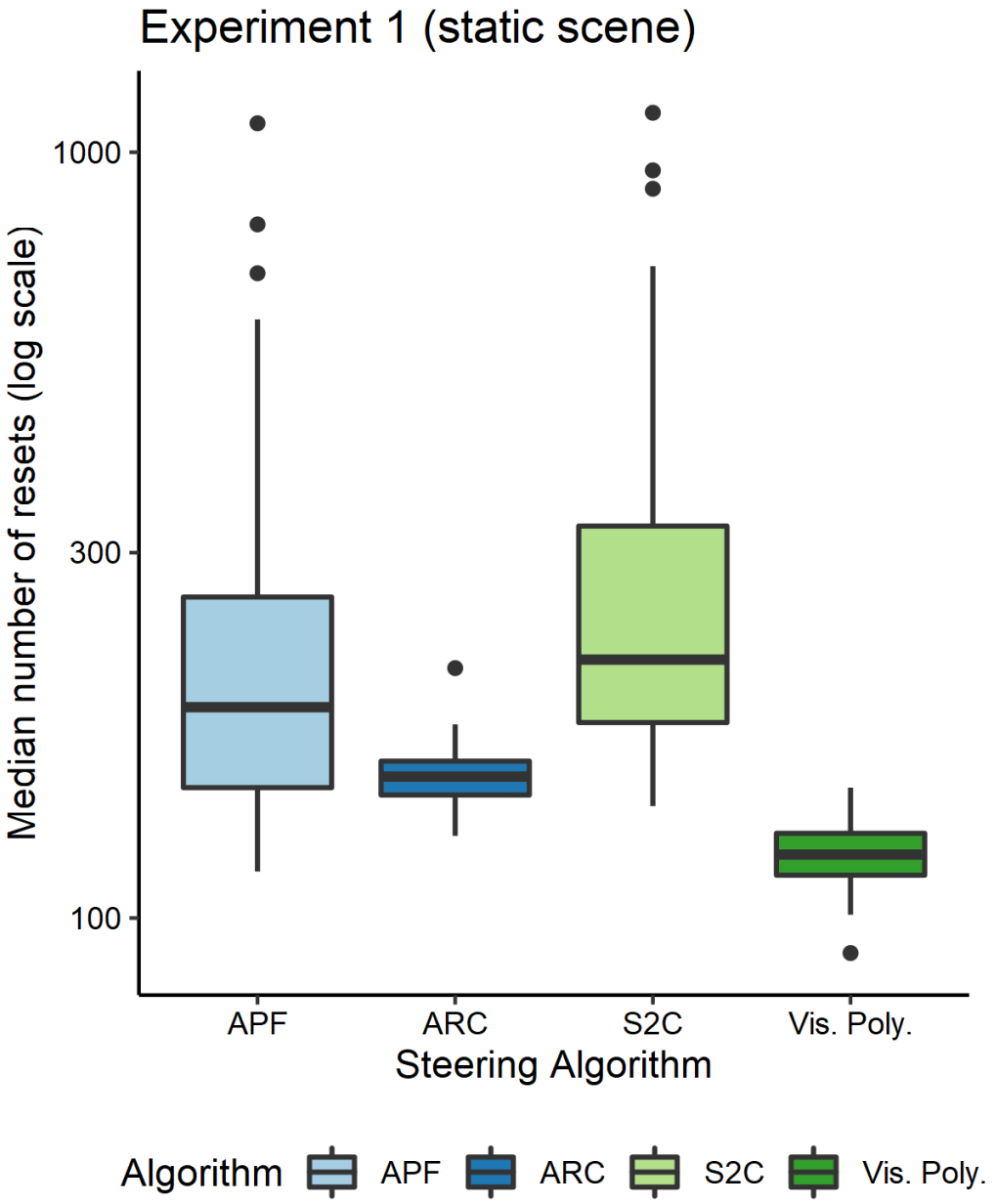


Figure 4.7: Boxplot of the number of resets for each algorithm, across all 100 paths in Experiment 1. Our visibility-based algorithm significantly outperformed each of the other redirection controllers.

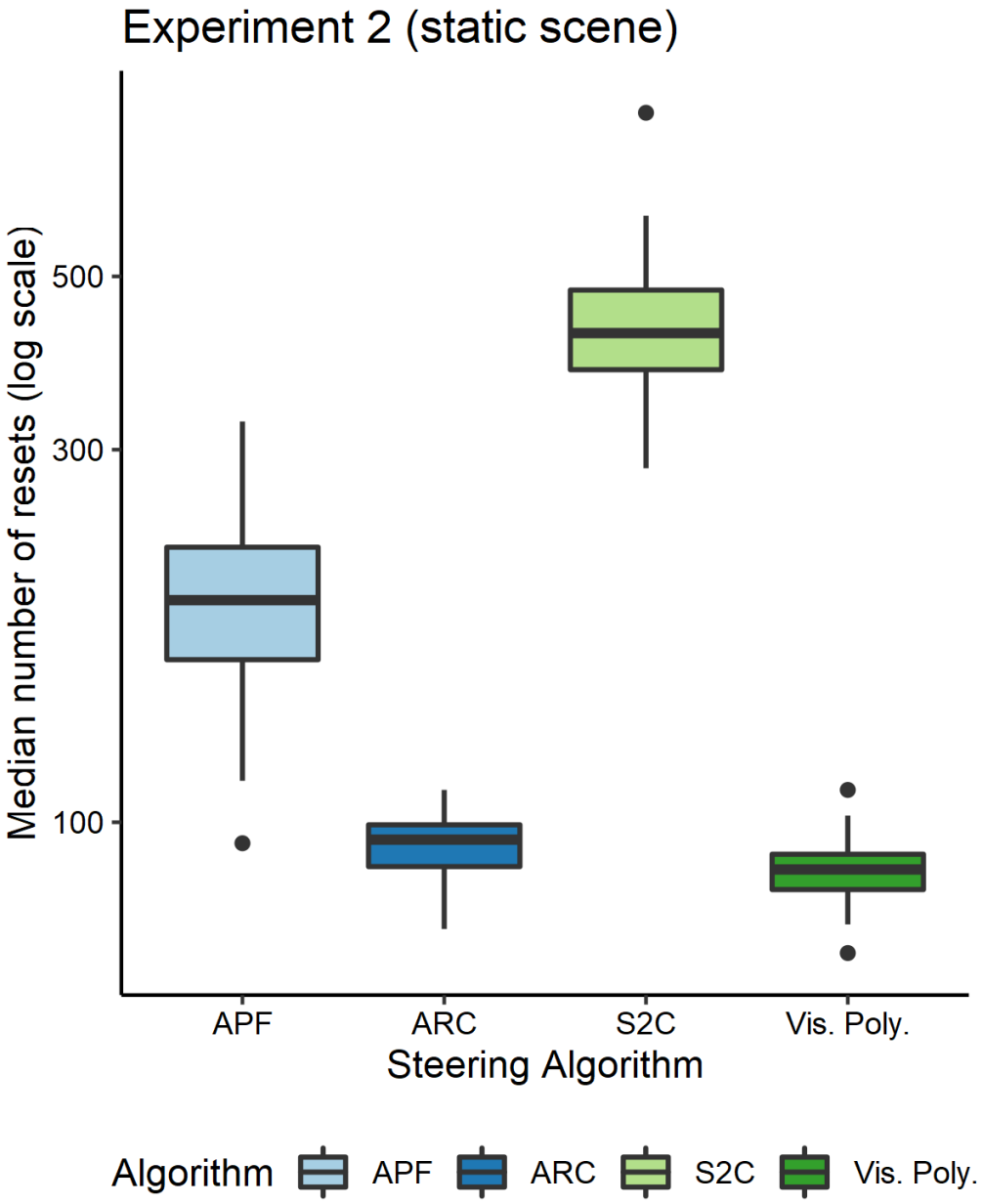


Figure 4.8: Boxplot of the number of resets for each algorithm, across all 100 paths in Experiment 2. The difference in the number of resets incurred is much larger for APF and S2C, which do not take advantage of alignment. ARC and our visibility-based controller (Vis. Poly.) have more similar performance levels, but our algorithm still produced significantly fewer resets.

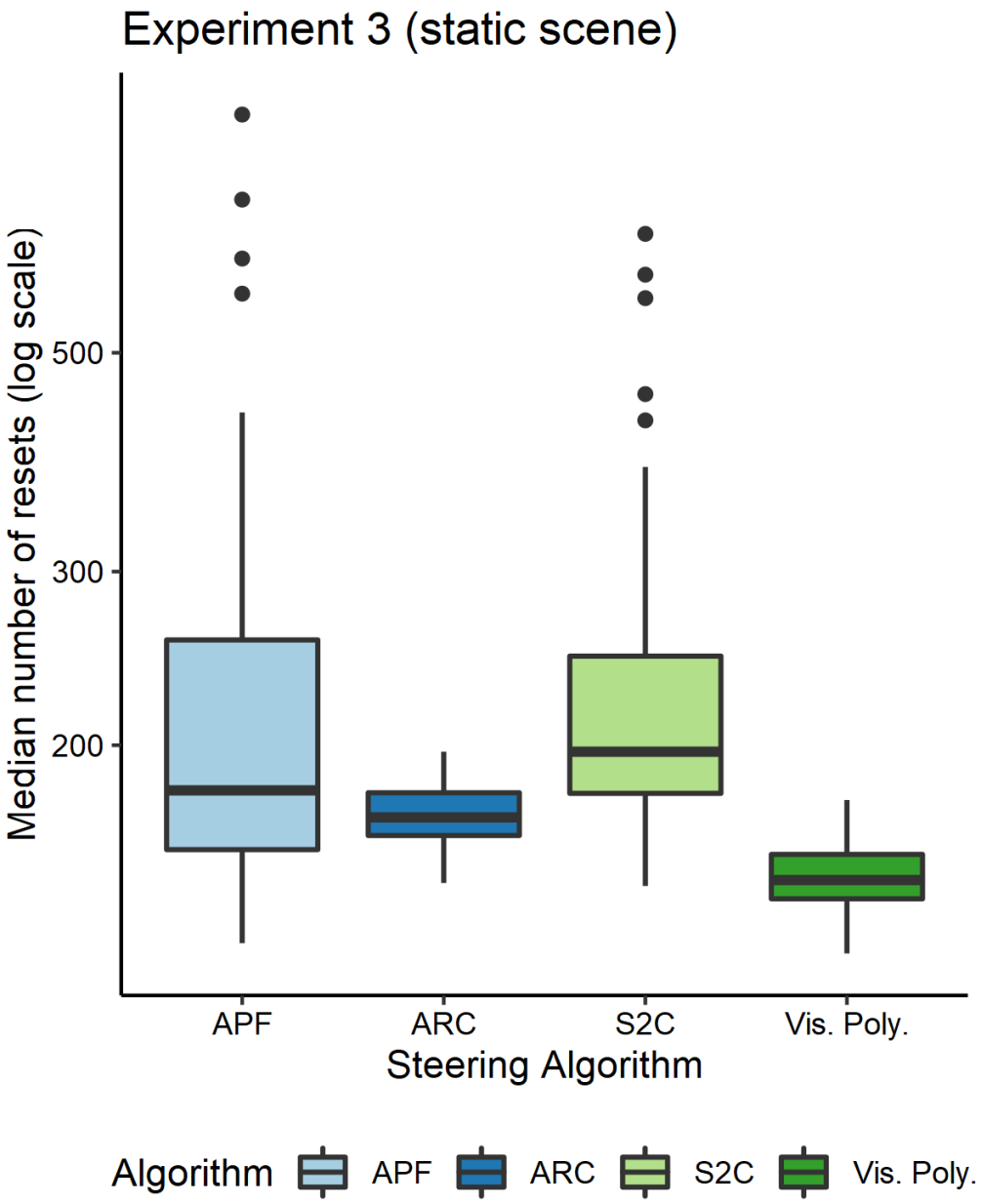


Figure 4.9: Boxplot of the number of resets for each algorithm, across all 100 paths in Experiment 3. Our controller based on visibility polygons performed significantly better than all other controllers.

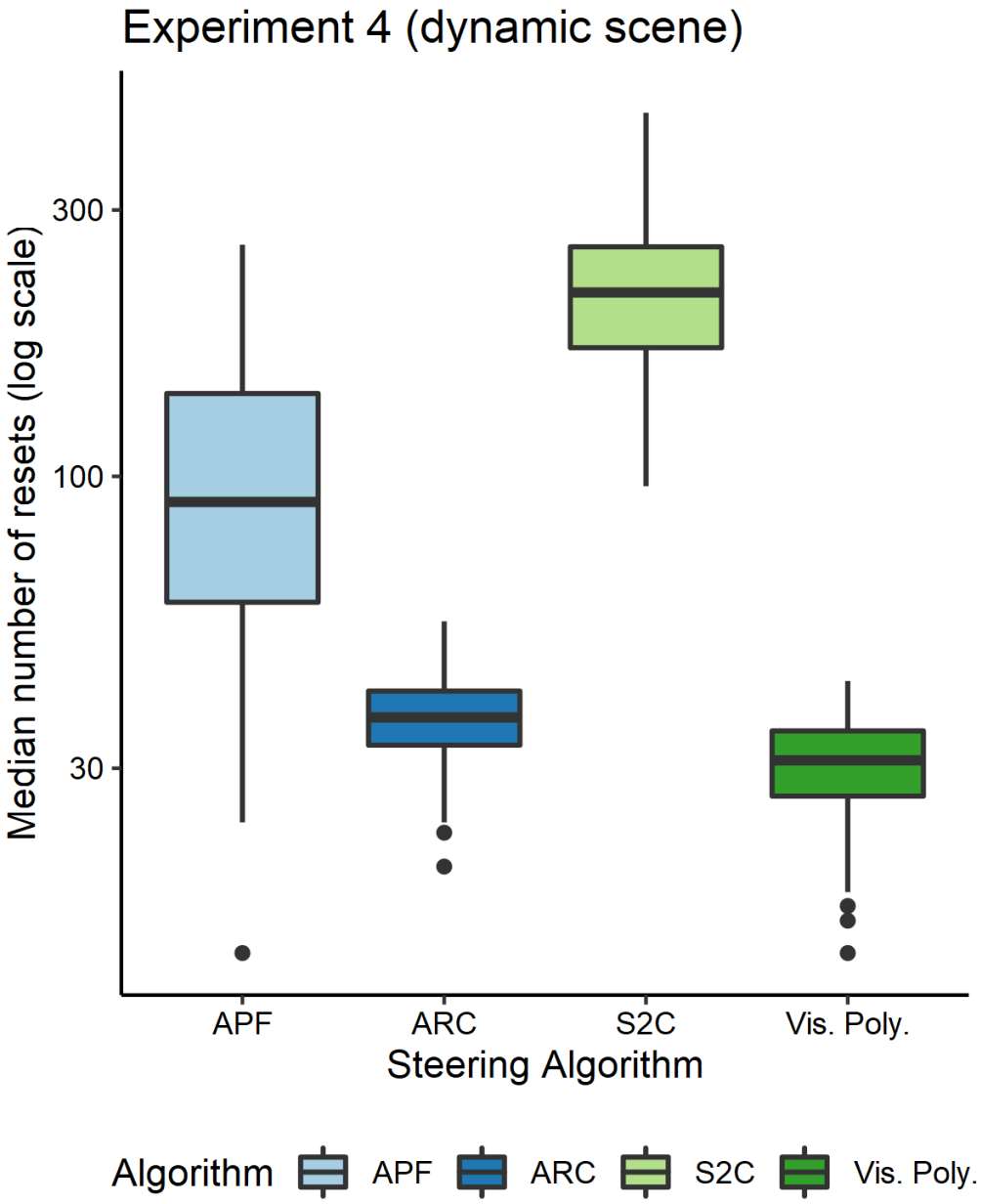


Figure 4.10: Boxplot of the number of resets for each algorithm, across all 100 paths in Experiment 4. In the dynamic scene we tested, we once again found that our visibility-based algorithm was significantly better than the other controllers at avoiding resets with physical obstacles.

Chapter 5: Distractor-Based Redirection

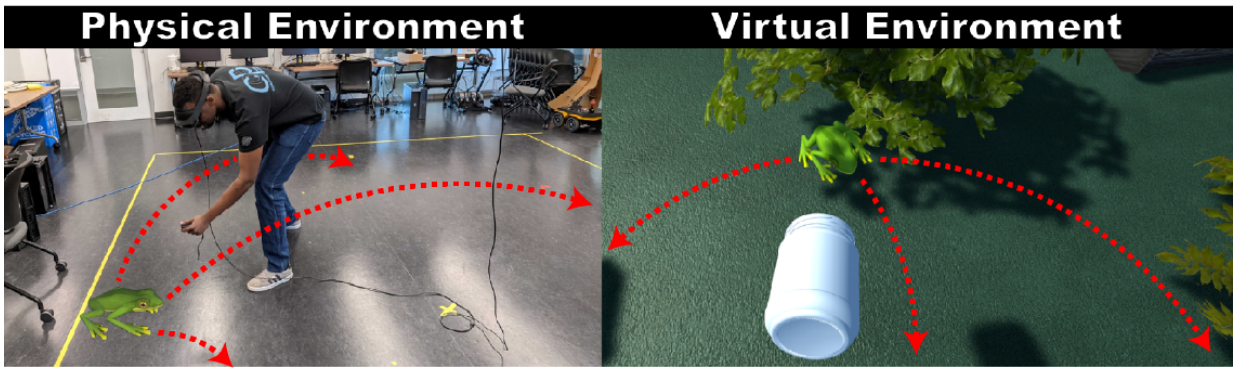


Figure 5.1: A visualization of our distractor-driven locomotion interface designed to enable exploration of large virtual environments using natural walking. In the virtual environment, the user approaches a distractor (a frog) in an attempt to collect it using a virtual, hand-tracked jar. Our algorithm detects this interaction and updates the behavior of the distractor in order to guide the user away from the nearby boundary of the physical space (yellow tape in the physical environment). In particular, our algorithm causes the frog to jump away to one of many candidate positions (dashed red arrows) in the virtual environment that corresponds to safe positions in the physical environment. The use of such distractors causes the user to alter their *virtual* trajectory such that it is more compatible with their physical surroundings, which allows the user to explore virtual worlds with with longer collisions-free trajectories even when they are located in a small physical environment.

In this chapter, we present a method for modifying the behavior of virtual content that the user interacts with as part of their virtual experience such that the user alters their virtual trajectory to yield fewer collisions with unseen obstacles in their physical surroundings. To achieve this, our method uses two main components: a safe zone computation module and a distractor behavior computation module. When an interaction with a virtual object (a *distractor*) is detected, our system first computes a location in the physical environment (PE) that the user can be safely guided towards and then modifies the behavior of the distractor such that it guides

the user towards the chosen safe location in the PE. To make our method effective in small PEs ($\sim 25\text{m}^2$), we take advantage of *persistent distractors*, elements of the VE that the user is likely to continually interact with over the course of their experience, to encourage the user to walk on collision-free trajectories for the duration of the virtual experience. To evaluate the viability of our method, we conducted three experiments that studied how often users had to be interrupted to avoid physical collisions with and without our method, and studied how different tunable parameters of our method affect the distractors' performance. Results showed that, compared to a traditional redirected walking method for natural walking in VR, our method allowed participants to travel uninterrupted for longer distances (7.864m vs 6.269m, 25% increase). Furthermore, qualitative results showed that participants preferred our distractor-driven interface and that many participants did not realize that their trajectory was being influenced by the distractors' behavior.

5.1 Introduction

The ability to freely explore virtual environments (VEs) in virtual reality (VR) is important for improving a person's understanding of their virtual surroundings [37, 52, 188]. Furthermore, exploration of VEs using natural locomotion provides benefits to the user's sense of presence in the environment [209] and to their performance on tasks in the VE [84, 162]. However, achieving natural walking in VR is difficult because the user's desired path through the VE might correspond to a path in the physical environment (PE) that yields a collision with an obstacle or exits the tracked space boundaries. Many different locomotion interfaces have been developed to overcome this problem (e.g., route-planning navigation [22], teleportation [23], omnidirectional treadmills [110], walking-in-place metaphors [120]), and each comes with its own advantages and disadvantages [52]. In this work, we focus on locomotion interfaces that enable *natural*

walking as a means of virtual exploration due to its benefits to spatial awareness, low learning curve, and low cost of entry [209].

Natural exploration of VEs is especially challenging since the PE is usually significantly smaller than the VE. The greater this size difference, the more likely it is that a valid path that the user wants to travel on in the VE will correspond to an invalid physical path that yields a collision with a physical obstacle [233]. The two main approaches that enable natural walking and mitigate this simultaneous navigation problem are manipulated virtual architecture [56, 57, 194, 196, 197, 215] and redirected walking (RDW) [140, 157, 158]. With manipulated virtual architecture, the layout of the VE is modified [57, 194, 215] or the virtual geometry is warped [56, 197] to fit within the user's physical space. A downside of this approach is that it cannot be applied to applications where the precise layout of the VE is a hard constraint that cannot be manipulated (e.g., a virtual real estate tour). With RDW, subtle rotations and translations are injected into the user's virtual movements, which causes the user to subconsciously walk on a physical path that has a different shape than its virtual counterpart. One downside of RDW is that the benefit it provides scales with the size of the PE—RDW requires a large amount of space (and time) in order to significantly alter the user's physical trajectory such that they can be reliably steered away from obstacles [7, 132]. Considering that most people's living spaces, where they are likely to want to use VR, are quite small ($20-28\text{m}^2$ [61]), the utility of RDW is limited in the kinds of situations where a VR locomotion interface is most needed.

Main Results: In this work, we focus on enabling natural walking to explore large VEs when the user is located in a small PE, which we consider to have an area of approximately $20-28\text{m}^2$ [61]. We develop a new natural walking locomotion interface that leverages *persistent distractors*—elements of the virtual environment that the user continually interacts with over the

course of their virtual experience. Our interface consists of two main components, a "safe zone computation" module and a "distractor behavior modification" module. When an interaction between the user and a persistent distractor is initiated, our method computes the regions of the physical space that the user can be guided towards without colliding with any unseen physical obstacles, which we label as *"safe zones."* After a safe zone has been chosen, the next step is to compute and execute a behavior for the persistent distractor that will influence the user to alter their virtual trajectory such that they are guided towards the chosen safe zone. In this manner, we are able to guide the user along virtual trajectories that decrease the chance of a physical collision. Our method is general enough to be applicable to any physical and virtual environment as long as the physical safe zones can be computed and an appropriate distractor behavior can be executed to guide the user to a safe zone. To evaluate our interface, we conducted three experiments and compared against a common benchmark for generalized redirected walking (steer-to-center [84]) paired with randomized distractor behavior. Quantitative results of our study indicated that when steered using our distractor-based algorithm, users were able to walk on average 1.6m further ($\sim 20.3\%$ increase) before being forced to stop and reorient due to an imminent collision in the PE. Qualitative results indicated that participants found the distractor-driven algorithm to be engaging and that they were unaware that they were being explicitly guided around the PE. In summary, our main contributions are:

- A new VR locomotion algorithm that is based on continually integrating the virtual content into the path-planning system in the form of *distractors*. Unlike RDW, our algorithm causes users to overtly change their *virtual* trajectory such that it is less likely to yield a physical collision. We achieve this by computing candidate safe regions of the physical space and

modifying the behavior of persistent distractors that the user engages with to guide them along collision-free trajectories. By overtly changing the virtual trajectory, it is easier to avoid physical obstacles in small physical environments where there is insufficient space for RDW to be effective.

- Three user studies that compare the effectiveness of our algorithm against a classic redirection algorithm (steer-to-center) with unoptimized distractor behavior. Results of our studies show that distractor-guided navigation allows users to travel an average of 1.6m further ($\sim 25.8\%$ increase) before incurring a collision with an obstacle, and that it can yield more immersive and comfortable experiences for users.

5.2 Background & Related Work

5.2.1 Natural Walking in Virtual Reality

Many different locomotion interfaces have been developed to enable users to explore virtual environments (VEs) that are much larger than their surrounding physical environment (PE) [52, 188]. Common locomotion interfaces for VR include teleportation [23], walk-in-place [120], flying [209], and step-driven locomotion. Each interface comes with different advantages and disadvantages in terms of their learning curve, the level of presence they afford, and their efficiency. In this work, we focus on locomotion interfaces that allow users to explore VEs using natural, everyday walking since prior work has shown that these walking-based interfaces have a low learning curve and tend to improve users' sense of presence, memory, and task performance in VR [84, 162, 209].

Redirected walking (RDW) is a popular walking-based interface that works by imperceptibly rotating or translating the VE around the virtual camera while the user explores [158]. By

injecting these rotations and translations slowly enough, users do not consciously perceive them but still adjust their physical trajectory in order to counteract the VE movements and remain on their intended path in the VE. Researchers have studied how easily users can perceive the VE motions under different conditions [92, 139, 187, 230] and have developed a myriad of algorithms that apply RDW to steer users away from physical obstacles in both single-user [9, 42, 113, 158, 192, 198, 205, 231, 232, 243] and multi-user scenarios [11, 54]. Recent work by Azmandian et al. [9] showed how to design a "meta-strategy" that switches between different RDW algorithms at runtime to reduce the number of resets incurred. One potential downside of RDW is that it creates a mismatch between the user's physical and virtual trajectories, which may make it difficult for users to interact with physical objects that are aligned with their virtual counterparts. To mitigate this downside, researchers have investigated the viability of using redirection for collision avoidance *and* physical-virtual alignment [206, 207, 231, 232].

Aside from slowly rotating and manipulating the VE, researchers have also directly edited the geometry of the VE to alter the user's physical path. Sun et al. [197] and Dong et al. [55, 56] compute mappings from the VE to the PE to ensure that all paths in the VE are navigable from within the given PE. Other geometry-based solutions include altering the structure of the VE while the user is not looking at particular regions [194, 214, 215], creating impossible environments with overlapping rooms [39], and procedurally generating the virtual environment according to the user's physical surroundings [35, 57, 166, 173, 182, 238]. Recent reviews of RDW were published by Nilsson et al. [140] and Li et al. [122].

5.2.2 Distractors in Virtual Reality

It is often the case that the virtual environment contains many dynamic elements and objects of interest that capture and occupy the user's attention (e.g. narrative elements of a video game). In the context of RDW, these elements are known as *distractors* [146]. Diegetic distractors are those attention-grabbing elements that make sense to exist within the surrounding context of the virtual experience (e.g. cars in a virtual city), while nondiegetic distractors are not plausible within the context of the experience (e.g. a floating arrow that guides users).

Peck et al. studied the effects of distractors when participants were about to walk out of the boundaries of the VR tracking space and found that they significantly improved the reorientation experience for users [147, 148, 149]. Similarly, Rewkowski et al. [159] showed that audio-based distractors can be an effective method for reorienting users in VR. Chen et al. [32] and Sra et al. [183] demonstrated how distractors can be integrated into the virtual content in a diegetic manner to make the reorientation events less jarring. Cools et al. [44] studied how different levels of interactivity with distractors can yield different benefits for the locomotion experience. Finally, Williams et al. showed that the user's sensitivity to RDW changes in the presence of distractors [230] and showed that haptics can be used to improve the effectiveness of distractors for redirected walking [234].

In this work, we build upon the distractors literature by incorporating distractors into the VR locomotion interface in a way that can be applied to any virtual experience, as long as certain constraints are fulfilled. Our work differs from the existing literature on distractors in that we focus on how to use distractors *throughout* the duration of the virtual experience, as opposed to invoking them only once the user reaches a physical obstacle or boundary. That is, we introduce

the notion of *persistent distractors* that exist in the VE as a core element of the experience and can be interacted with even when the user does not need to be reoriented away from obstacles. The work that is most similar to our current work is the haptic distractor by Williams et al. [234] since the distractor (a virtual dog on a leash) was always present in their work, but it only provided distracting stimuli if the user approached the boundaries of the tracked space.

5.3 Persistent Distractor-driven Locomotion

At a high level, our method operates by first detecting when the user interacts with a virtual object that can be used as a distractor, computing the safe region of the PE that the user should be guided towards to minimize the chance of collision, and then modifying the distractor's behavior to influence the user to change their *virtual* trajectory to move towards the safe physical region. In this section, we provide implementation details on our method. For each component of the system, we first explain the implementation in generalized terms that can be applied to any virtual experience and physical environment, then we provide details on how the component was implemented into our VR application that was used in our user study. An overview of our framework is shown in Figure 5.2.

To evaluate our method (Section 5.4), we implemented a VR game in which users were tasked with catching virtual frogs (Figure 5.1). In our application, the user was placed into a $20\text{m} \times 20\text{m}$ VE with trees, bushes, and stones scattered about the environment. The VE was also populated with frogs that jumped from one bush to another; these frogs served as the primary distractors whose behavior we modified to influence the user's trajectory. The user held a virtual jar in one hand (which was tracked using a VR controller) and could catch frogs by moving the jar close enough to a frog. To improve the realism of the experience, the bushes in the VE also

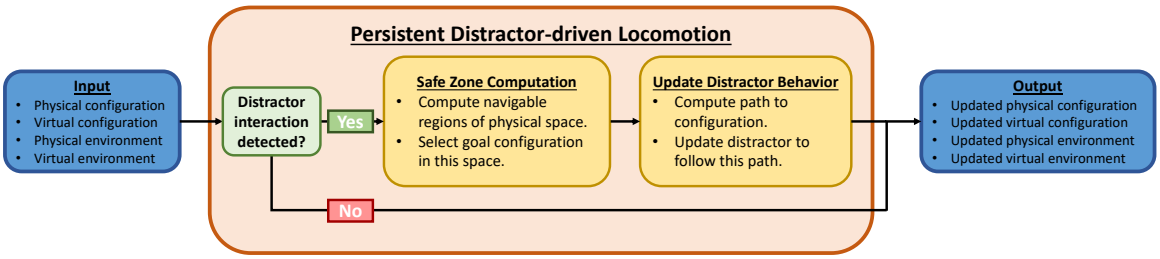


Figure 5.2: An overview of our persistent distractor-driven locomotion interface. Given as input the layouts of the physical and virtual environments and user's configuration in each environment, our system listens for an interaction between the user and a persistent distractor in the virtual environment. If an interaction is detected, the system computes the regions of the physical environment that the user can be safely guided towards (Subsection 5.3.3). Once a goal configuration to guide the user towards has been chosen, our system computes a path from the user's current physical configuration to the goal physical configuration and modifies the behavior of the distractor such that it guides the user along this computed path (Subsection 5.3.4).

made a rustling sound and animation at random intervals and when a frog entered or exited a bush. If the user reached the boundaries of the tracked space, a reset intervention was employed to maintain their safety and reorient them back towards the interior of the tracked space [229].

5.3.1 Definitions & Terminology

In VR, the user is located in a PE and a VE at the same time, with a particular position and orientation in each environment. We denote an environment as E, and the user's position and orientation as their configuration $q = \{p, \theta\}$, where $p \in \mathbb{R}^2$ is their position and $\theta \in [0, 2\pi)$ is their orientation in E. A user's trajectory in an environment is defined as an ordered set of configurations $Q = \{q^t\}_{t=0}^{T-1}$, where Q consists of T timesteps. We differentiate between the PE and VE using a subscript (e.g. q^3_{phys} is the user's configuration at timestep $t = 3$ in the physical environment E_{phys}). An environment is defined as a closed polygon $\mathcal{P} = \{v_1, v_2, ..., v_n\}$ where consecutive vertices are connected with an edge. Vertices are represented by an x- and y-coordinate. Obstacles in E are represented as holes in $\mathcal{P}$, which are also represented as ordered sets of vertices connected by edges. The set of physical configurations that yield a collision with

a physical obstacle is the obstacle space $\mathcal{C}_{obs}$, and the set of configurations that do not yield any collisions is the free space $\mathcal{C}_{free}$, where $E_{phys} = \mathcal{C}_{free} \cup \mathcal{C}_{obs}$. If the user gets too close to a physical obstacle (either an object or the boundaries of the tracked space), the locomotion interface must initiate a *reset* maneuver that reorients the user away from the nearby obstacle. The goal of a locomotion interface is to minimize the number of times the user undergoes a reset.

5.3.2 Distractor Interaction Detection

In order for a given VR application to be able to track interactions with distractors, the main requirement is that the application has a notion of what elements in the VE can serve as distractors, and that it can detect when the user begins to interact with one such element. In practice, such a feature is common since VR is an interactive technology and needs to track the user's interaction with the VE in order to have the virtual elements respond to the user's actions accordingly in order to maintain presence and plausibility illusions [176, 177].

User Study Implementation: In our application, a distractor interaction was triggered any time the user moved the virtual jar within 0.45m of the frog. If a distractor interaction was detected, the frog had a 90% chance to flee and jump towards a different bush in the VE. The 10% chance was chosen to allow users to catch *some* frogs so that they do not get too discouraged or frustrated due to repeated failed attempts.

5.3.3 Safe Zone Computation

Once an interaction between the user and a persistent distractor is triggered, our method computes the nearby regions of the PE that the user can be guided towards. These regions, which we call "safe zones," should be reachable without incurring a physical collision and, ideally, should not lead to a collision shortly after the user has reached it (e.g., being located in a corner

is likely to lead to a collision in the near future). In order to compute a safe zone in any arbitrary environment, the system requires some representation (a map) of the PE E_{phys}. From this map, we wish to compute the safe zones S which is a set of configurations in the PE that can be reached without incurring a collision:

$$S = \{q^*_{phys} \mid \exists\, Q^*_{phys} \in \mathcal{C}_{free}\}, \tag{5.3.3.1}$$

where $Q^*_{phys} = \{q^t_{phys}, ..., q^*_{phys}\}$ is a physical trajectory starting at the user's current configuration q^t_{phys} and ending at a goal configuration q^*_{phys} that the user can reach without any collision.

The difficulty in computing S will vary depending on the amount of information the system has about the PE. In the best case, the VR system has a global map of the user's entire physical space, in which case $S = \mathcal{C}_{free}$ and the user can be guided towards any navigable location in the PE using a distractor. In practice, most VR systems do not have access to a full map of the user's physical surroundings due to the limited range and noisy output of trackers. In this case, S will be the set of physical configurations that the system can *sense* and *can validate to be navigable to* at the current time step. For example, the visibility polygon defined at the user's location in the PE provides a well-defined set of configurations that are guaranteed to be reachable by definition [232]. In the RDW community, researchers have developed heuristics using techniques like artificial potential fields [11, 132, 205] and deep learning [30, 117, 192] to (explicitly or implicitly) compute safe regions of the PE that the user can be steered towards. However, in those RDW approaches, the computed safe zone is defined by the mechanics of the RDW algorithm (e.g., the potential function or the training data); in our formulation, the safe zone is simply any space in the PE that the user can reach without a collision and it does not depend on the mechanics of the collision-avoidance module (in our case, persistent distractors). This notion

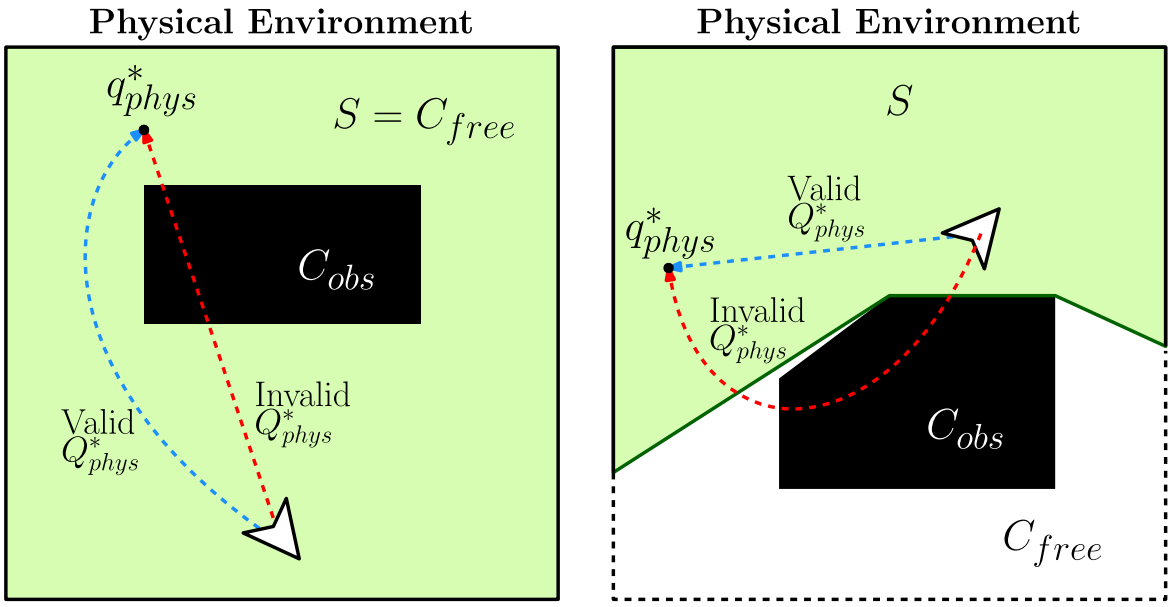


Figure 5.3: A visualization of the safe zone concept. The navigable space $\mathcal{C}_{free}$ is shown in green, and the regions that yield a collision $\mathcal{C}_{obs}$ are shown in black. In this figure, the user (white cursor icon) is heading towards a physical obstacle and should be guided towards a safe region of the physical space. An example point q^*_{phys} in the safe zone S is shown in the top left corner of the environment. A valid path that leads the user towards q^*_{phys} is shown in blue, and an invalid path is shown in red. **Left:** When the VR system has access to a full map of the user's PE, the safe zone S is equal to the entire free space $\mathcal{C}_{free}$. **Right:** When the VR system does not have a full map of the user's PE, the safe zone S can be a partial representation of the user's physical surroundings (subset of $\mathcal{C}_{free}$), such as the visibility polygon centered at their current position. Portions of the environment that are not known to the system are shown as black dashed lines.

of safe region computation is a well-studied problem in robotics that relates to control barrier functions [3] and simultaneous localization and mapping (SLAM) [25, 200]. An illustration of the safe zone concept is shown in Figure 5.3.

User Study Implementation: In our implementation, the physical space was a $4.3\text{m} \times 6.125\text{m}$ room with no obstacles inside it. Since we had access to a full map of the PE in our implementation, the safe zone computation module was treated as a pre-processing step and S was not computed at run time. Our PE was a simple convex shape, so the center of the PE is generally the safest region for the user to be, since it is the point that is on average furthest from all of the PE boundaries. However, one problem with guiding the user towards the center of a

convex environment, as is done in the steer-to-center (S2C) RDW algorithm [84, 157], is that as soon as the user reaches the center point, their next steps take them directly away from this safe zone (the center of the PE) and towards a boundary [157]. To resolve this issue, Razzaque et al. proposed the steer-to-orbit (S2O) and steer-to-multiple-targets (S2MT) methods that instead directs the user along a circular path or along a series of pre-determined waypoints, respectively [157]. Our distractor-driven locomotion is directly inspired by the S2O and S2MT algorithms when computing safe zones to steer the user towards. When a frog flees from a user, the safe zone is computed as one of four pre-determined points near the borders of the PE. These points are located along the major and minor axes of the PE, 75% of the way along the vector originating from the center of the PE and pointing in each of the cardinal directions (see Figure 5.4). If the user was located closer to the east or west points and a distractor interaction was triggered, the next safe zone was chosen randomly between the north and south points or vice versa. Thus, the frog jumped to a bush in the VE that was closest in proximity to the chosen safe zone in the PE (details in Subsection 5.3.4).

5.3.4 Update Distractor Behavior

Once a safe zone has been computed, the next step in our method is to modify the persistent distractor's behavior such that it influences the user to change their *virtual* trajectory to move towards the safe zone. In order to do this, a path Q^*_{phys} from the user's current physical configuration q^t_{phys} to the goal configuration q^*_{phys} must be computed. Since the walkable physical space $\mathcal{C}_{free}$ is a topological space, this path can be formulated as the function $f : [0, 1] \rightarrow S$ such that $f(0) = q^t_{phys}$ and $f(1) = q^*_{phys}$. Intuitively, this function is simply the trajectory that the user must travel on, starting at q^t_{phys} and ending at q^*_{phys}, in order to continue their interaction with

the distractor and avoid a physical collision. Computing this path can be formulated as a motion planning problem and there exist many different algorithms for computing such a path [115]. Once f has been computed, the system must modify the behavior of the distractor such that the user will walk along f as they continue to interact with the persistent distractor. In practice, this usually means changing the trajectory of the distractor to move closer to a configuration q^*_{virt} that is closely aligned with q^*_{phys}, at which point the user will follow the distractor and reach the safe zone. Note that in general, it cannot be guaranteed that the q^*_{phys} corresponds to a reachable section of the VE. If the user cannot reach the safe zone without breaking the plausibility or presence illusion in VR, an alternate safe zone should be computed to guide the user towards. This concept is illustrated in Figure 5.5.

User Study Implementation: Once a safe zone has been computed, our system directs the frog being interacted with to jump towards the bush in the VE that is closest in proximity to the chosen safe zone, i.e., the frog jumped in a straight-line trajectory to a new configuration q^*_{virt} that corresponds most closely to q^*_{phys}. In this manner, the user is encouraged to walk on a straight path that guides them close to q^*_{phys}.

5.3.5 Trade-offs Between Collision Avoidance & Immersion

So far, our system has been presented under the assumption that the only purpose of distractors is to help the user avoid collisions. In practice, however, this is not the case. Since VR is an immersive experience and maintaining a sense of presence is important for delivering an effective virtual experience, a distractor-driven locomotion interface should ideally be designed so as to avoid creating breaks in presence (BiPs) for the user. This can be achieved in part by using context-aware distractors, as opposed to out-of-place distractors that are not congruent with the

virtual surroundings [147]. However, the feasibility of safely guiding the user also depends on the goal configuration q^*_{phys} and trajectory chosen (Subsection 5.3.4). Consider, for example, a guided virtual house tour. If our system computes an optimal goal configuration q^*_{phys} that corresponds to a region of the virtual house that has already been visited, it may be difficult to justify guiding the user back to a location they have already visited (since there is likely nothing new to see there). Worse still, if q^*_{phys} corresponds to an invalid location in the VE (e.g. inside of a virtual object), there may be no realistic options for redirecting the user to q^*_{phys}. Thus, when computing q^*_{phys} and the path f, it is important to consider not only the likelihood of avoiding collisions, but *also* the user's level of presence during the distractor-guided navigation. Estimating the user's level of presence beforehand can be difficult, so we recommend conducting pilot studies to evaluate users' level of presence (e.g. via the Slater-Usoh-Steed presence questionnaire [210]) and adjusting the implementation accordingly.

5.4 Experiments & Results

To measure the efficacy of our method, we used *average virtual distance walked between resets* as the main performance metric. We used this metric because it is a path length-normalized measure for the number of resets a user incurs. If our distractor-driven interface is effective at reducing the frequency of resets, we expect a larger distance walked between resets. We also collected qualitative data in the form of observations during experimentation and informal interviews with participants, and we discuss that data in Section 5.5.

5.4.1 Experiment 1: Comparison with RDW

When being directed by our implementation of the distractor-driven framework, users walked an average of 7.864m $(\sigma = 2.085)$ between resets. When directed by the randomized

distractor behavior, users only walked an average of 6.269m ($\sigma = 1.657$) between resets. Results of a dependent samples *t*-test indicated that this difference was statistically significant ($t(15) = 2.439, p = 0.0276, r = 0.533$). This result indicates that our framework can be used to create a locomotion experience yields improved collision-avoidance performance over an implementation with purely random distractor behavior paired with RDW. This result supports our hypothesis for Experiment 1.

5.4.2 Experiment 2: Impact of Collision-avoidance Bias

When users were directed using our collision-aware distractors, but with only a 30% chance for distractors to behavior optimally, users walked an average of 6.881m ($\sigma = 1.710$) between resets. When directed with RDW and random distractor behavior, users walked an average of 6.289m ($\sigma = 1.541$) between resets. Results of a dependent samples *t*-test indicated that this difference was *not* statistically significant ($t(15) = 1.243, p = 0.233, r = 0.306$). By increasing the likelihood that the distractors follow a naïve behavior (jumping to a random bush instead of to a bush near q^*_{phys}), users spent more time chasing frogs to locations outside of the PE, which caused them to incur a similar number of resets to the purely random behavior condition. This result highlights the importance of having distractors execute behaviors that guide users towards physical safe zones to avoid collisions. This result supports our hypothesis for Experiment 2.

5.4.3 Experiment 3: Impact of Distractor Behavior Feasibility

When users were directed by our collision-aware distractors (with a 90% chance to follow collision-aware behavior, as in Experiment 1) but the VE constrained the distractors' ability to guide the user towards q^*_{phys}, users travelled an average of 5.940m ($\sigma = 1.353$) between resets. In the same constrained VE, with randomized distractor behavior, users walked a similar average

distance of 5.383m ($\sigma = 0.682$) between resets. Results of a dependent samples *t*-test indicated that this difference was not statistically significant ($t(15) = 1.428, p = 0.174, r = 0.346$). This result highlights the importance of designing a VE that allows the distractors to execute diegetic behaviors that can still guide the user towards q^*_{phys}. This result supports our hypothesis for Experiment 3.

5.5 Discussion

Although Experiment 1 showed that our implementation can be an effective method for guiding the user along collision-free trajectories, it is important to consider how the different components of our framework affect the final locomotion experience. In particular, how the distractor behaves (Subsection 5.3.4) and the reachability of safe zones (Subsection 5.3.3) have a significant impact on collision avoidance. In Experiment 2, we modified how likely the frogs were to follow a collision-aware strategy or not. The results of this experiment showed the importance of having distractor behaviors that guide the user towards safe zones often enough; without collision-aware distractors, the user's exploration of and interaction with the virtual environment is continuously interrupted by resets. In Experiment 3, we evaluated how important it was for the distractors to be able to guide the user towards a virtual configuration that was aligned with the chosen safe physical configuration q^*_{phys}. By decreasing the density of bushes and adding obstacles in the VE, the frogs that the user chased were less likely to be able to reach a location that aligned well with q^*_{phys}. Results of this experiment showed that, on average, inability to effectively guide the user towards q^*_{phys} had a notable impact on the locomotion experience, such that the number of resets the user incurred was *the same* with and without our collision-aware distractor guidance (indicated by the lack of significant differences in Experiment 3).

To better understand how the distractor behavior impacted users' trajectories, we show example trajectories in Figure 5.5. In Figure 5.5, we show how the different parameters of the distractor behavior (Experiments 1-3) influence the user's trajectory in the PE and VE. In particular, we see that participants were able to visit roughly the same amount of different locations in the VE, but that the user incurred significantly fewer resets when the distractors behaved in an ideal manner (Experiment 1). Interestingly, the user did not spend much of their time in one location of the VE chasing the same frog back and forth between safe zones. However, in Experiment 3, participants spent most of their time walking long distances in the VE to reach the next bush that a frog had jumped to.

During experimentation, we also collected qualitative data in the form of observations made by the researcher running the experiment and comments made by participants during or after the experiment. In general, participants' main strategy was to scan the VE in place to find a frog, and then walk towards that frog to catch it. If users could not find any frogs, they started to walking to explore the VE in search of frogs. Participants had different catching strategies as they approached the frogs: some participants tried crouching down and approaching very slowly, some participants always approached the frogs from behind in an attempt to remain out of the frog's field of view, and other participants tried slowly approaching and then quickly thrusting the virtual jar at the frogs.

Participants had mostly positive feedback about the experiment, noting that they found it fun and often did not notice any redirection. When the behavior of the frogs was explained to them after completing the experiment, most participants commented that they did not realize they were being redirected by the frog behavior, which adds further support for the strength of distractors for collision avoidance. One participant reported that the collision-aware distractors

steering felt "way better" and that he hardly noticed the reset screen in the collision-aware distractor condition. After completing the experiment, another participant commented that he remembered thinking he had been "walking for a really long time" without incurring any resets while searching for frogs. Participants also reported trying to change their strategy after repeatedly failing to catch any frogs, which supports the notion that our distractors were effective at occupying their attention and suggests that changing the distractor behavior can strongly influence users' behavior. Interestingly, some participants assumed that if a bush rustled, it meant that a frog was hiding in that bush (which was not necessarily true). Once they realized that a bush rustle does not always signify the presence of a frog, they eventually stopped paying attention to these rustles (not all participants made this realization). This highlights the importance of considering *all* distractors in a VE, since it is clear that even auxiliary elements of the VE that are not core to the virtual task can capture the user's attention and influence their behavior. Furthermore, it suggests that some distractors are more "powerful" than others when it comes to collision avoidance, since only some participants were influenced by the bush rustling, and only some of these influenced users eventually started ignoring the rustling distraction. Finally, some participants expressed frustration due to the low chance of successfully catching frogs. Users also sometimes became discouraged from chasing the frogs if they jumped too far away, since they did not want to walk a long distance only to fail again. Overall, the qualitative data provide support for the notion that a distractor-driven locomotion interface can be effective for helping the user avoid collisions *without* interfering with their virtual experience *and* supporting natural walking as a locomotion method.

5.5.1 Additional Implementation Examples

To help other researchers understand how our distractor-driven interface can be applied to applications other than our frog-catching game, this section provides high-level descriptions of other VR applications and how our framework can be integrated into them.

Virtual Guided House Tour: In a virtual guided tour application, the user may wish to explore a virtual house that they are considering purchasing. In such an application, a virtual human companion would serve as a tour guide and would take the user to different rooms in the house. The persistent distractor would be the tour guide, and they could provide distraction cues in the form of instructions (*"Please follow me so that I can show you the kitchen."*) and walking in a direction that avoid collisions in the physical space.

Search and Rescue Training: In a search and rescue training application, the user could be learning how to carry out search and rescue tasks as part of a natural disaster response team. The user could be paired with a virtual dog companion that takes them to areas of interest in the VE (e.g., towards a person buried under rubble who calls out in need of assistance) that correspond with safe regions of the PE.

Hide and Seek Game: In a hide and seek game, the user can be assigned either a "hide" role or a "seek" role. When the user plays as the seeker, the persistent distractors would be virtual agents that the user must search for and capture (similar to the frogs used in our application). However, if the user plays as a hider, the distractor would be the seeker. As the seeker approaches closer and closer to the user in the VE, they would be encouraged to move to a new virtual location that is further away from the seeker. In such a situation, the distractor would actually serve as a *deterrent* that the user wants to *avoid*, instead of serving as something that the user wants to

chase in order to engage with it in close proximity.

5.6 Conclusion, Limitations, and Future Work

In this work, we presented a framework that allowed users to explore large VEs, while located in a small PE, using natural walking. Our framework is centered around the observation that VR experiences are *interactive* and that we can influence the user to alter their intended virtual trajectory such that it is less likely to yield a collision in the physical space. To do this, we control the behavior of *persistent distractors*, which we define as elements of the VE that the user continually interacts with throughout the course of their virtual experience. In particular, when our system identifies that the user has started an interaction with a distractor, we identify a safe location in the PE that the user can be guided towards and then modify the distractor's behavior to move to a virtual configuration that is aligned with the identified safe location in the PE. In practice, the user changes their trajectory in the VE to follow the distractor to its new virtual location, which also guides the user towards to safe physical location and away from collisions with physical obstacles. To verify the effectiveness of our framework, we conducted three experiments using a frog-catching game that implements our proposed framework. Results of our experiments showed that our approach is effective for reducing the frequency with which users get too close to physical obstacles and demonstrated that continuous engagement with context-aware distractors is a powerful tool for collision avoidance.

Despite these benefits, our work has some limitations. First, our distractor-based algorithm relies on being able to accurately compute the safe regions of the physical surroundings in order to know where the user should be guided. Developing such an algorithm that works across a variety of physical environments and on different VR hardware is difficult due to limitations in real

time sensing, reconstruction, and simultaneous localization and mapping [200]. Additionally, the exact implementation of distractors will depend greatly on the virtual experience and the user's particular configuration in the PE and VE, which can make it difficult to understand exactly how reliable our approach is for collision avoidance in a variety of different experiences and user configurations. Finally, the ratio between male and female participants in our experiment was not balanced, which may have created some bias in our results [150, 167].

Future work in this area should study to what degree distractor-driven locomotion can be implemented using not just visual but also aural, haptic, and cognitive distractors.Additionally, extensive studies on the impacts of the different components of our framework on collision-avoidance should also be studied, similar to the experiments we conducted in the present study. Additionally, more advanced methods for computing safe zones and user behavior modelling and prediction will lead to significant improvements.

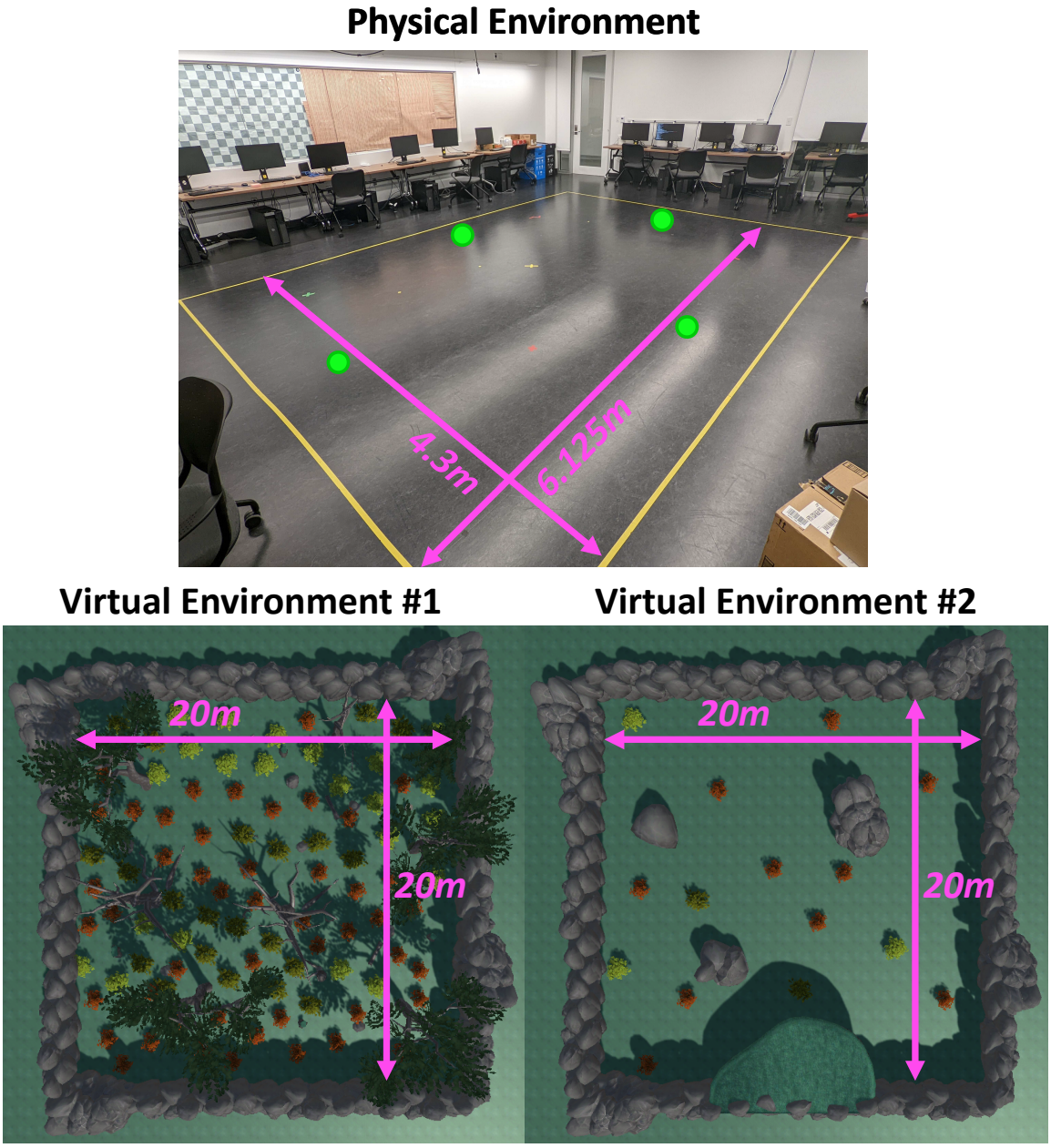
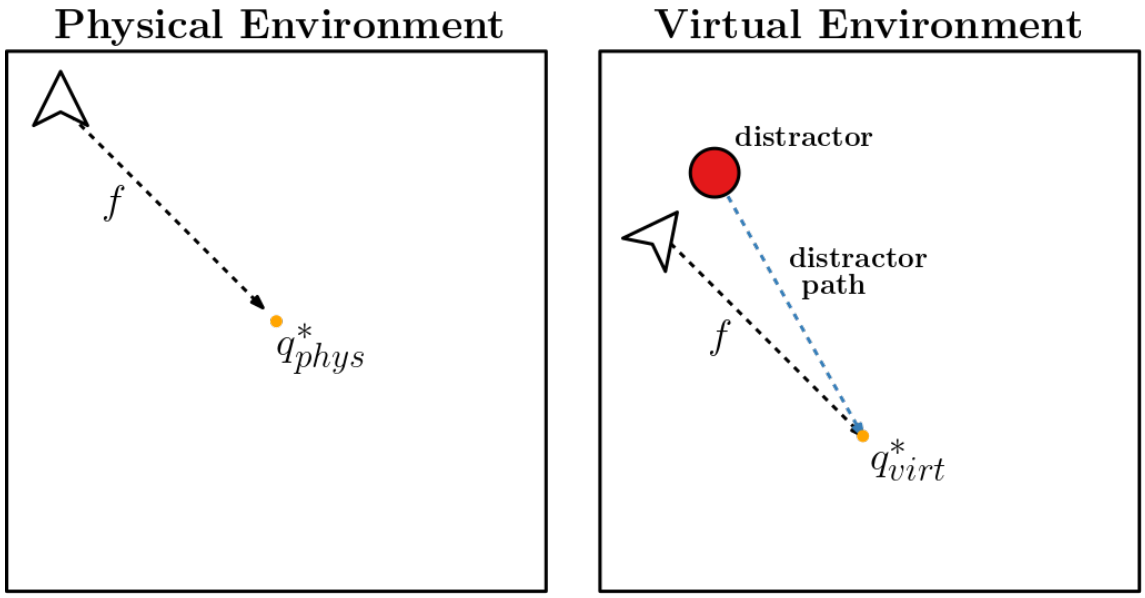
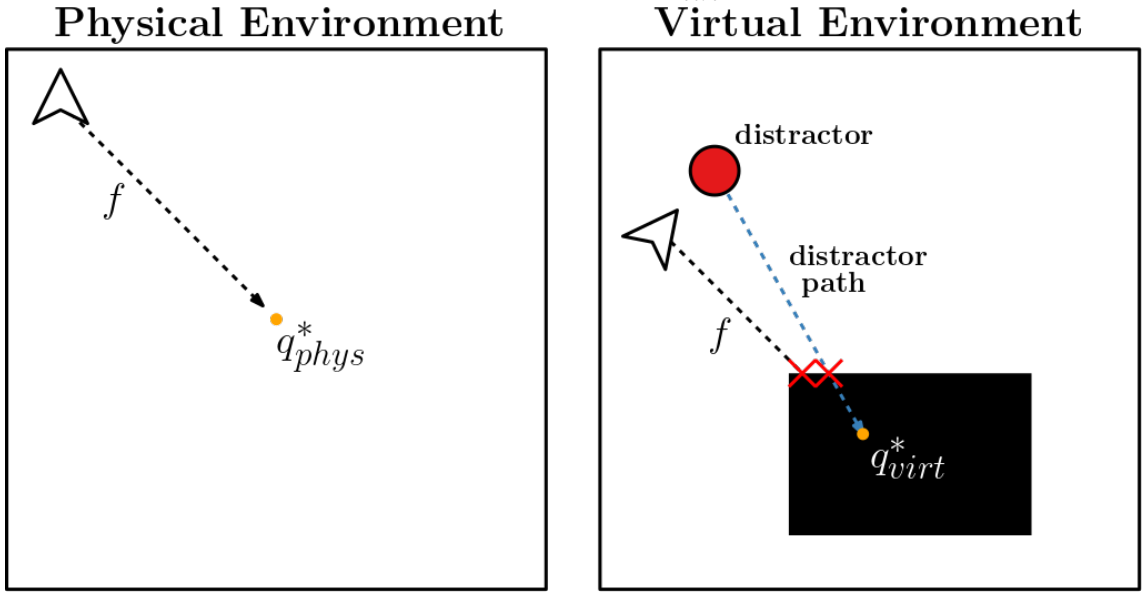


Figure 5.4: The physical and virtual environments used in our implementation. The physical environment (top) was a $4.3\text{m} \times 6.125\text{m}$ space. The green dots represent the four pre-computed safe zones that users were guided towards by the persistent distractor (frogs). The first virtual environment (bottom left), used in Experiments 1 and 2, was a $20\text{m} \times 20\text{m}$ environment with bushes, trees, and other miscellaneous forest objects. The second virtual environment (bottom right), used in Experiment 3, was a $20\text{m} \times 20\text{m}$ environment with significantly fewer bushes and large rocks and a plateau that obstructed the movements of our persistent distractors.



(a) Configurations of the PE and VE that allow the distractor to guide the user towards q^*_{phys}. The path that takes the user to q^*_{phys} is represented by the function f (black dashed arrow). The blue dashed arrow in the VE represents a path through the VE that the distractor could travel on in order to guide the user towards q^*_{phys}. Since there are no virtual objects that obstruct the distractor's path, the distractor can simply follow the trajectory f to reach q^*_{virt}.



(b) Configurations of the PE and VE that do not allow the distractor to guide the user towards q^*_{phys}. The virtual configuration q^*_{virt} that corresponds with the physical goal configuration q^*_{phys} lies *inside* of a virtual object. Thus, any path from the distractor to q^*_{virt} is invalid because it would create an unrealistic experience that would break the user's sense of presence in the VE (clipping through a virtual object).

Figure 5.5: A diagram illustrating how the distractor behavior can be used to guide the user towards the desired goal configuration q^*_{phys}. When the user gets too close to a physical obstacle and needs to be guided back to a safe location in the PE, our algorithm updates the behavior of a nearby persistent distractor in the VE to naturally guide the user towards the virtual configuration q^*_{virt} that corresponds with q^*_{phys}.

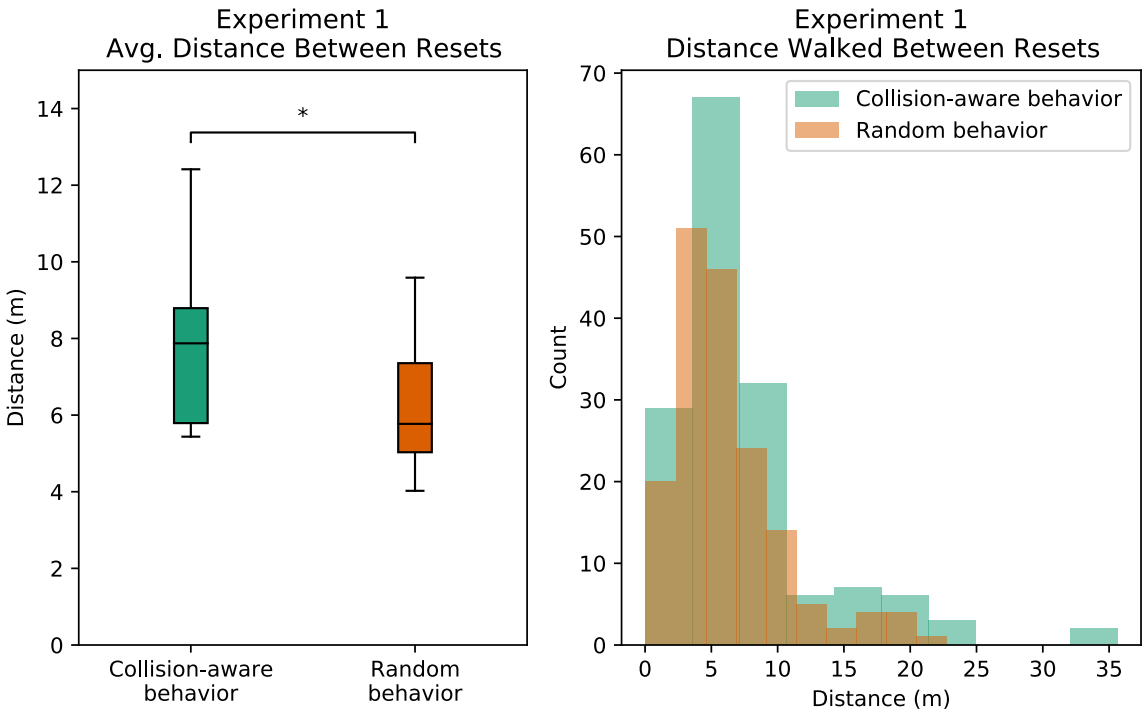


Figure 5.6: **Left:** Box plot of the average distance walked between resets for each participant, grouped by the experiment condition. Results show that, compared to randomized distractor behavior, participants walked significantly further (7.864m vs 6.269m) before incurring a reset when they navigated through the VE using our collision-aware persistent distractors ($^* p < 0.05$). **Right:** A histogram of the distances of paths travelled between resets, grouped by experiment condition. Paths that were influenced by a collision-aware distractor were on average longer than those traveled when interacting with random distractors and RDW.

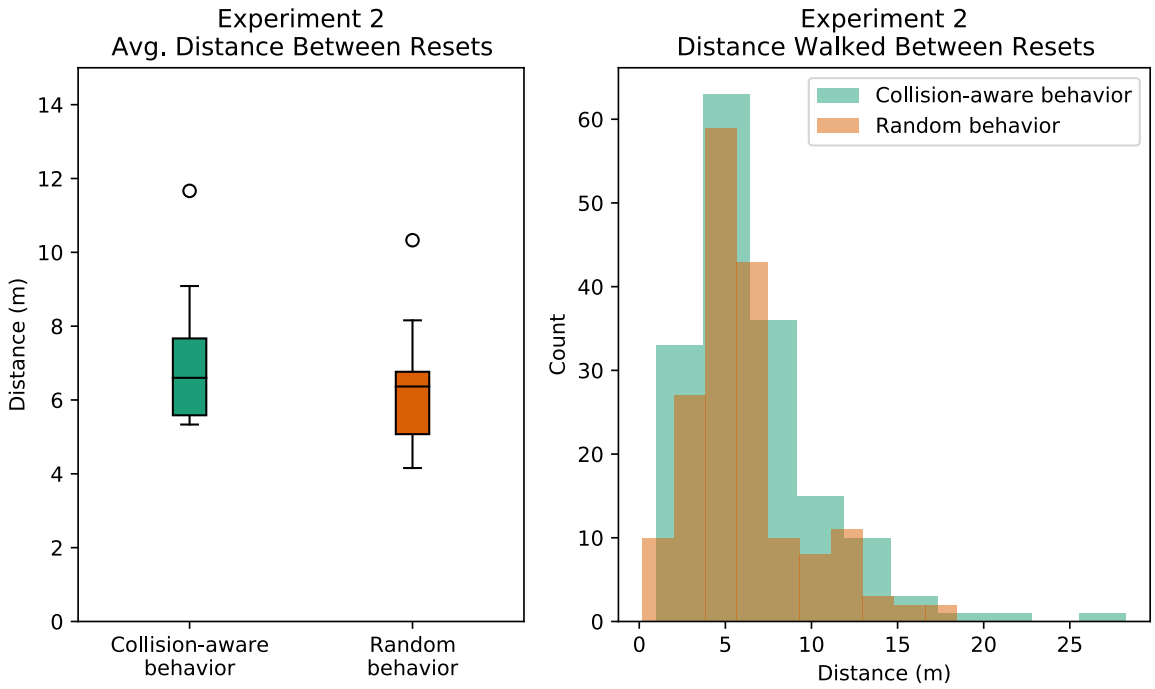


Figure 5.7: **Left:** A box plot of the average distance walked between resets for each participant, grouped by the experiment condition. In Experiment 2, the likelihood that distractors followed a collision-aware trajectory was reduced, to show the impact of this parameter on the user's locomotion experience. Compared to random distractor behavior, participants traveled, on average, similar distances while being guided with our collision-aware distractors. The lack of significant differences highlights the importance of generating distractor behaviors that try to guide the user towards safe zones. **Right:** A histogram of the distances of paths travelled between resets, with and without our collision-aware distractor behavior. Unlike Experiment 1, the two distributions have more overlap, indicating that users walked roughly equal distances between resets regardless of the distractor behavior.

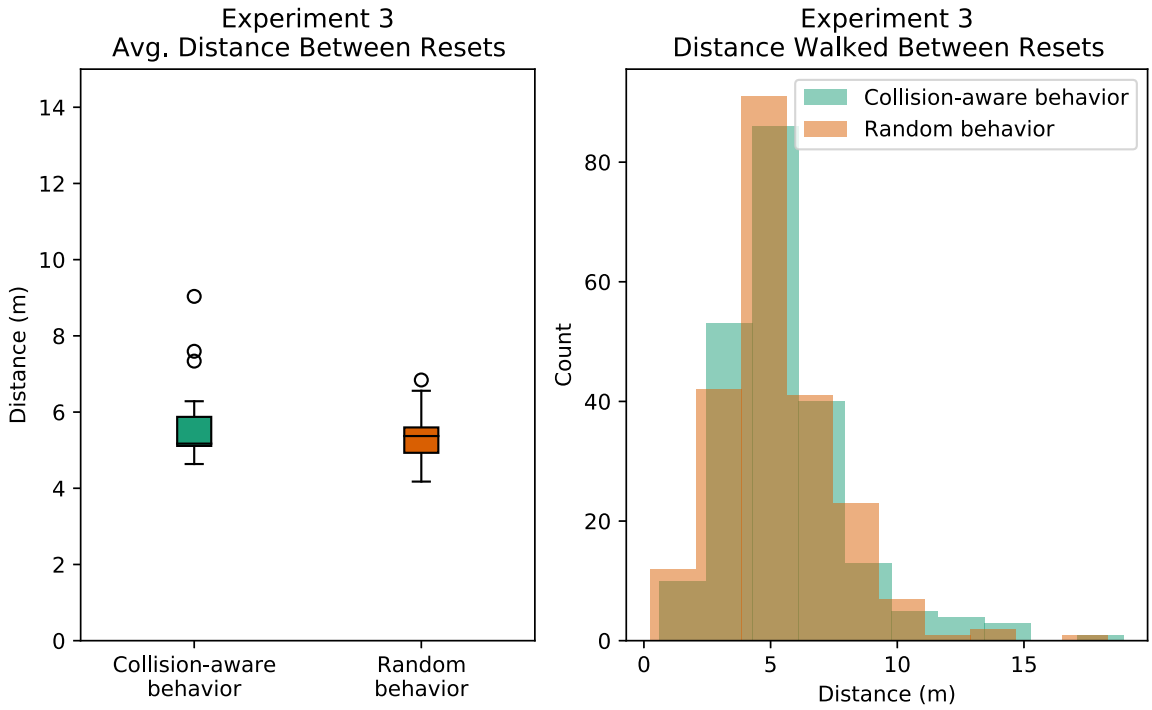


Figure 5.8: **Left:** A box plot showing the average distance walked between resets for each condition in Experiment 3. In Experiment 3, the VE was modified so that it was harder for distractors to execute diegetic behaviors that could guide the user towards the safe zone. We see that participants travelled a similar distance regardless of the presence of collision-aware or naïve distractors. This lack of significant differences highlights the importance of having a virtual experience in which the persistent distractors can reliably guide the user towards a virtual location that corresponds closely to the physical safe zone. **Right:** Histogram of the distances of paths walked between resets. Similar to Experiment 2, we see a high amount of overlap between the two distributions, which highlights that the users' locomotion patterns were similar despite the use of collision-aware distractors in one of the conditions.

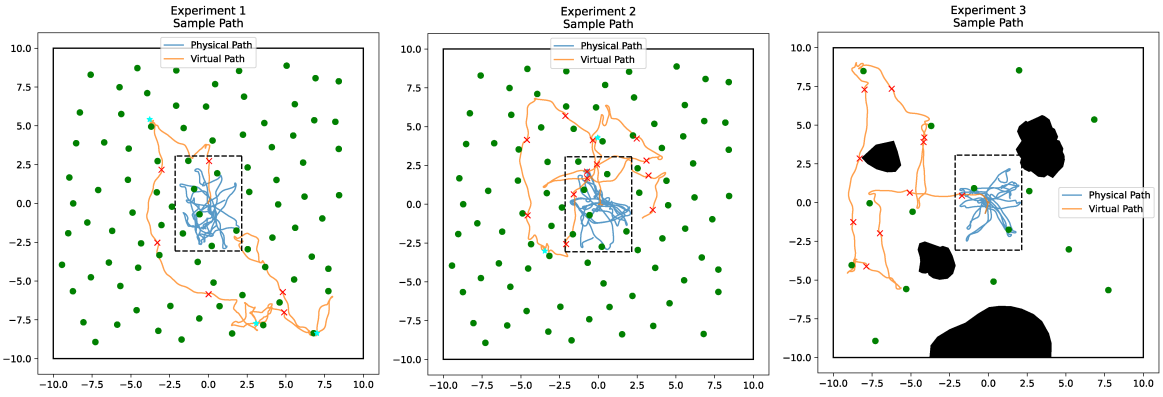


Figure 5.9: Plots of an example path that a participant travelled for each experiment. In each plot, the user is being guided using our distractor-driven interface. Green circles represent bushes in the VE that frogs jumped to, red X's represent positions where the user incurred a reset, teal stars represent positions where the user caught a frog, and black shapes represent obstacles. The boundary of the PE is the dashed black rectangle ($6.125\text{m} \times 4.3\text{m}$), and the boundary of the VE is the solid black square ($20\text{m} \times 20\text{m}$). The user's physical and virtual trajectories are shown in blue and orange, respectively. **Left:** When frogs had a 90% chance to flee and 90% chance to choose a destination using our method (as in Experiment 1), the user spends some of their time following the frog around a small area of the VE, then chases the frog to another location in the VE if it randomly jumps to a far-away location, or searching for new frogs after catching one. **Middle:** When the frogs had a 90% chance to flee and only a 30% chance to choose a destination using our method (as in Experiment 2), the user incurred more resets than in Experiment 1. **Right:** When the frogs had a 90% chance to flee and a 90% chance to choose a destination using our method but it was harder for the distractor to reach a location that aligned closely to the goal physical configuration q^*_{phys}, the user incurred a similar number of resets as in Experiment 2 and was not able to catch any frogs.

Chapter 6: Quantifying Environment Navigability for Natural Walking in Virtual Reality

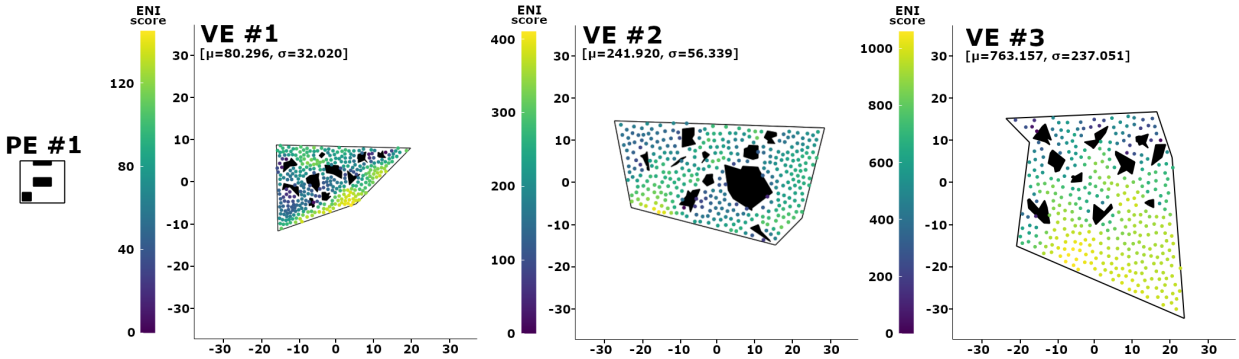


Figure 6.1: A visualization of our Environment Navigation Incompatibility (ENI) scores for physical environment paired with three different virtual environments with increasing environment area. Our metric is used to accurately quantify whether it is possible to compute a good mapping between the geometric layouts of these environment. We sample points across the virtual environment to represent the user's position (shown as colored circles) and compute the corresponding point in the physical environment based on comparing the local neighborhoods. Overall, our metric helps us to determine which regions or subsets of the VE are more compatible with the PE. Through this visualization, we can see which regions of the VE are more or less compatible with the PE.

In this chapter, we present a novel metric to analyze the similarity between the physical environment and the virtual environment for natural walking in virtual reality. Our approach is general and can be applied to any pair of physical and virtual environments. We use geometric techniques based on conforming constrained Delaunay triangulations and visibility polygons to compute the Environment Navigation Incompatibility (ENI) metric that can be used to measure the complexity of performing simultaneous navigation. We demonstrate applications of ENI for highlighting regions of incompatibility for a pair of environments, guiding the design of the

virtual environments to make them more compatible with a fixed physical environment, and evaluating the performance of different redirected walking controllers. We validate the ENI metric using simulations and two user studies. Results of our simulations and user studies show that in the environment pair that our metric identified as more navigable, users were able to walk for longer before colliding with objects in the physical environment. Overall, ENI is the first general metric that can automatically identify regions of high and low compatibility in physical and virtual environments.

6.1 Introduction

Locomotion, the ability to explore a space, is a fundamental task in virtual reality (VR). Locomotion interfaces are techniques that enable a user to explore a virtual environment (VE). The main goal of locomotion interfaces is to allow users to comfortably and safely explore the VE, which may be very large and dynamic, while they are located in a small physical environment (PE). While many locomotion interfaces have been developed [52], interfaces that allow users to explore VEs using natural walking are often preferred since they afford a higher sense of presence [209] and tend to lead to better performance at tasks in VR applications [84, 162]. Although natural walking interfaces have many benefits, not all virtual tasks and pairings of physical and virtual environments are best suited for a natural walking interface [52, 193]. If the PE is prohibitively small or has a high number of obstacles, the user may have a more comfortable virtual experience with a locomotion interface that does not involve natural walking (such as teleportation or joystick movement).

A key issue with respect to locomotion interfaces that use natural walking is to determine whether a particular pair of physical and virtual environments (denoted $\langle$PE, VE$\rangle$) is amenable

to collision-free locomotion. It is well-known in the environmental psychology community that the layout (i.e., the geometric structure) of an environment influences the shapes of the paths that a users travel [69, 175, 228]. Indeed, similar studies have shown that the user's perception of a virtual environment's complexity and navigation also depends on its layout [47, 65]. Although we have some understanding of the effects of environment layout on navigation in either PE or VR, a key issue with such locomotion interfaces is simultaneous exploration of a physical *and* virtual environment. Some prior work has studied the effects of environment layout on the feasibility of collision-free navigation with natural walking in the context of redirected walking [158]. However, such studies are limited due to ambiguity in terms of how they define the layout of the environments, or lack of simultaneous consideration of the layouts of the PE and VE *relative to each other*.

Main Results: We address the problem of understanding the influence of environment layouts on the VR locomotion experience based on natural walking. Our goal is to accurately quantify to what degree the layouts of a given PE and VE influence a user's ability to avoid collisions during locomotion. We introduce an Environment Navigation Incompatibility (ENI) metric, which quantifies the difficulty of performing collision-free VR navigation in a given $\langle$PE, VE$\rangle$ pair. ENI works by uniformly sampling locations across the PE and VE, and computing the most compatible physical location for each sampled location in the VE. This compatibility computation is based on the visibility polygon [49], which characterizes the local structure of an environment around a location. We formulate ENI on the visibility polygon due its ability to characterize environment layout, and to capture local features of an environment, which are also used by humans to navigate through environments. The final output of ENI is an n-dimensional vector of real numbers, where n is the number of sampled locations in the VE. We compute the

mean and standard deviation of this n-dimensional vector and automatically create interactive visualizations to summarize the output of ENI and make it more interpretable. Using ENI, we can better understand how different regions of the VE and PE contribute to collision-free navigation. ENI highlights regions of low and high compatibility between the PE and VE without requiring us to collect any locomotion data in the environment pair. To summarize, our main contributions are:

- A novel metric that quantifies the ease of collision-free navigation for a given pair of physical and virtual environments. ENI is based only on the geometric layout of the environments, making it computable for any static ⟨PE, VE⟩ pair, assuming the layouts of the environments are known. ENI is the first general VR navigability metric that simultaneously considers the layouts of the PE and VE relative to each other.

- We highlight multiple benefits of ENI, including analyses of how changes in the VE influence navigability, guidelines on how to design VEs to be more amenable to navigation for a fixed PE, and evaluation of the performance of RDW controllers.

- Evaluation of ENI using extensive simulations and two user studies. We validate that ENI is capable of identifying ⟨PE, VE⟩ pairs that are amenable to collision-free navigation *without* the need for any locomotion data.

6.2 Background and Prior Work

6.2.1 Navigability Metrics

In this work, we define the navigability of an environment as the average distance an agent can walk before colliding with an obstacle, in all directions across all positions in the

environment. Extending this to VR, where the user is simultaneously located in a PE and VE, the navigability of a ⟨PE, VE⟩ pair is the average distance the user can walk before colliding with a *physical* obstacle, in all directions across all positions in both environments. Our goal is to develop a metric that can quantify this notion of navigability for a ⟨PE, VE⟩ pair. Navigability in VR depends on many factors, including the layouts of the environments, the user's path through the VE, and the user's cognitive load during locomotion. While all of these features are important to consider when assessing navigability, in this work we only study the effect of the environments' layouts. In particular, we use the term "navigability" to refer to the difficulty of collision-free navigation; it is also common for researchers to use the term "complexity" to refer to the same idea.

Quantifying navigability has been studied in related fields, including robot navigation and environmental psychology. The factors that contribute the most to navigability depend on the domain in which navigability is being evaluated. Thus, when discussing navigability metrics, it is important to consider the context in which the metric is being developed, since this context will influence which features a metric emphasizes.

6.2.1.1 Environmental Psychology

Researchers in environmental psychology have developed metrics to better understand how humans perceive the navigability of indoor spaces. Wiener et al. [228] characterized environment complexity using geometric properties of isovists (also known as visibility polygons), which are the 2D planar region of space in an environment that can be seen from a given location. They found a correlation between participants' perception of the complexity of the environments and some properties of isovists in these environments, such as isovist jaggedness and area.

Stamps [185] explored the relationship between isovist properties and humans' perception of enclosure or permeability of urban environments. Stamps [184] also conducted a meta-analysis that found correlations between perceptions of enclosure of the environment and properties of a human's location, such as horizontal distance to the nearest obstacle. To better understand the relationship between human navigation and layouts for the entire environment (as opposed to only local features that are captured with isovists), researchers have proposed space syntax measures [79, 80]. Haq et al. [75] and Peponis [151] showed relationships between the global structure of environments and humans' navigation behavior through them. Our approach is motivated by these prior methods and our metric is designed to quantify environment structure both on a local and global scale using isovists (visibility polygons) and random sampling, respectively.

6.2.1.2 Locomotion in Virtual Environments

One of the most popular locomotion interface that enables real walking is redirected walking (RDW) [158]. Thus, in this section we focus mainly on prior work studying real walking in VR using RDW. There is some work on understanding how the shape of an environment influences the efficacy of the RDW steering algorithm. Azmandian et al. [7] studied the effect of the size and shape of the tracking space on the number of times that users have to orient away from physical obstacles. Messinger et al. [132] studied the effect of the size of the tracking space and shape on the number of resets during RDW. They considered square PEs of varying sizes in addition to PEs with different shapes including rectangular, trapezoidal, cross- and L-shaped. Their results showed that users were able to avoid more collisions as the PE size grew. Moreover, non-convex PEs like the cross and L-shaped rooms lead to more collisions than the convex PEs. Lee et al. [119] also studied how RDW algorithms perform as the shape and size of the PE

changes. In their work, they considered square PEs of varied sizes, as well as PEs in the shape of a square, trapezoid, cross, circle, T, and L, each with a roughly equal area. Lee et al. [119] observed that larger PEs lead to fewer collisions and that non-convex PE shapes like cross, T, and L lead to more collisions than the convex PEs. Williams et al. [231] introduced the Complexity Ratio (CR) metric to quantify the navigability of a ⟨PE, VE⟩ pair. CR is defined as the ratio of the average distance to the nearest object in the PE and VE, averaged across many sampled points. They showed that as the environments become more complex, users incur more resets. Our Environment Navigation Incompatibility metric is more general than the methods described in this section, and provides a more accurate indication of the navigability of a given ⟨PE, VE⟩ pair.

6.2.2 Shape Similarity

There is considerable work on shape analysis of objects and environments in geometric computing. The more similar a PE and VE are in terms of geometric shape, the more likely it is that a collision-free path in the VE corresponds to a collision-free path in the PE. Therefore, any measure on the the similarity of a PE and VE provides a proxy to measuring the likelihood that a user can travel on collision-free paths in that ⟨PE, VE⟩ pair.

Many metrics have been proposed for shape similarity. The Hausdorff distance pairs points from one shape to points on the other shape, and the distance measure is the longest distance between a pair of points [91]. Another popular method for comparing polygons is the turning function, which parameterizes a polygon by the lengths of its edges and the interior angles between adjacent edges, which simplifies the problem to the comparison of 1D functions [5]. Symmetric difference is a similarity measure that takes into account the area of overlap of

the two polygons. For two polygons A and B, the symmetric difference metric is defined as $\texttt{area}((A - B) \cup (B - A))$ [218]. In our approach, we also use the metric $\texttt{area}(A - B)$ to measure the similarity of two visibility polygons that represent the user's position and orientation in an environment.

Shape similarity for 3D objects and environments has also been extensively studied [17, 28]. Osada et al. [142] introduced the notion of a *shape function*, which is a function that characterizes the shape of an object when it is computed over a sufficiently dense set of random points on the object's surface. Using such a function, Osada et al. computed histograms of shape functions for different objects and reduced the 3D shape similarity problem to comparing histograms. Our approach is also motivated by such techniques and we define appropriate shape functions to characterize the structure of an environment. Moreover, we compare pairs of physical and virtual shape function values to measure the similarity of the PE and VE layouts. Shape correspondence is a problem that is closely related to shape similarity. In the correspondence problem, we wish to compute a mapping of features (such as points on a surface) of one object to features on the other object [213]. When computing corresponding features, the mapping is usually defined according to geometric properties, and the mapping can be constrained in different ways, such as being a one-to-one or one-to-many mapping. In our approach, we take inspiration from shape correspondence literature by finding the corresponding location in one environment that has the best "local similarity" to a given location from the other environment.

Analogous to the geometric shape similarity problem, the robot motion planning community has developed metrics to quantify the layouts of environments with respect to collision-free navigation. Anderson et al. [4] proposed a metric that is a combination of the entropy and compressibility of the environment. Similar to entropy, Crandall [45] used the branching factor

and environment clutter as a measure of complexity for maze environments. Shell et al. [169] borrowed concepts from space syntax [79, 80] to define complexity in terms of the distance between adjacent convex regions of the environment. El-Hussieny et al. [59] focused on robots that explore unknown environments, and proposed an environment complexity metric that measures the difference between the expected and actual number of locations the robot needs to visit in order to map out the entire environment.

6.3 Environment Navigation Incompatibility Metric

Our goal is to formulate a metric that quantifies a user's ability to navigate with collision-free paths in a given ⟨PE, VE⟩ pair. Since our driving application is VR locomotion with RDW, our metric is designed such that it accounts for the factors that are important for RDW. Currently, we only take into account the geometric layouts of the PE and VE. In VR locomotion, the user's ability to walk on collision-free paths depends primarily on their proximity to obstacles in both environments, which is defined by their relative position and orientation. If the layouts of the PE and VE are similar, the user's proximity to the obstacles would exhibit similar characteristics and thereby make it easier to navigate without collisions. We consider a ⟨PE, VE⟩ pair to be *compatible* for RDW when they have a high degree of similarity. In this section, we detail the steps involved in formulating our ENI metric and analyze them.

6.3.1 Environment Representation

Since our metric is based only on the layouts of the PE and VE, our goal is to use a general representation of environment geometry. While environments typically consist of 3D objects, we only consider the 2D projections of the environment onto the plane. To prevent an environment from being infinitely large, we represent an environment as a closed polygon $\mathcal{P}$, and obstacles

in the environment as holes of $\mathcal{P}$. We use many standard geometric concepts to represent the environment geometry and a user's position and orientation in an environment. Throughout this chapter, a subscript of $phys$ or $virt$ on the symbols below is used to clarify if the symbol belongs to the physical or virtual environment, respectively.

- $\mathcal{P}$: A closed polygon, specified as an ordered set of vertices $\{v_1, v_2, ..., v_m\}$, where consecutive vertices are connected with an edge. If $\mathcal{P}$ has holes, they are specified using simple polygons, which are also represented as ordered sets of vertices connected by edges.
- E: An environment, specified as a closed polygon, potentially with holes representing obstacles in the environment.
- p: A location in an environment, specified as a vector in $\mathbb{R}^2$.
- θ: A user's orientation in an environment, in the range $[0, 2\pi)$.
- q: A user's configuration (or state) in an environment. Their configuration consists of a position p and an orientation θ.
- $\mathcal{C}_{obs}$: The set of all states that correspond to the user as having collided with an obstacle in E (also known as the obstacle space).
- $\mathcal{C}_{free}$: The set of all states that correspond to the user not collided with an obstacle in E (also known as the free space). This set corresponds with the set of points in E that are not inside any holes (obstacles) of E. This represents the regions of E that the user can walk in.
- $Free_{phys}$, $Free_{virt}$: The free space in the physical or virtual environment, respectively.

6.3.2 User Position and Orientation

Since visual perception plays a large role in driving a person's locomotion experience [145], our goal is to define a metric using a geometric representation that is congruent with what users see during navigation in an environment. As such, we use the visibility polygon as a representation of the local surroundings that a user sees at a single time instance during navigation. For a given point p in the plane, the visibility polygon $\mathcal{P}$ is the set of all points in the plane that are visible from p. The point from which the visibility polygon is computed is also known as the kernel, k. For an environment E and any point $p \in \mathcal{C}_{free}$, we know that $\mathcal{P} \subset \mathcal{C}_{free}$ by the definitions of $\mathcal{P}$ and $\mathcal{C}_{free}$. For an environment with a set of obstacles $\mathcal{O}$, we have a set S of line segments that denotes the boundaries of all obstacles in the environment. The visibility polygon can be computed in $O(s \log s)$ time [199], where $s = |S|$, which makes it a fairly efficient representation of the user's local surroundings.

While the visibility polygon allows us to represent the user's surroundings at a single moment during locomotion, a single visibility polygon does not account for the different positions and orientations a user could have across an entire path. An environment has many different positions at which the user can be located, and the local surroundings could be different for every unique position. In the following subsections, we explain how we extend the notion of visibility polygons to account for the many different positions and orientations a user can have during locomotion, thus providing a way to summarize the *entire* environment using *local* features which are most prominent during locomotion (i.e. visibility polygons).

6.3.2.1 Positions

In order to represent the user's position relative to local obstacles *and* account for the many different possible positions the user can be located at, we uniformly sample points in $\mathcal{C}_{free}$ and compute a visibility polygon at each sampled point. For an environment represented by a polygon $\mathcal{P}$ (potentially with holes), $\mathcal{C}_{free}$ is represented as the set of points in $\mathcal{P}$ that are not in any holes of $\mathcal{P}$.

First, we compute a conforming constrained Delaunay triangulation [170] of $\mathcal{C}_{free}$ with the constraint that each triangle has a maximum area, which is a free parameter that can be adjusted.[1] Once the triangulation is computed, we choose the sampled points as the set vertices of the triangulation that lie in the free space (see Figure 6.2). We denote this set of sampled points as $P = \{p | p \in \mathcal{C}_{free} \text{ and } p \in \text{CDT}\}$, where CDT is the conforming constrained Delaunay triangulation of E. Let n be the number of points in P. For each point $p \in P$, we compute a visibility polygon at p (that is, the kernel k of the visibility polygon is p). The final output of our uniform sampling is a set of n visibility polygons $\mathfrak{P} = \{\mathcal{P}_1, \mathcal{P}_2, ..., \mathcal{P}_n\}$. In this manner, $\mathfrak{P}$ is an approximation of all the different local surroundings that the user can have in E along any given path.

We use a conforming constrained Delaunay triangulation to sample $\mathcal{C}_{free}$ since it tends to produce fairly uniformly-sized triangulations, which means the points we sample from the triangulation tend to be evenly spaced. Other sampling methods like random sampling or importance sampling can be used, but care must be taken to ensure that the sampling scheme yields points that are evenly spread across the environment *without* producing too many points, since this would slow down the metric computation process (see Subsection 6.3.5). In this work, we make no

[1] The implementation used is available here: `rufat.be/triangle`

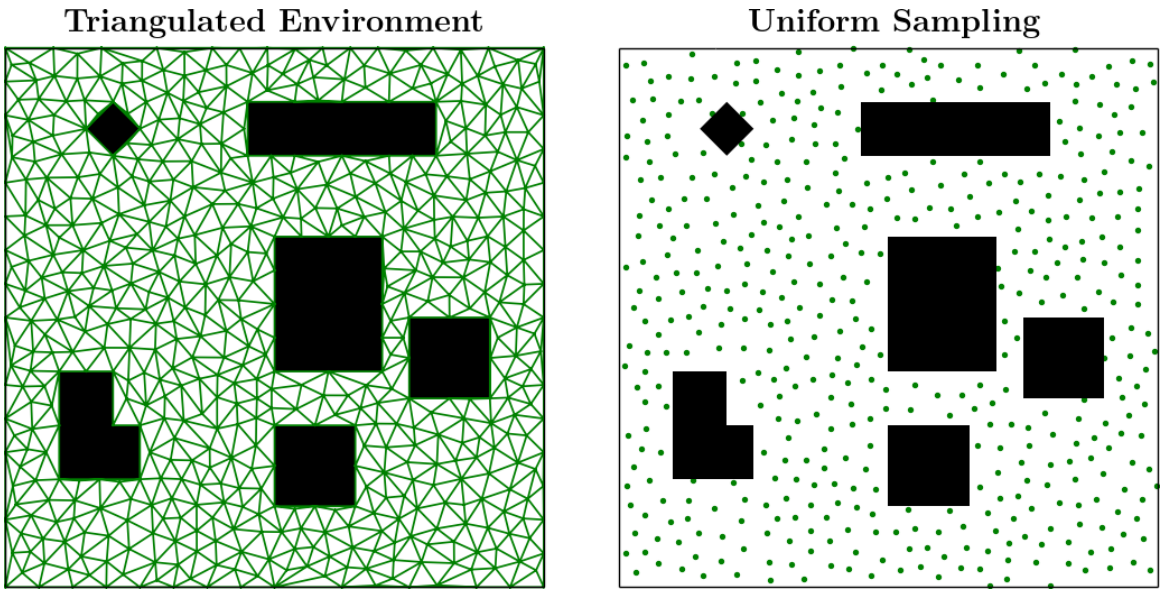


Figure 6.2: *Left:* An environment with obstacles (black) and the constrained Delaunay triangulation (green) of the free space. *Right:* The vertices (green) of the constrained Delaunay triangulation that lie inside the free space. These vertices are the sampled points at which we compute visibility polygons to describe the structure of the environment and compute our ENI metric.

assumptions about which regions of $\mathcal{C}_{free}$ the user may be located in, so a uniform sampling is best suited for our metric computation.

6.3.2.2 Orientations

While the visibility polygons in $\mathfrak{P}$ represent the different local surroundings the user can perceive as they change their position in E, these polygons do not account for the fact that the user's perception of their surroundings will also depend on their orientation in the environment. To account for the user's orientation at a position in E, we rotate the visibility polygon around its kernel (since the kernel also corresponds to the user's position in E). For a visibility polygon $\mathcal{P}$ with kernel k and vertices $\{v_1, v_2, ..., v_m\}$, we define the visibility polygon rotated counterclockwise

by θ radians as:

$$\mathcal{P}^{\theta} = \left\{ \left(\begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} v.x - k.x \\ v.y - k.y \end{bmatrix} \right) + \begin{bmatrix} k.x \\ k.y \end{bmatrix} \middle| \; v \in \mathcal{P} \right\} \tag{6.3.2.1}$$

Here, $v.x$ and $v.y$ represent the x- and y-coordinates of the vertex v, respectively. Thus, $\mathcal{P}^{\theta}$ has the same shape as that of $\mathcal{P}$, with the only difference being the orientation of this polygon in the plane.

6.3.3 Measuring Compatibility of Local Surroundings

The more similar the physical and virtual visibility polygons are, the more likely it is that the user is able to walk on collision-free paths in the local neighborhood of the current location (see Figure 6.3). Thus, to measure the compatibility of a user's physical and virtual surroundings, we need a way to measure the similarity of two visibility polygons. This provides a way to assess the navigability of a user's configuration in the PE and VE.

As mentioned in Subsection 6.2.2, there are many different ways to measure the similarity of shapes. In this work, we are mainly concerned with the user's proximity to obstacles. Since the visibility polygon already encodes the proximity to obstacles in all directions, our notion of shape similarity depends on the sizes of the polygons. Thus, to measure the similarity (compatibility) of two visibility polygons, we measure the amount of overlapping area between the polygons.

Given a physical position $p_{phys} \in Free_{phys}$ and a virtual position $p_{virt} \in Free_{virt}$, let $\mathcal{P}_{phys}$ and $\mathcal{P}_{virt}$ be the visibility polygons with kernels $k_{phys} = p_{phys}$ and $k_{virt} = p_{virt}$, respectively. We define the similarity of $\mathcal{P}_{phys}$ and $\mathcal{P}_{virt}$ as the area of $\mathcal{P}_{virt}$ that is "inaccessible" from $\mathcal{P}_{phys}$. That is, if the user is standing at k_{phys} and k_{virt}, our similarity metric for two visibility polygons is the total area of $\mathcal{P}_{virt}$ that cannot be reached due to occlusion by an edge of the boundary of $\mathcal{P}_{phys}$

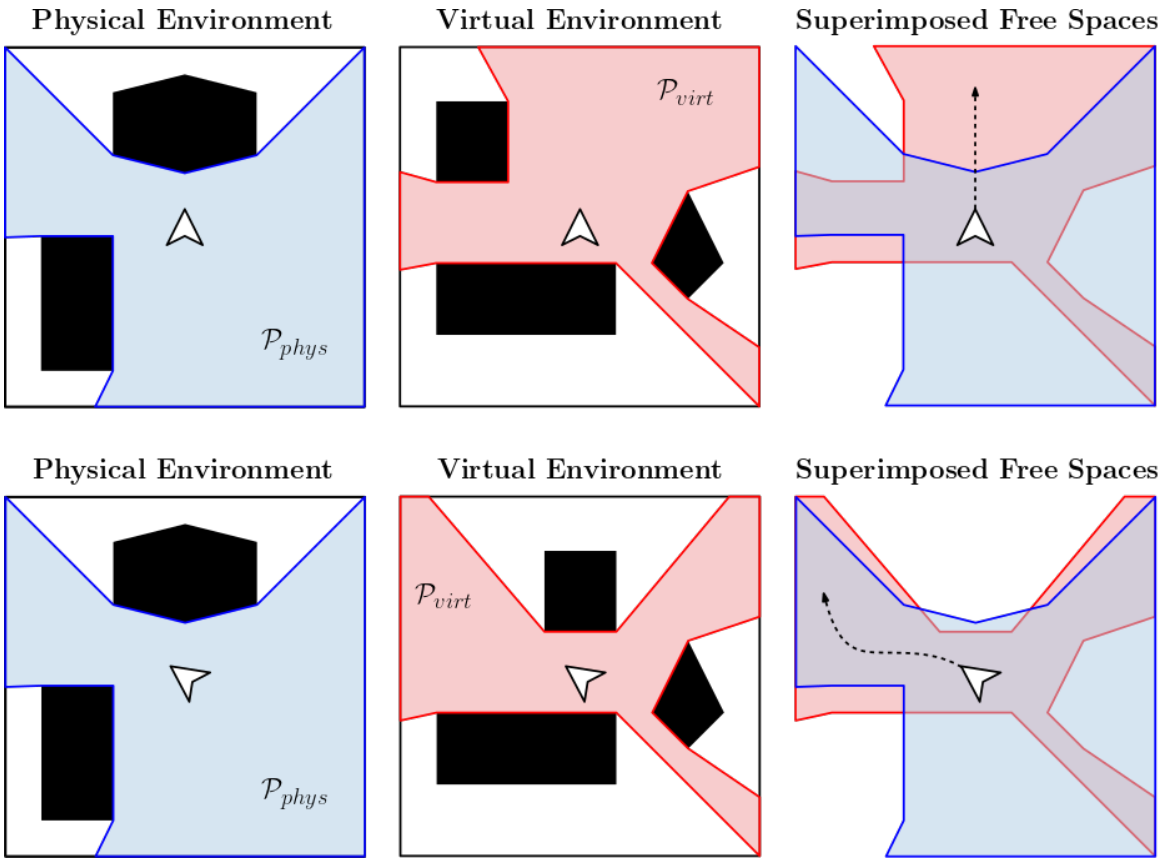


Figure 6.3: An illustration of the impact of the similarity between the user's physical and virtual surroundings on their ability to travel on collision-free paths. In the top row, the user (shown as the white cursor) cannot walk forward in the VE without colliding with an object in the PE. In the bottom row, the user's proximity to obstacles in the two environments is more similar, so more of the possible paths in the VE correspond to collision-free paths in the PE. In our metric, we compute this area of the virtual surroundings that cannot be accessed from a particular physical surrounding as a measure of the navigability at a pair of physical and virtual configurations.

(see Figure 6.3). Formally, the similarity of visibility polygons $\mathcal{P}_1$ and $\mathcal{P}_2$ is computed using the Boolean difference operation [49]:

$$\phi\left(\mathcal{P}_1, \mathcal{P}_2\right) = \texttt{area}\left(\mathcal{P}_1 \setminus \mathcal{P}_2\right). \tag{6.3.3.1}$$

Here, the `area(`x`)` function returns the total area of the set of polygons x. Note that $\mathcal{P}_1$ and $\mathcal{P}_2$ must be translated such that their kernels lie at the same coordinates in the plane (i.e. $\mathcal{P}_1$ and $\mathcal{P}_2$ must be "overlayed" ontop of each other). A visualization of this similarity metric is shown in

Figure 6.4. If $\phi(\mathcal{P}_1, \mathcal{P}_2) = 0$, this implies that $\mathcal{P}_1$ is entirely contained inside $\mathcal{P}_2$, and that the user can reach all regions of $\mathcal{P}_1$ without colliding with any obstacles represented by the edges of $\mathcal{P}_2$. The larger $\phi(\mathcal{P}_1, \mathcal{P}_2)$ is, the more dissimilar $\mathcal{P}_1$ and $\mathcal{P}_2$ are, which increases the amount of space in $\mathcal{P}_1$ that is inaccessible when the user is standing at $k_2 \in \mathcal{P}_2$. Note that ϕ is *not* symmetric, so $\phi(\mathcal{P}_1, \mathcal{P}_2) \neq \phi(\mathcal{P}_2, \mathcal{P}_1)$. In our formulation, we only compute $\phi(\mathcal{P}_{virt}, \mathcal{P}_{phys})$, since our primary concern is regions of the VE that are inaccessible due to constraints imposed by the PE.

It should be noted that area is only one measure of the similarity of $\mathcal{P}_{phys}$ and $\mathcal{P}_{virt}$. It is also possible to use other properties of $\mathcal{P}$ as the basis of our comparison, such as the average distance between the boundary of $\mathcal{P}$ and k, or the length of the shortest line connecting two points on the boundary of $\mathcal{P}$ that also passes through k. In our benchmarks, we observed that area was an overall better metric because it provides a holistic summary of the differences between $\mathcal{P}_1$ and $\mathcal{P}_2$, while other metrics tended to ignore regions of either polygon.

6.3.4 ENI Metric

In this section, we describe how various components described above are used to compute our Environment Navigation Incompatibility (ENI) metric. Our goal is to estimate, for any possible virtual state $q_{virt} \in Free_{virt}$, which physical state $q_{phys} \in Free_{phys}$ is most compatible with q_{virt}. If we can compute this for all states in $Free_{virt}$, we will have a measure that tells us how easy, in the ideal case, it will be for a user to navigate on a collision-free path in the given ⟨PE, VE⟩ pair. Our metric requires as input the 2D layouts of a physical environment E_{phys} and a virtual environment E_{virt} (Subsection 6.3.1). Given the layouts of E_{phys} and E_{virt}, we sample points uniformly across each environment using the technique detailed in subsubsection 6.3.2.1, with a maximum area constraint of $0.1m$. For each sampled point $p \in E$, we compute the

associated visibility polygon with kernel p.

This yields two sets of visibility polygons, $\mathfrak{P}_{phys}$ and $\mathfrak{P}_{virt}$, which are an approximation of all the possible states that the user can have in E_{phys} or E_{virt}. For each state $q_{virt} \in Free_{virt}$ (i.e. each visibility polygon $\mathcal{P}_{virt} \in \mathfrak{P}_{virt}$), we wish to find the physical state $q_{phys} \in Free_{phys}$ that is most similar to q_{virt}. To do this, we compare each $\mathcal{P}_{phys} \in \mathfrak{P}_{phys}$ to each $\mathcal{P}_{virt} \in \mathfrak{P}_{virt}$ using the similarity metric in Equation 6.3.3.1. However, it is not enough to compute $\phi(\mathcal{P}_{virt}, \mathcal{P}_{phys})$ with the polygons $\mathcal{P}_{phys}$ and $\mathcal{P}_{virt}$. The polygons computed from our uniform sampling represent the different positions the user can have, but they do not account for the user's orientation. To measure the similarity of $\mathcal{P}_{phys}$ and $\mathcal{P}_{virt}$ while also accounting for the different orientations the user can have, we aim to solve the following optimization problem:

$$\Phi^* \left(\mathcal{P}_{virt}, \mathcal{P}_{phys}\right) = \min_{\theta \in [0, 2\pi)} \phi \left(\mathcal{P}_{virt}, \mathcal{P}^{\theta}_{phys}\right). \tag{6.3.4.1}$$

That is, we want to find the $\theta \in [0, 2\pi)$ that minimizes the value of $\phi(\mathcal{P}_{virt}, \mathcal{P}^{\theta}_{phys})$ (i.e. maximizes the similarity between $\mathcal{P}_{virt}$ and $\mathcal{P}_{phys}$). In practice, we found that computing $\Phi^* \left(\mathcal{P}_{virt}, \mathcal{P}_{phys}\right)$ for every pair of visibility polygons in $\{\mathfrak{P}_{virt} \times \mathfrak{P}_{phys}\}$ was too expensive. To lower the computation time, we limit Φ^* to optimize ϕ in the domain $\Theta_{\Delta} = \{\theta_1, \theta_2, ...\theta_{10}\}$, where $\theta_1 = 0^\circ$ and θ increases in increments of 36°. Thus, for a pair of physical and virtual visibility polygons $\mathcal{P}_{phys}$ and $\mathcal{P}_{virt}$, we approximate the maximum similarity of the polygons as:

$$\Phi \left(\mathcal{P}_{virt}, \mathcal{P}_{phys}\right) = \min_{\theta \in \Theta_{\Delta}} \phi \left(\mathcal{P}_{virt}, \mathcal{P}^{\theta}_{phys}\right). \tag{6.3.4.2}$$

Now we have everything necessary to approximate the optimal q_{phys} for a given q_{virt}. Using visibility polygons to represent q_{phys} and q_{virt}, we compute the q_{phys} that is most compatible with

q_{virt} as:

$$\mathcal{P}^*_{phys} = \underset{\mathcal{P}_{phys} \in \mathfrak{P}_{phys}}{\arg\min} \; \Phi\left(\mathcal{P}_{virt}, \mathcal{P}_{phys}\right). \tag{6.3.4.3}$$

To compute the ENI metric, we compute the compatibility score between each virtual visibility polygon and its most compatible physical visibility polygon, yielding a vector of real numbers representing the best-case compatibility for each sampled state in $Free_{virt}$. Formally, this is defined as:

$$\mathbf{x} = \left\{ \Phi\left(\mathcal{P}_{virt}, \underset{\mathcal{P}_{phys} \in \mathfrak{P}_{phys}}{\arg\min} \; \Phi\left(\mathcal{P}_{virt}, \mathcal{P}_{phys}\right)\right) \middle| \mathcal{P}_{virt} \in \mathfrak{P}_{virt} \right\}, \tag{6.3.4.4}$$

where $\mathbf{x}$ is an n-dimensional vector and $n = |\mathfrak{P}_{virt}|$. This vector $\mathbf{x}$ is the final output of the ENI metric. Since n can be in the thousands, we summarize the output of ENI using the mean and standard deviation of the vector, which we denote as $[\mu, \sigma]$. Note that this summary does not perfectly characterize the ENI measure, since two distinct measures can have the same mean and standard deviation. Details on how to accurately interpret $\mathbf{x}$ are discussed in Subsection 6.4.1.

6.3.5 ENI Metric: Analysis

In this section, we discuss the properties of our Environment Navigation Incompatibility metric. These properties help ensure that ENI avoids ambiguity and accurately models the important features of the VR locomotion problem during computation.

Sensitivity to input: Assuming the point sampling parameters are fixed (i.e. for a fixed input, computing the sampled points multiple times yields the same set of points each time), the output of the ENI metric is always the same. This is because our metric performs an exhaustive search of all pairs of physical and virtual states $\{\mathfrak{P}_{phys} \times \mathfrak{P}_{virt}\}$ when computing the compatibility of the PE and VE. As a result of this property, we avoid ambiguity that can arise from using

environment properties that do not completely characterize the layout of the environment (e.g. environment area). Note that since we compute a triangulation of each environment, small perturbations in the input geometry will result in different triangulations (and sampled points), which will yield slightly different metric measurements.

Coupled computation: The ENI metric requires a ⟨PE, VE⟩ *pair* as input in order to be computable. This is because the metric is designed to compute the compatibility of the two environments. ENI was intentionally designed in this way since it is the differences between the PE and VE that make collision-free navigation difficult. This property ensures that our metric appropriately considers the layouts of the PE and VE *relative to each other*, which makes ENI a more faithful measure of navigability in VR.

Sampling density: The ENI metric has one free parameter, which is the maximum area of triangles in the constrained Delaunay triangulation of $\mathcal{C}_{free}$ that is used to uniformly sample points in E (subsubsection 6.3.2.1). The smaller this parameter is, the denser the sampling of points in E. A denser sampling yields a more more accurate measure for the ENI metric, but also increases the size of the output $\mathbf{x}$ and the computation time. In our implementation, we set the maximum area such that each environment has roughly 500 samples (*i.e.*, the maximum area parameter depends on the area of $\mathcal{C}_{free}$).

To validate that 500 samples was sufficient, we computed the ENI measure for a ⟨PE, VE⟩ pair with varying amounts of sample density, and compared the changes in the mean and standard deviation of the ENI measures. The results are shown in Table 6.1. From these results, we can see that increasing the number of points more than tenfold does not yield a noticeable change in the μ or σ of the ENI metric, suggesting that our 500 samples points is sufficiently high resolution. Furthermore, increasing the sample density leads to prohibitively high computation times for

relatively little increase in metric accuracy, since the runtime complexity of the ENI metric is $O(nm)$, where $n = |\mathfrak{P}_{virt}|$ and $m = |\mathfrak{P}_{phys}|$.

Sampled points	ENI μ	ENI σ	Computation time (s)
257	486.9	92.3	168
517	486.7	93.4	347
1050	482.3	98.5	664
2129	482.3	100.0	1277
4337	481.8	100.1	2593

Table 6.1: Effect of sample density on the accuracy of the ENI metric, using the ⟨PE #1, VE #1⟩ environment pair. After increasing the density of our point sampling by roughly $16\times$, the mean and standard deviation of the ENI metric exhibited very little change in values, but suffered a significantly greater computation time.

6.4 Applications and Benefits of ENI

6.4.1 Analyzing Areas with Low and High Compatibility

Since the output of ENI is an n-dimensional vector of real numbers, it is difficult to directly interpret the ENI measure. To aid in interpretation, we visualize the ENI measure using an interactive visualization which can be seen in Figure 6.5. Our visualization includes a map of the PE and VE, and a histogram of the individual compatibility scores computed for each pair of physical and virtual visibility polygons (see Subsection 6.3.4). Each circle drawn in E_{virt} represents the kernel of a virtual visibility polygon. A circle's color is determined by the compatibility score Φ computed from Equation 6.3.4.2 using the visibility polygon centered at that circle's location. By taking a dense, uniform sampling of points and coloring them according to their compatibility scores, our visualization provides an easy way to see which regions of the VE lead to the most incompatibility with respect to the given PE.

To further improve the interpretability of the ENI metric, our visualization also includes interactive tools that allow researchers to explore the metric output. With a lasso tool (Figure 6.6),

users can select points in E_{virt} which will also highlight the corresponding most compatible points in E_{phys} that were computed from Equation 6.3.4.3. This helps researchers to understand, for a selected group of configurations in the VE, which configurations in the PE will be most amenable to collision-free navigation. Additionally, hovering over any bar in the histogram will highlight the physical and virtual points that contributed to the selected histogram bar. Using this interactive visualization, researchers can explore the compatibility of a pair of environments on both a broad and a specific level, which makes it easier to design improved RDW controllers and more compatible ⟨PE, VE⟩ pairs.

6.4.2 Analysis of Changes in VE on Compatibility

The ENI metric helps us to understand how different changes to the VE can effect the ease of collision-free navigation in the given PE. We assume that the PE is fixed, since this is usually true in practice. To demonstrate how ENI can be used to understand the effects of VE changes on navigability, we show the results of two examples. First, we look at the effects of changing the density of objects in the VE on navigability, and second we look at the effects of changing the size of the VE on navigability.

6.4.2.1 Changes in Virtual Object Density

For this example, we consider the PE and VEs shown in Figure 6.7. The PE is designed to represent a room that could be found in a home, such as a living room. The VEs are chosen to have the same hexagonal boundary shape with an area of $900m^2$, but with varying amounts of random polygonal objects. The first environment has four objects, the second has eight objects, and the third environment has sixteen objects. Objects that are present in one VE are still present in the VEs with higher object density, to make comparisons between conditions easier.

Results of the ENI metric measurements for each of the ⟨PE, VE⟩ pairs are shown in Figure 6.7. As the VE is populated with more obstacles, the average and maximum ENI scores decrease, indicating that the more cluttered VEs are more compatible with the given PE for navigation. Intuitively, this makes sense since the amount of space in the VE that is visible (and thus, immediately navigable) goes down as the object density increases. As the area of immediately navigable space in the VE decreases, it approaches the area of immediately navigable space in the much smaller PE. Results from the validation of the ENI metric (see Subsection 6.5.1) confirm that navigability increases as the density of objects in the VE increases. From this example, we can see that introducing obstacles into an environment can actually make it *easier* for users to avoid collisions in the PE, if the PE also has some obstacles that may obstruct the user's path.

6.4.2.2 Changes in Virtual Environment Size

When investigating the effects of the size of the VE on navigability, we considered the PE and VEs shown in Figure 6.1. Similar to the object density experiment, we chose a fixed PE that could be found in a home (via [232]), and constructed three VEs of varying sizes ($400m^2$, $900m^2$, and $1600m^2$), each with different boundary shapes and ten different, random polygonal objects.

We see that the mean and maximum ENI values increase as the area of the VE increases. Intuitively, this is expected since VEs with larger area are more likely to lead to longer paths that cannot be traversed from within the small PE. When the navigable area of the VE decreases, users are more likely to travel on virtual paths with shorter segments, since they will be forced to make more turns to avoid objects in the VE. These turns increase the chance that the user will also turn away from nearby objects in the PE and incur fewer collisions, as the ENI scores

suggest. This result is also confirmed by the validation experiment in Subsection 6.5.1, in which we estimate the navigability of the ⟨PE, VE⟩ pairs in Figure 6.1 via simulated paths. Thus, from this experiment, it is clear that increasing the area of the navigable space in the VE leads to more collisions in VR locomotion.

6.4.3 Design Guidelines Based on ENI

Although ENI helps us understand how the layout of the environment influence navigability of a ⟨PE, VE⟩ pair, it is not always straightforward to translate an ENI metric into a more compatible ⟨PE, VE⟩ pair. In this section, we provide some high-level guidelines on how to design VEs that are more amenable for VR locomotion, relative to a given PE. We note that these are not strict rules for designing virtual environments, but rather are intended to be suggestions on how to create environments that are more likely to yield a more comfortable navigation experience for users.

In general, collision-free locomotion in VR is most difficult when the VE contains large, open spaces while the PE is small and cluttered with objects. This is an undesirable situation because in these cases, it is more likely that the user will travel on a long, straight path in the VE which cannot be traversed in the PE due to obstacles. To avoid this, designers of VEs should try to place objects in the VE such that the navigable area in the VE is reduced (e.g. subsubsection 6.4.2.1). This forces users to travel on virtual paths with more turns as they avoid virtual objects–these turns make it more likely that the user will also turn away from obstacles in the PE. Virtual structures like narrow corridors are favorable since they restrict the number of possible paths that the user can travel along, which decreases the chance that their particular path yields a collision in the PE.

In addition to reducing the frequency of large, open spaces, designers may want to place objects in the VE that have a similar size, shape, and distribution to objects in the PE if possible. This has the effect of making the PE and VE more similar on a local scale (*i.e.* the visibility polygons are more similar), which further increases the chance that the user can travel along collision-free paths (see Figure 6.3).

6.4.4 Performance Analysis of RDW Controllers

Redirection controllers are algorithms that are used to compute the correct redirection that is applied to optimally steer the user away from physical objects [140]. The performance of controllers is typically evaluated by measuring the number of times the user is reset after getting too close to a physical object [11, 132, 205, 231, 232], the average distance the user is able to walk before initiating a reset [11, 192, 205, 231], or the average strength of redirection during locomotion [11, 119, 231]. These performance metrics are useful, but they do not provide any information in terms of how similar is the user's proximity to obstacles in the PE and VE. By calculating the ENI measure between corresponding physical and virtual configurations along a given path through both environments, we can gain an understanding of the differences in the user's physical and virtual proximity to objects. This kind of performance metric provides information about the user's locomotion experience (for a given RDW controller) that cannot be derived from traditional performance metrics such as number of resets or intensity of redirection.

In Figure 6.8, we show an example result of computing the ENI of a user's configurations across a path. In this example, the user is located in an identical $10m \times 10m$ $\langle$PE, VE$\rangle$ pair with no obstacles. We simulated the user walking on the same virtual path while being steered with three different RDW controllers: artificial potential fields (APF) [205], alignment-based

redirection controller (ARC) [231], and steer-to-center (S2C) [84]. For the path segment shown in Figure 6.8, the ENI measures were $[\mu = 34.387, \sigma = 11.680]$ for APF, $[\mu = 11.468, \sigma = 8.513]$ for ARC, and $[\mu = 34.177, \sigma = 9.532]$ for S2C. This result suggest that when steered by ARC, the user is, on average, in a physical configuration that is more compatible with their virtual configuration than when they are steered with APF or S2C. That is, for most configurations along the path, the user is more likely to be able to travel on a collision-free path when steered by ARC than by APF or S2C. In the short path segment we considered in this example, this is indeed the case as the user incurs zero collisions when steered by ARC, but incurs two resets when steered by APF and S2C. ARC was designed to steer the user in an attempt to match their physical and virtual proximity [231], so our metric's performance matches the expected behavior of ARC. This conclusion is further supported by comparing the shape of the virtual path to each of the physical paths. It is clear from Figure 6.8 that the user's physical path when steered by ARC is more similar to the virtual path than the paths generated by APF and S2C algorithms.

6.5 User Studies and Validation

To validate the ENI metric, we collected navigation data from simulations and two user studies. The goal of the ENI metric is to provide insight into the relationship between environment layouts and ease of collision-free locomotion in VR. Therefore, it is expected that our metric is correlated with the navigation behavior of users or provides new insights in terms of designing $\langle$PE, VE$\rangle$ pairs. To this end, we show that our metric correctly identifies pairs of physical and virtual configurations that allow for easier collision-free navigation (Subsection 6.5.1) and that ENI is correlated with users' tendency to avoid physical objects during locomotion in VR (Subsection 6.5.2).

6.5.1 Experiment 1: Simulation Experiment

6.5.1.1 Design

In this experiment, we simulated a user walking along 50 paths in different pairs of physical and virtual environments. Specifically, we tested the six environment pairs examined in Subsection 6.4.2 and the three pairs used in [231]. In each environment, we simulated the user traveling along 50 different paths with a random start and end configuration in the VE, and a random starting configuration in the PE. Paths through the VE were generated using the RRT* algorithm [102] due to its efficiency and ability to guarantee complete paths. For each path, the simulated user would traverse the path in the VE and the PE simultaneously. If the user got too close to an object in the PE, a reset maneuver was initiated such that they were reoriented away from the nearby object. Users were reset with the reset-to-gradient technique presented in [205] due to its ability to work in a variety of different environments. To quantify the navigability of the environments in accordance with the definition of navigability we adopt in this chapter (see Subsection 6.2.1), we computed the average distance travelled before a reset was incurred across all 50 paths in each ⟨PE, VE⟩ pair.

6.5.1.2 Results

The results of the average distance walked between resets is shown in Table 6.2. In general, we see that as the ENI decreases (i.e. navigability improves), the average distance walked between resets increases, indicating that the simulated user was able to travel further before being interrupted by a reset. This trend in the results confirms that ENI is correctly able to identify ⟨PE, VE⟩ pairs that are better or worse for natural walking in VR.

Although the high-level trends showed that lower ENI is associated with increased navigability,

there are some interesting results in the data. First, we see that there is only a small difference in average distance walked between the ⟨PE #1, VE #2⟩-⟨PE #1, VE #3⟩ and ⟨PE #2, VE #4⟩-⟨PE #2, VE #5⟩ environment pairs, despite the large differences in their ENI scores. While the cause of this is not clear, we believe that this plateauing effect in the distance walked might be an indication of the lower bounds on navigability for a given PE. That is, beyond a certain ENI score, the navigability does not get significantly worse as the ENI increases because the difficulty in navigation becomes maximally constrained by the layout of the PE.

Finally, we see that the ⟨PE #4, VE #8⟩ pair yields the worst navigability, despite having a low ENI score. Though initially surprising, this result makes sense when we consider the actual shape of PE #4 (see Figure 6.12). Although the PE and VE have similar local structure, the PE consists only of narrow corridors, which are inherently difficult to navigate without getting too close to any obstacles. In this case, small deviations from a path that travels directly down a corridor are likely to lead to resets. Additionally, the reset-to-gradient maneuver reorients users such that they face directly away from the object they got too close to. In this particular environment, this means that the user often faces the opposite wall of the corridor, which they then walk directly towards after finishing the reset. Thus, the simulated user gets stuck, oscillating back and forth between the walls of the corridor. This result highlights one shortcoming of the ENI metric: by only considering the geometry of the environments, and *not* considering the motion behavior of the user through the environments, some aspects of the environment structure that are important for navigability cannot be properly evaluated using our metric.

	Distance Between Resets		ENI score	
⟨PE, VE⟩ pair	μ (m)	σ (m)	μ	σ
⟨PE #1, VE #1⟩	3.337	1.869	80.297	32.020
⟨PE #1, VE #2⟩	3.082	1.802	241.510	56.741
⟨PE #1, VE #3⟩	2.991	1.701	760.146	245.238
⟨PE #2, VE #4⟩	3.031	2.382	485.231	96.049
⟨PE #2, VE #5⟩	3.065	2.515	345.408	112.748
⟨PE #2, VE #6⟩	3.505	2.347	206.276	75.778
⟨PE #3, VE #7⟩	6.075	2.835	0.530	0.251
⟨PE #4, VE #8⟩	0.989	0.878	9.101	4.836
⟨PE #5, VE #9⟩	3.129	1.993	78.357	33.292

Table 6.2: Navigability results from simulating 50 random walking paths in different ⟨PE, VE⟩ pairs. Here, we define navigability as the average distance that the user can walk in the VE before colliding with an object in the PE, across all configurations in the PE and VE (Subsection 6.2.1). In general, the navigability of the environments decreases as the ENI score increases, indicating that our metric is able to correctly identify ⟨PE, VE⟩ pairs that are less amenable to real walking.

6.5.2 Experiment 2: User Studies

In our second experiment, we hypothesized that ⟨PE, VE⟩ pairs with higher ENI scores will cause users to incur more resets than ⟨PE, VE⟩ pairs with low ENI scores.

6.5.2.1 Design

We conducted two user studies. In both studies, participants were tasked with reaching a goal in the VE, indicated by a floating yellow block. In the first study, participants navigated through a VE with no obstacles, while the layout of the PE changed to have increasing density of objects. Participants completed the walking task a total of three times for each ⟨PE, VE⟩ pair, with the goal location being different in each of the three trials (see Figure 6.10). The dimensions of the PE and VE were the same ($4.37m \times 6.125m$). Twenty people participated in the first study (8 female (age $\mu = 23, \sigma = 2.6$), 11 male (age $\mu = 23.4, \sigma = 2.5$), 1 non-binary (age 25)). Participants completed the task in blocks organized by PE (all three tasks in one PE were

completed before changing to the next PE), and the order of PEs was counterbalanced. After completing the experiment, all participants completed the Kennedy-Lane Simulator Sickness Questionnaire (SSQ) [104]; the largest reported SSQ score was 33.66 ($\mu = 8.727, \sigma = 9.494$). The first study took about 20 minutes for each participant, and they were compensated with a $\$10$ Amazon gift card.

In the second study, participants navigated through three different VEs, each with a varying size (similar to Figure 6.1). The PE was the same for each of the three VEs. Participants completed a total of three walking trials, each experiencing a random goal location in each trial (the same for each participant). The order in which participants experienced the ⟨PE, VE⟩ pairs was the same across participants. A total of 10 people participated in the second study (9 male (age $\mu = 25, \sigma = 3.5$), 1 female (age 25)). Participants completed the navigation task once in three different VEs, in the same order across all participants. After completing the experiment, all participants completed the Kennedy-Lane Simulator Sickness Questionnaire (SSQ) [104]; the largest reported SSQ score was 29.92 ($\mu = 7.106, \sigma = 10.043$). The second study took about 15 minutes for each participant, and they were compensated with a $\$10$ Amazon gift card.

During our user studies, if the user got within $0.25m$ of an obstacle, a reset was initiated and they were instructed to turn $180°$ in the PE, while the virtual camera did not rotate. To increase the variety in the traveled paths, each trial started with the user facing a random direction in the VE (participants all experienced the same random direction for a particular trial, but the directions were random from trial to trial). Participants also completed a practice trial at the start of the experiment to get them accustomed to the hardware and experiment task. Both studies were approved by the authors' university's Institutional Review Board.

6.5.2.2 Results

The results of the two user studies can be seen in Table 6.3. We validate our metric by measuring the navigability (average distance walked between resets) as the ENI changes. Results show that as the ENI increases (i.e., ⟨PE, VE⟩ compatibility decreases), users walked less distance between resets (i.e. navigability decreased). This confirms that our ENI metric is correctly able to identify ⟨PE, VE⟩ pairs which have comparatively more or less navigability.

Interestingly, the trends in the results from the user studies do not show the same plateauing effect as the results from Subsection 6.5.1. In the simulation experiments, our results show that two ⟨PE, VE⟩ pairs can have very large differences in ENI scores, but can yield very similar distances walked between resets. While we are not certain why the plateauing effect did not appear in the user study results, we believe it may be due to the lower total number of paths collected, or due to the smaller differences in ENI scores between ⟨PE, VE⟩ pairs, compared to the differences in the simulation experiments. Future work should study this plateauing effect in more detail.

	Distance Between Resets		**ENI score**	
⟨PE, VE⟩ pair	$\mu\ (m)$	$\sigma\ (m)$	μ	σ
⟨PE #6, VE #10⟩	3.217	1.398	0.530	0.251
⟨PE #7, VE #10⟩	2.927	1.548	8.072	1.013
⟨PE #8, VE #10⟩	2.390	1.153	14.097	0.612
⟨PE #9, VE #11⟩	4.098	2.168	27.131	8.864
⟨PE #9, VE #12⟩	3.688	2.604	102.129	24.390
⟨PE #9, VE #13⟩	3.559	2.302	217.361	48.958

Table 6.3: Navigability results from two separate user studies. In the first user study (first three rows), users walked towards a goal location in a static VE while located in three different PEs. In the second study (bottom three rows), users searched for a goal location in different VEs while located in the same PE. In both situations, our results showed that navigability decreases as the ENI score increases, validating the correctness of our metric.

6.6 Conclusion, Limitations, & Future Work

In this work, we presented Environment Navigation Incompatibility (ENI) metric, a novel metric for quantifying the navigability of a pair of physical and virtual environments, based on their geometric layouts. ENI measures the navigability of a ⟨PE, VE⟩ pair by measuring the similarity (compatibility) of the two environments, since collisions during locomotion in VR are mainly caused by differences in the layouts of the PE and VE. By uniformly sampling the environments and computing visibility polygons at sampled points, our metric accurately captures the features of the environments that are amenable to collision-free navigation (namely, the local surroundings of a user across the PE and VE). We validate our metric through simulations and two user studies, showing that ENI can accurately identify ⟨PE, VE⟩ pairs that are more amenable to collision-free navigation *without* requiring locomotion data. In general, users were able to walk further before incurring a reset in environment pairs that our metric identified as more navigable.

Although the ENI metric is effective at identifying compatible ⟨PE, VE⟩ pairs for RDW, it has some limitations. First, the computation time can be long if the environments have a large number of sampled points or have a high number of obstacles. Our implementation was done in Python and was not parallelized, so there is room for significant speed-up by porting the implementation to a faster language and by taking advantage of multithreading for visibility polygon computation. Another limitation of ENI is that it is currently limited to static environments. Extensions of ENI to environments with dynamic obstacles will likely require adding a temporal component to the computation, which may increase the computation time even further. Additionally, ENI provides a "best case" mapping of virtual configurations to physical configurations. This best case mapping can be misleading for assessing navigability in some cases, since many virtual

configurations can map onto the same physical configuration. Another limitation of ENI is that it only considers the layouts of the environments, and not any other factors that are known to influence the navigation experience during locomotion, such as the specific paths travelled. Finally, the validation experiments were limited in that we could not test ENI on a large corpus of environment pairs. Although we believe this does not affect the validity of our metric, it is important to evaluate the accuracy of any metric in as many scenarios as possible.

There are many avenues for future work in this area. Since our experiments showed mixed results in terms of the correlation between ENI scores and navigability measures, we would like to further study the ENI metric with a larger set of benchmark environments to get a better understanding of the relationship between our ENI metric and the navigability of environment pairs. Additionally, more user studies should be conducted to investigate whether or not ENI aligns with users' subjective perception of the navigability, though this will require careful consideration to ensure that all participants have the same notion of "navigability." Another area for future work is to investigate the cause of the plateauing effect we saw in navigability scores in Subsection 6.5.1. A detailed understanding of the worst-case navigability for given $\langle$PE, VE$\rangle$ pairs may allow us create standardized benchmarks against which we can compare the efficacy of different locomotion interfaces. Finally, if the computation time for ENI can be improved to interactive rates, we believe that it will also be interesting to evaluate the effectiveness of using ENI to help optimize the layouts of VEs to make them more amenable to navigation (similar to other architecture tools like Goldstein et al. [68]).

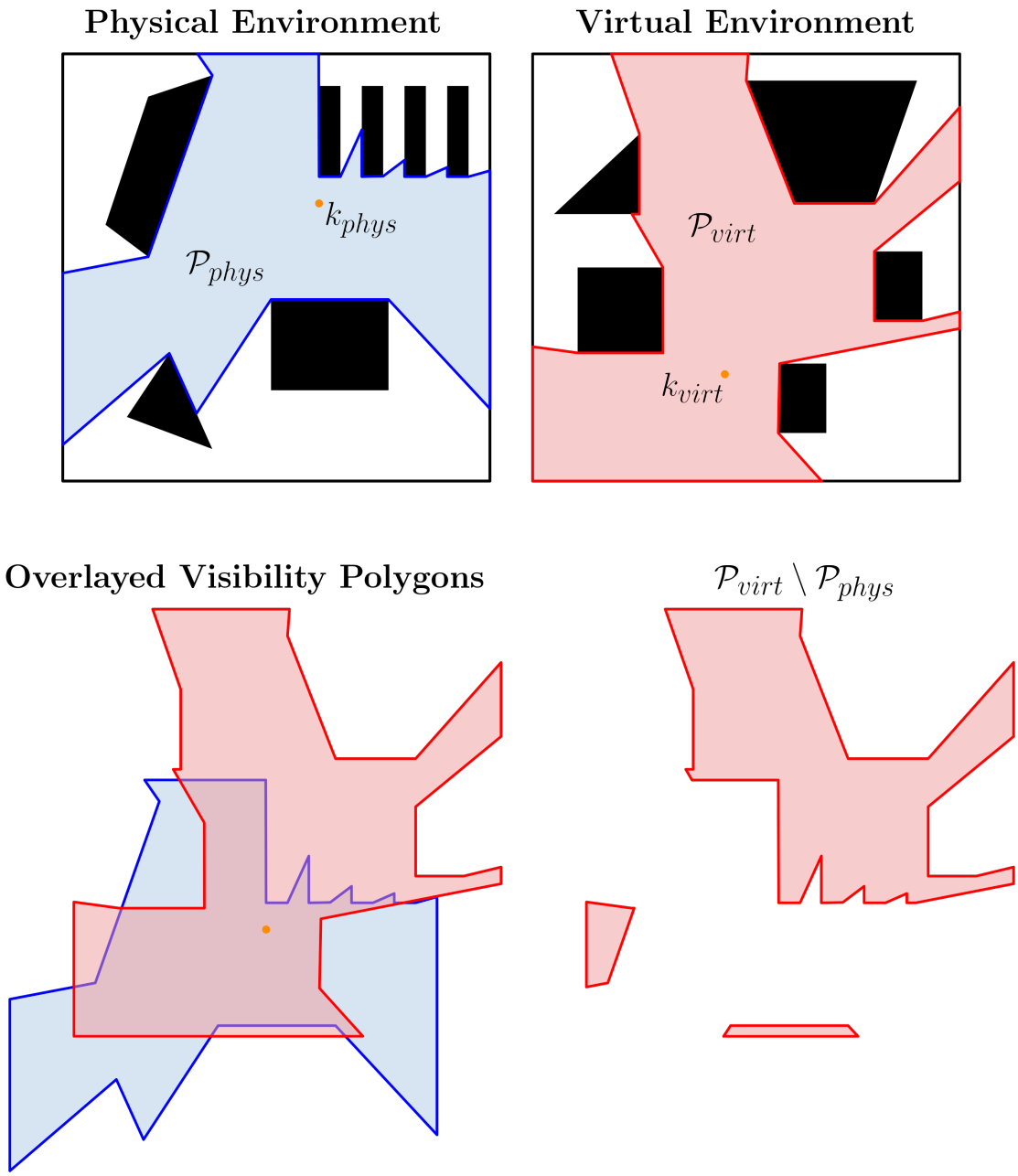


Figure 6.4: *Top row:* Two visibility polygons $\mathcal{P}_{phys}$ and $\mathcal{P}_{virt}$ in a ⟨PE, VE⟩ pair. *Bottom row (left):* $\mathcal{P}_{phys}$ and $\mathcal{P}_{virt}$ have been translated such that their kernels lie on the same 2D position in the plane. *Bottom row (right):* The result of the boolean difference operation $\mathcal{P}_{virt} \setminus \mathcal{P}_{phys}$ is shown as the red polygons. These polygons represent all the regions of $\mathcal{P}_{virt}$ that cannot be accessed when the user is located at k_{phys} and k_{virt} in the PE and VE, respectively. Our metric uses the total area of $\mathcal{P}_{virt} \setminus \mathcal{P}_{phys}$ as a measure of the similarity of the user's local physical and virtual surroundings.

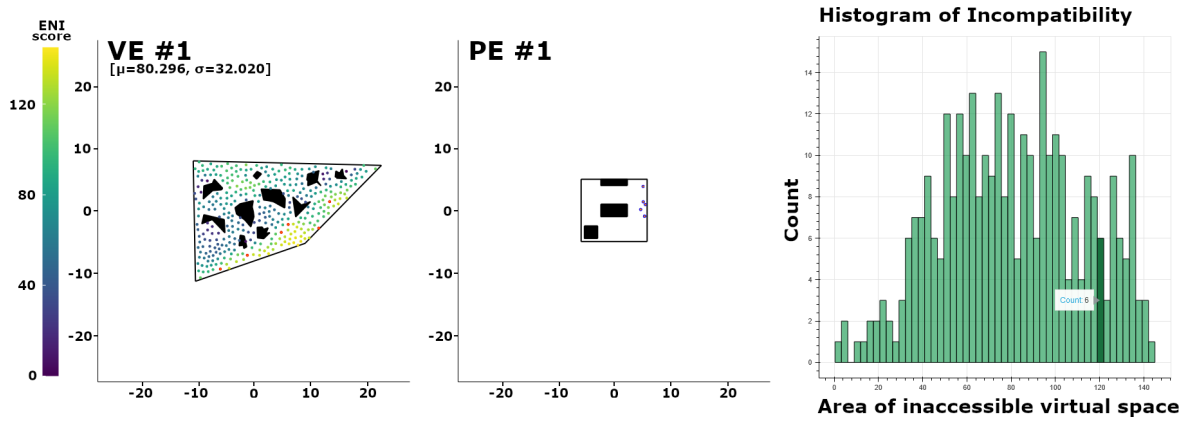


Figure 6.5: The effect of selecting a bar of the histogram in our interactive visualization. When a bar is selected (right), the physical and virtual points that contribute towards this histogram bar are highlighted in orange (left and middle).

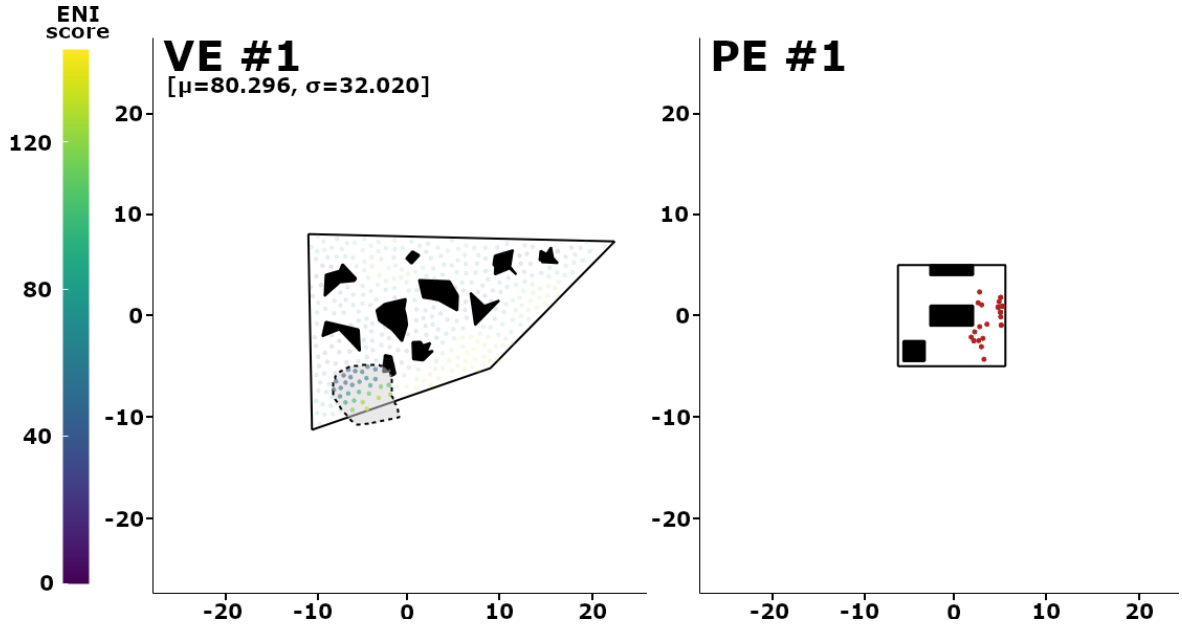


Figure 6.6: The effect of selecting a set of virtual points using the lasso tool. When virtual points are selected (left), the corresponding most compatible points (computed via Equation 6.3.4.3) are shown in red in the PE (right).

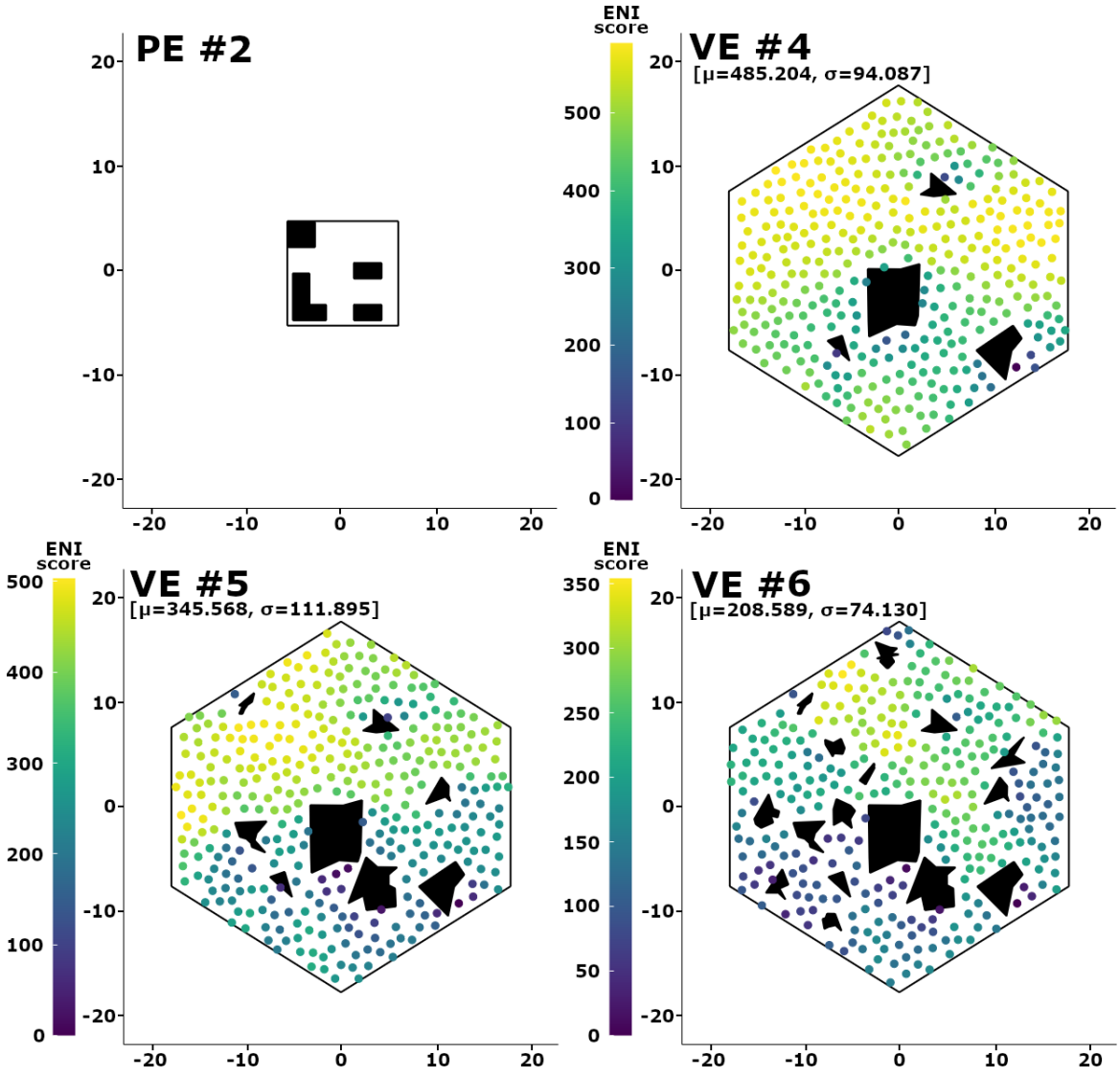


Figure 6.7: ENI metric scores for three different ⟨PE, VE⟩ pairs, where the PE is static and the density of objects in the VE increases. As the density increases, the amount of navigable space decreases, creating a more navigable ⟨PE, VE⟩ pair due to the small size of the PE.

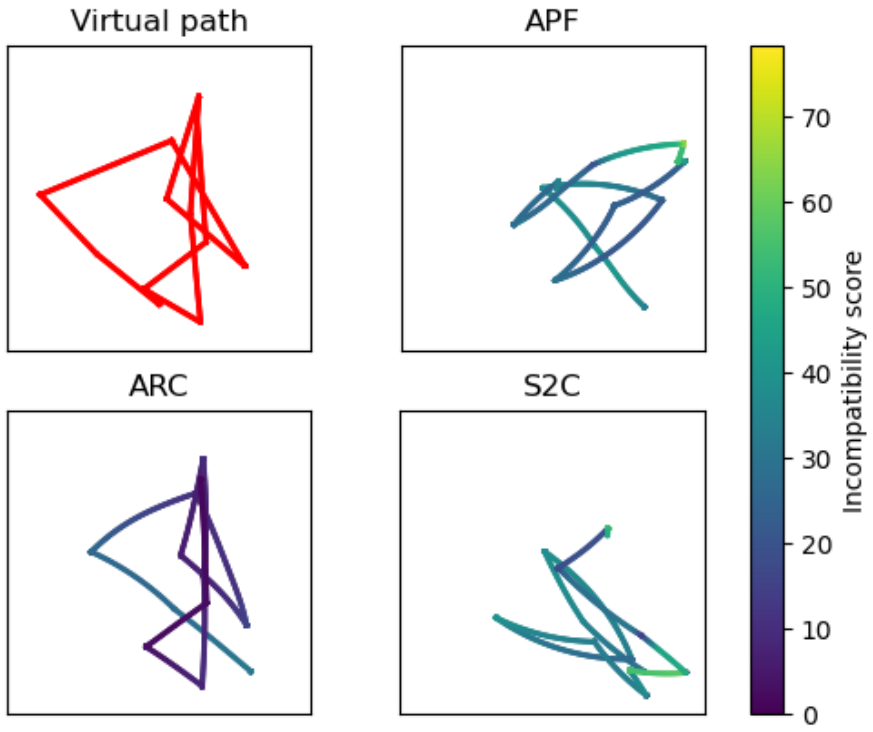


Figure 6.8: *Top left:* A virtual path in an empty $10m \times 10m$ VE. *Top right, bottom left, and bottom right:* The physical path the user travels on when steered by APF [205], ARC [231], and S2C [84]. The physical paths are colored according to the ENI scores between the corresponding points along the physical and virtual paths. The path yielded from ARC is more compatible with the virtual path, suggesting that the user is less likely to incur collisions during locomotion with ARC.

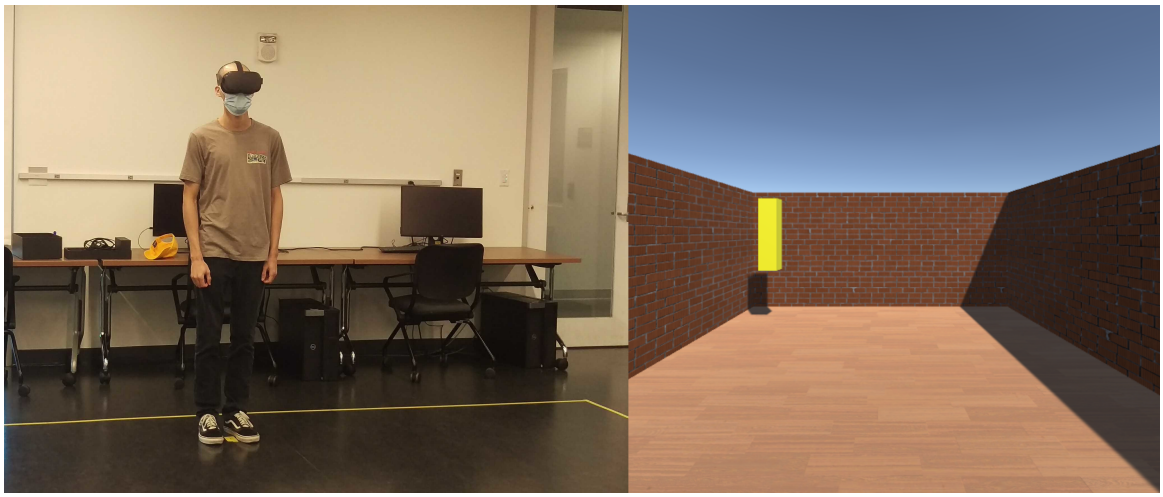


Figure 6.9: *Left:* A user in the lab space in which we conducted our user evaluations. *Right:* A screenshot of the user's starting configuration in the VE at the beginning of a trial in our first user study.

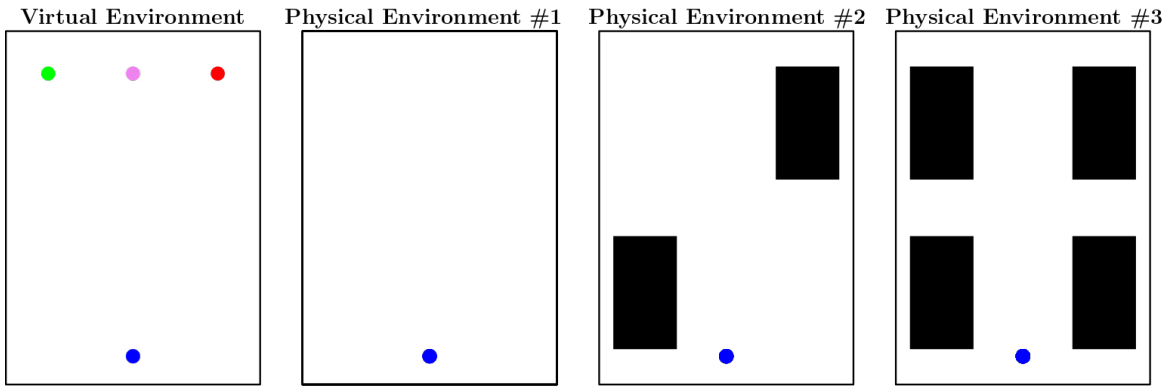


Figure 6.10: Diagrams of the layouts of the VE and PEs used in the first user study. The blue circle indicates the user's starting position in each environment, and the green, pink, and red circles indicate the locations of the goal in the VE during different trials. The dimensions of each environment are $4.37m \times 6.125m$.

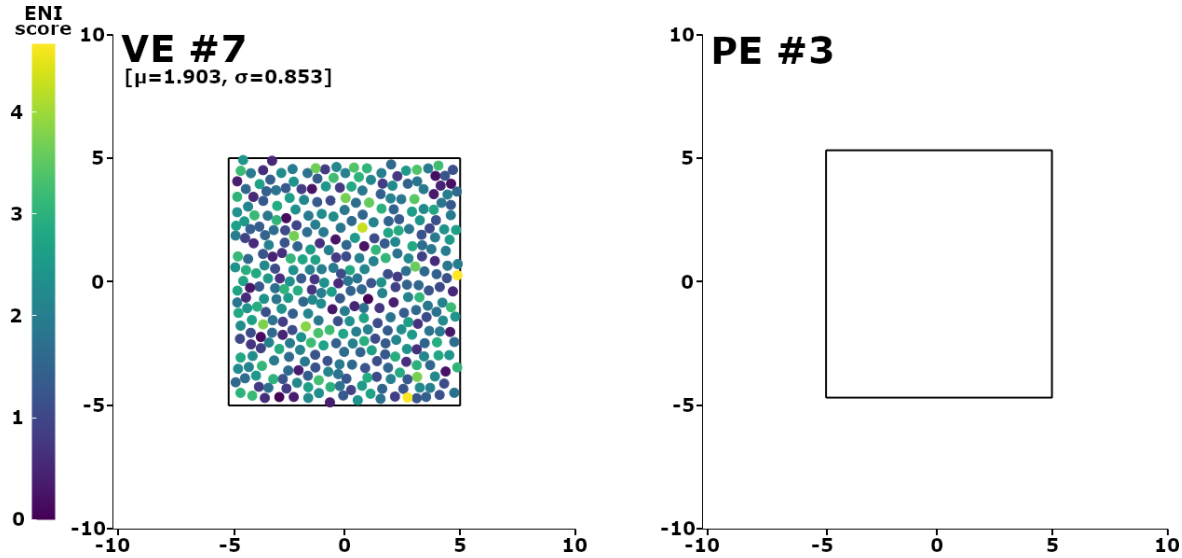


Figure 6.11: Environment A introduced by Williams et al. in [231].

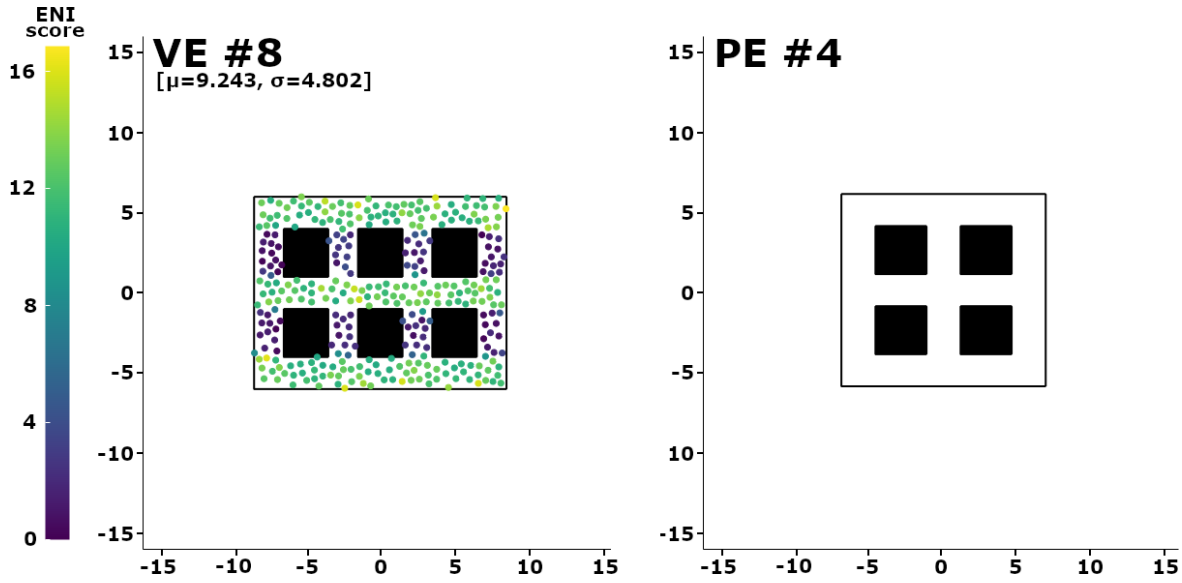


Figure 6.12: Environment B introduced by Williams et al. in [231].

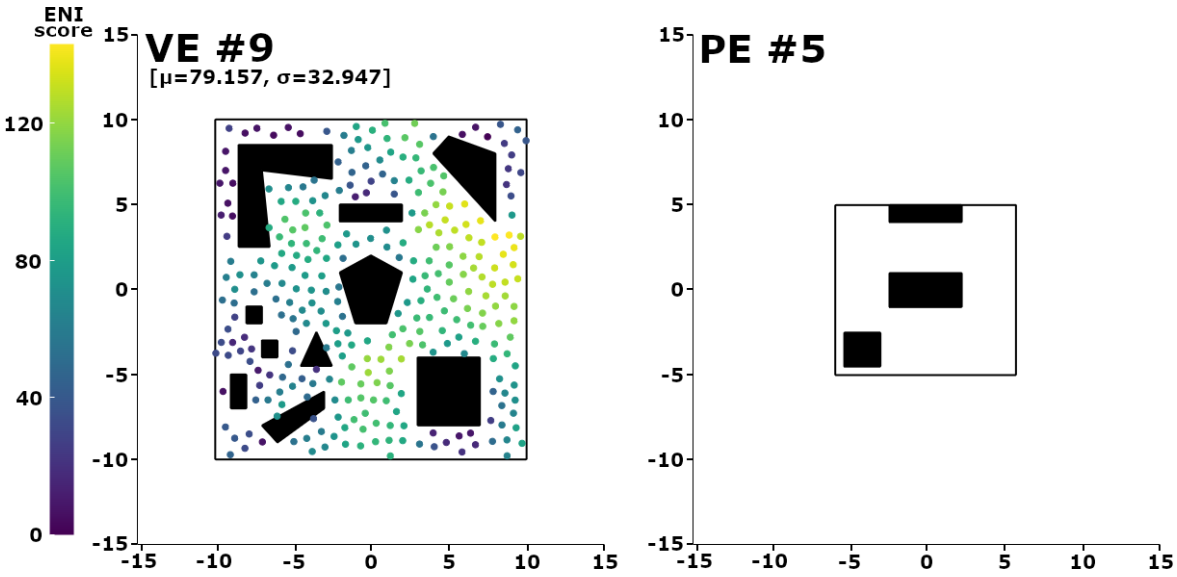


Figure 6.13: Environment C introduced by Williams et al. in [231].

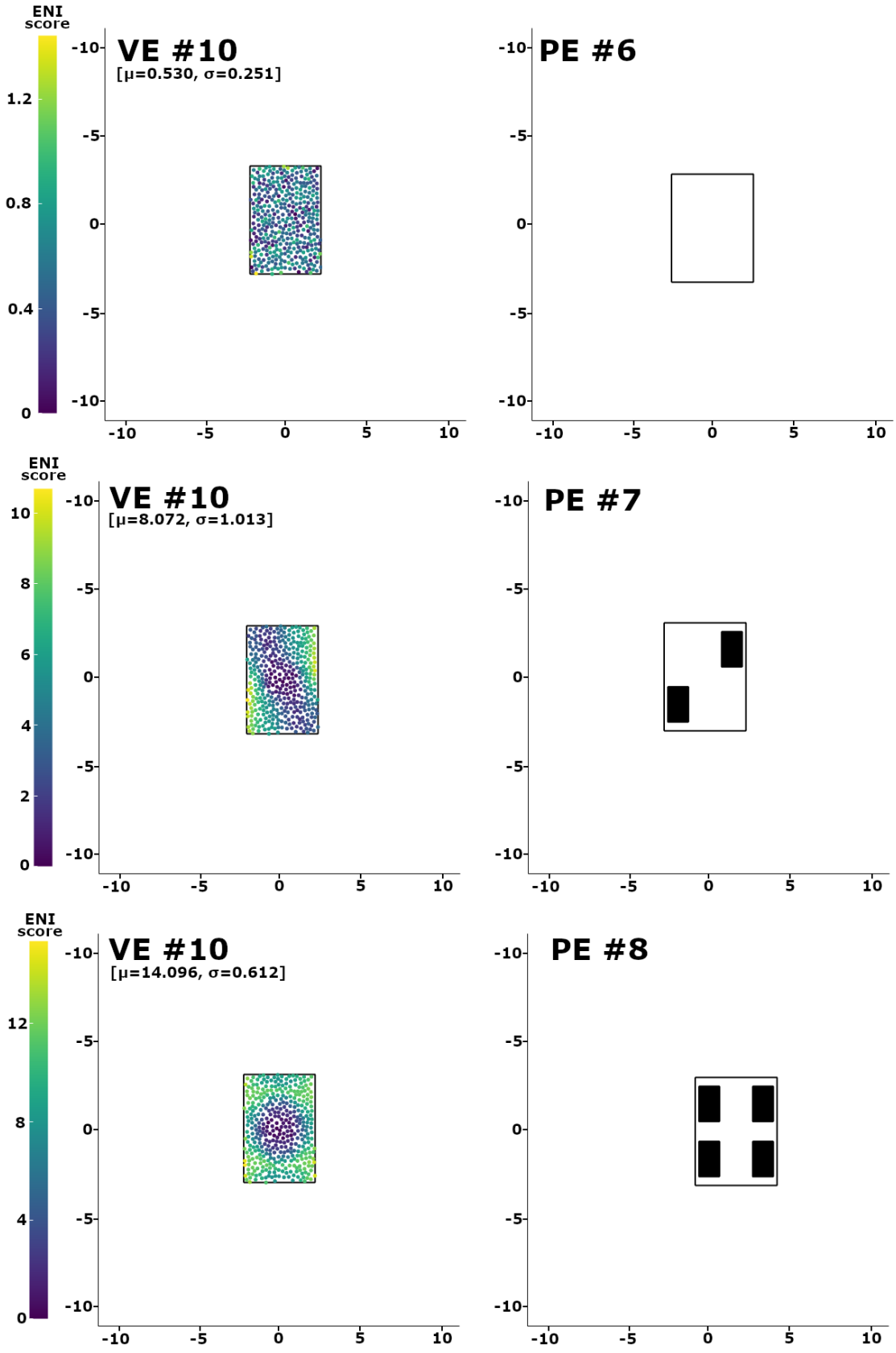


Figure 6.14: The three environment pairs used in our first user study.

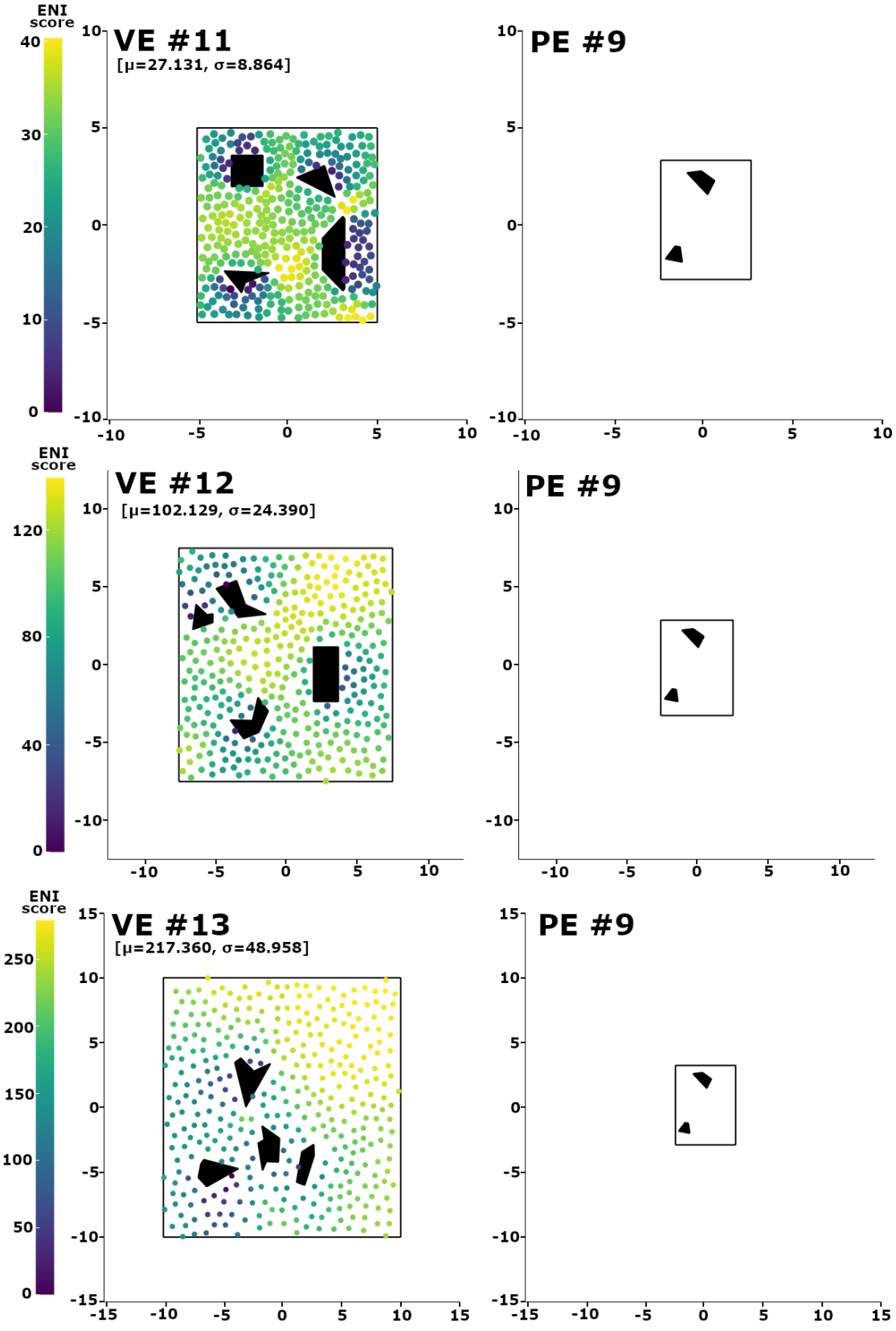


Figure 6.15: The three environment pairs used in our second user study.

Part III

Perception and Physiology During Natural Walking in Virtual Reality

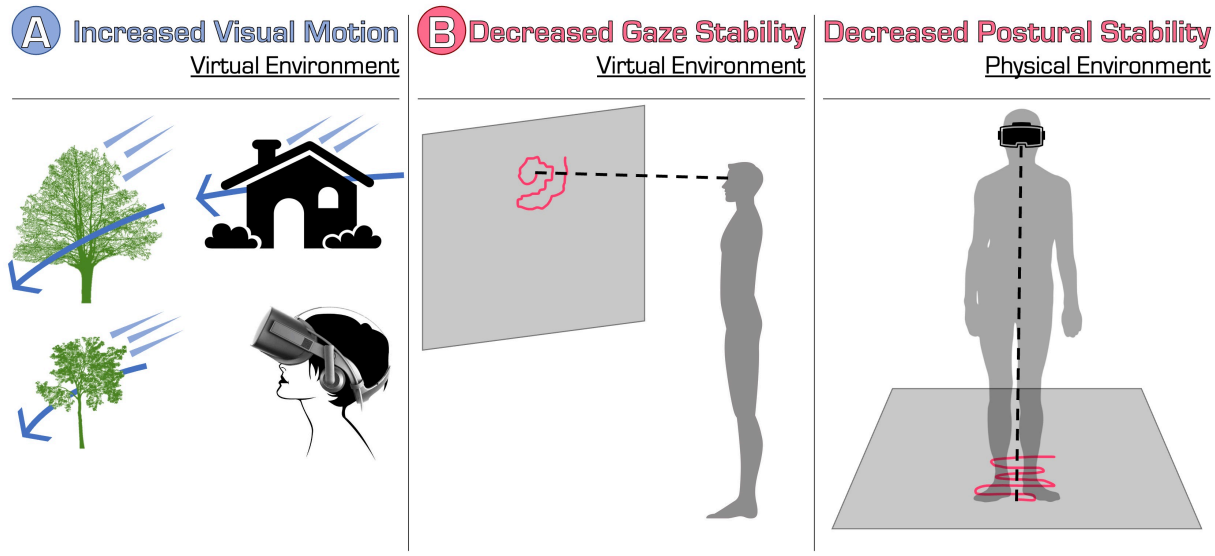


Figure 7.1: A visualization of our experiment paradigm and the properties of physiological signals that we found to be correlated with scene motion during redirected walking (RDW). **(A)** We conducted a psychophysical experiment in which participants completed a rotation task across hundreds of trials, with different amounts of additional scene motion injected into the virtual environment during the rotation. Participants reported on whether or not they perceived the additional injected motions and we computed their visual sensitivity to these motions. **(B)** Our analyses revealed that as the speed of injected motions increased, the stability of participants' gaze (left) and posture (right) *decreased*. These results show, for the first time, a direct correlation between the strength of redirection (injected visual motion gains) and physiological signals.

In this chapter, we study the correlations between redirected walking (RDW) rotation gains and patterns in users' posture and gaze data during locomotion in virtual reality (VR). To do this, we conducted a psychophysical experiment to measure users' sensitivity to RDW rotation gains and collect gaze and posture data during the experiment. Using multilevel modeling, we studied how different factors of the VR system and user affected their physiological signals.

In particular, we studied the effects of redirection gain, trial duration, trial number (i.e., time spent in VR), and participant gender on postural sway, gaze velocity (a proxy for gaze stability), and saccade and blink rate. Our results showed that, in general, physiological signals were significantly positively correlated with the strength of redirection gain, the duration of trials, and the trial number. Gaze velocity was negatively correlated with trial duration. Additionally, we measured users' sensitivity to rotation gains in well-lit (photopic) and dimly-lit (mesopic) virtual lighting conditions. Results showed that there were no significant differences in RDW detection thresholds between the photopic and mesopic luminance conditions.

7.1 Introduction

Natural walking in virtual environments (VEs) is important for providing users with a comfortable and immersive virtual reality (VR) experience [52, 188]. Redirected walking (RDW) is a promising natural locomotion interface that allows users to explore VEs that are larger than the available physical environment (PE) [157, 158]. RDW works by adding imperceptible rotations and translations (controlled by *gains*) to the virtual camera's trajectory through the VE as the user moves around in the PE. In order for RDW to be effective, it is crucial that these added movements are small enough to remain imperceptible to the user. If they are not, it is common for the user to feel symptoms of simulator sickness or have difficulty controlling their locomotion in the VE [140, 187]. To avoid this, researchers have measured the maximum strength of redirection that can be injected before users can reliably perceive it; the weakest RDW gain that is perceptible to the user is known as the *perceptual threshold* (or *detection* threshold), and it is generally advised that gains should not exceed this threshold value in order to avoid causing discomfort for the user [187].

A significant amount of research has been done to study how RDW thresholds change according to different system configurations and using different measurement methods. One major takeaway that can be drawn from prior work is that perceptual thresholds for RDW are highly variable depending on system and user factors: thresholds change depending on gender [139, 230], field of view [230], level of embodiment [139], amount of exposure to RDW [19], audio feedback [66], and from user to user [92]. Typically, RDW thresholds are measured using a psychophysical experiment, which often takes multiple hours for one participant, can be very repetitive and tiring for users, and different psychophysical fitting methods can yield different results under the same experimental conditions or may be sensitive to lapses in perception [226, 227]. The high variability of RDW thresholds is not compatible with the psychophysical methods used to measure them–small changes in the system configuration may yield different thresholds that would warrant re-running the long psychophysical calibration process in order to ensure that the applied redirection gains remain below the user's detection thresholds. Thus, in order for RDW to become a more usable locomotion interface in real-world consumer settings, it is crucial that we develop better methods for *quickly* and *accurately* estimating the imperceptibility of RDW gains during virtual locomotion.

Physiological signals may be one potential solution to this problem. Many researchers have shown a link between users' perceptual experiences in VR (e.g., sense of presence [131] and jitter perception [127]) and their physiological signals (e.g., gaze signals [141], galvanic skin response [95], EEG signals [109], and postural stability [134]). Similarly, researchers in the human vision community have shown that gaze behavior can be informative for estimating an observer's level of engagement with the scene content [155]. However, despite the growing evidence that physiological signals can be useful for inferring about a user's subjective experience

and internal state, researchers have yet to demonstrate a direct correlation between RDW gains and any physiological signals.

Main Results: In this work, we take a first step towards a more efficient and accurate estimation of RDW gain perceptibility by investigating the correlations between RDW gain strength and patterns in users' physiological signals. We conducted a psychophysical experiment to measure users' perceptual thresholds for RDW rotation gains and recorded physiological signals (gaze and posture data) during the experiment. The first main contribution of our work is an initial investigation into the correlations between RDW rotation gain strength and patterns in gaze and posture data. To improve the applicability of our results, we chose to study gaze and posture data because these are physiological signals that are readily available in most modern consumer HMDs; other signals typically require the integration of additional sensors. We limited our study to rotation gains since they allow for a wider range of gains to be applied before users begin to perceive them, compared to curvature and translation gains [187]. The second main contribution of our work is that we measured how rotation gain thresholds change as a function of the luminance of the virtual content. We measured perceptual thresholds in photopic and mesopic light conditions to observe how the perceptibility of rotation gains changes depending on the light level of the virtual content being viewed. Our results showed that gaze and postural stability are significantly correlated with the properties of the RDW system, and that detection thresholds do not change significantly in photopic compared to mesopic luminance conditions. In particular, we found that:

- There were small but significant correlations between rotation gain strength and physiological signals of posture and gaze. Specifically:

  - Each unit increase in rotation gain was correlated with, on average, a postural sway increase of 4.553 cm, a gaze velocity increase of $0.897°/s$, a blink frequency decrease of 0.804 per trial, and a saccade frequency increase of 2.869 per trial.
  - Each additional trial completed (i.e., time spent in VR under the effects of RDW) was correlated with, on average, a gaze velocity increase of $0.00704°/s$ and a blink frequency increase of 0.00520 per trial.
  - Each additional second spent under the effects of RDW per trial (i.e., trial duration) was correlated with a gaze velocity decrease of $0.111°/s$, a blink frequency increase of 0.275 per trial, and a saccade frequency increase of 2.773 per trial.
- Detection thresholds for RDW rotation gains were not significantly different between the well-lit (photopic) and poorly-lit (mesopic) virtual lighting conditions.

7.2 Background and Related Work

7.2.1 Redirected Walking Thresholds

Redirected walking (RDW) is a locomotion interface that enables users to explore VEs that are larger than their PE [158]. It works by adding imperceptible rotations and translations to the virtual camera's trajectory as the user moves around in the physical space, effectively creating a mismatch between the user's physical and virtual movements. The magnitude of these injected rotations and translations is determined by parameters called *gains*, and the smallest gain that is reliably perceptible to a user is known as their *perceptual threshold*. Rotation gains modify the amount that the user turns in place, translation gains modify how far the user travels while walking, and curvature gains cause users to walk on curve physical trajectories while following straight virtual trajectories. Ideally, the gains applied during locomotion are smaller than the

perceptual thresholds in order to mitigate the likelihood that the user feels symptoms of simulator sickness or breaks in presence [187].

Applying perceptually comfortable thresholds requires us to estimate the user's perceptual thresholds, and this is typically done using psychophysical experiments which often take multiple hours to complete per participant [67]. The first comprehensive study that measured RDW thresholds was conducted by Steinicke et al. [187]. Since then, many researchers have studied how thresholds change according to different VR system factors. For example, prior work has shown that RDW thresholds are influenced by HMD field of view [230], walking speed [137, 138], level of embodiment in VR [139], the rate of change of gains [41], and the amount of exposure to RDW [19]. Additionally, individual differences between users have an effect on thresholds. Differences in thresholds have been found between men and women [138, 139, 230], and individual thresholds can vary widely from participant to participant [92, 138]. Research has also shown that sensitivity to RDW can change depending on the context of the virtual experience, such as while the user is opening a virtual door [87] or is distracted by virtual objects or tasks [139, 148, 230], or the size of the virtual room [106]. Finally, there is some evidence that estimated threshold values can vary depending on the psychophysical methodology used [71, 92].

In summary, these results show that RDW thresholds are complex and nuanced: thresholds depend on user factors, device factors, VE context, and amount of user experience. As VR technology improves and increases in popularity, VR devices will gain new features and capabilities, and users will engage with VR in a wider range of (physical and virtual) contexts. Thus, in order to help make RDW a practical technology that does not require hours of calibration before it can be used in every new scenario, our work investigates the use of physiological signals for estimating users' ability to perceive redirection. Furthermore, to test the reliability of these

signals in a variety of different configurations, we study how thresholds change according to the luminance level of the display. We chose to study thresholds as a function of luminance because it is a common visual feature of virtual content that is likely to vary widely for different virtual experiences, and since we expect luminance to become a more-relevant feature in future devices with the advent of high-dynamic range VR [129]. Furthermore, prior work in motion perception [72, 74, 135] provides evidence that luminance has an effect on an observer's perception of motion, which has implications for users' perception of RDW gains in VR.

7.2.2 Physiological Signals of Users' Internal State

In addition to thresholds being highly dependent on system and user factors, another challenge with understanding user sensitivity to RDW is the fact that psychophysical experiments are often very time consuming and expensive [143, 223, 225]. Psychophysical experiments often take multiple hours and require users to complete a repetitive, tedious task in order for researchers to estimate their sensitivity to the stimulus in question. To mitigate this problem, researchers have developed different techniques for sampling the stimulus parameter space to more efficiently converge to the user's detection threshold (e.g. PEST [204], QUEST [223], AEPsych [143]), but these methods can still be sensitive to lapses in judgement or may require careful calibration to maximize the likelihood of convergence. Furthermore, traditional psychophysics requires users to first perceive the stimulus (possibly multiple times) and then answer a question about what they perceived, which is an intrusive process that requires its own dedicated calibration session. To help mitigate these constraints, researchers have shown that physiological signals can be used to draw conclusions about a user's subjective feelings of comfort or other aspects of their internal state such as level of engagement.

7.2.2.1 Gaze Stability

Prior work has shown that gaze movements, and in particular metrics for gaze stability, are correlated with symptoms of motion sickness in users. Ujike et al. [208] found that torsional eye movements had a significant correlation with symptoms of motion sickness for rotations about the roll axis. Similarly, Hemmerich et al. [77] found a small but significant correlation between motion sickness and eye movements (specifically, scan-path length and dispersion). Bouyer et al. [20] had participants repeatedly perform a "torso rotation" movement in which they shifted their gaze side-to-side between two targets placed at either end of their horizontal peripheral vision, which required coordinated head, eye, and torso rotations. This motion is fairly similar to the rotation movement participants completed in our rotation gain discrimination task. Bouyer et al. found that gaze stability (measured by participants' ability to consistently focus their gaze on a target) decreased as feelings of motion sickness increased. Inspired by those works, we evaluate the correlations between redirection strength and participants' average gaze velocity, where gaze velocity is used as a proxy for gaze stability since we did not instruct participants to fixate on a particular target in the virtual environment. Additionally, we measure correlations between redirection strength and saccade and blink frequency since these are gaze metrics that are readily available in eye-tracked HMDs.

7.2.2.2 Postural Stability

Posture control and optic flow are tightly linked to locomotion control [86, 100, 103, 114, 130]. That is, prior work has shown that optic flow at least partially modulates locomotion such that users alter their locomotion behavior to better match the perceived optic flow [58, 153]. Furthermore, alteration of the visual environment can induce a compensatory postural response

[118]. Wang et al. [219] showed that large changes in torso sway can be strongly correlated with perturbations in balance, which precede falls by the walking person. In VR, researchers have used measures of postural stability to estimate users' level of simulator sickness. Chardonnet et al. [31] showed that the spread of the center of gravity and the shape of postural sway in the time domain can serve as indicators of simulator sickness in virtual reality. Finally, Soffel et al. [181] demonstrated how a VR HMD can be used to measure postural stability without the need for a balance board or force plates. Riccio and Stoffregen [160] proposed a theory that postural instability is the source of motion sickness that user experience (both in real and virtual environments). Numerous studies have been conducted that provide some empirical evidence to support this theory [12, 189, 190, 191, 202], though we note that some studies have found a lack of relationship between postural stability and feelings of motion sickness [40, 51, 222]. Given the evidence from prior work that shows a link between postural sway and motion sickness, we chose to study correlations between postural sway and redirection gain strength since simulator sickness is a common side effect of RDW.

7.2.3 Luminance & Motion Perception

Researchers in the human vision community have studied the degree to which an observer's ability to detect motion is affected by the luminance of the visual content. Sara et al. [164] found that observers had decreased motion detection performance in low-light conditions, while Takeuchi et al. [201] showed that velocity discrimination thresholds vary depending on luminance level. For illusory motion in static stimlui, Hisakata et al. [82] showed that motion perception varied as a function of retinal illuminance. Interestingly, Billino et al. [18] found that detection thresholds for biological motion were the same for photopic and scopotic conditions, despite the

majority of prior research showing that motion perception performance generally decreases as luminance decreases [239]. Overall, the literature on motion perception in different light levels shows that perception of motion changes according to changes in luminance, which motivated us to study how users' sensitivity to RDW gains changes as a function of the luminance of the virtual content shown on the HMD. Indeed, with the introduction of high dynamic range VR HMDs [129], we believe that it is important for us to gain a better understanding of motion perception in VR according to the light level of virtual content.

7.3 Experimental Methodology

In order to effectively utilize RDW, the injected rotations and translations should lie below the user's detection thresholds so as to avoid making the user feel sick. However, measuring these detection thresholds is a tiring and repetitive process that often entails multiple hours of psychophysics. Furthermore, it is well-known that a user's sensitivity to RDW can change depending on various system and user factors such as field of view [230], amount of exposure to RDW [19], and individual differences in perception [92]. This poses a problem since the dynamic nature of RDW perceptibility implies that thresholds should be estimated frequently (as different factors influence a user's sensitivity change during the virtual experience), but current methods for estimating sensitivity to RDW often require multiple hours or, at best, on the order of $\sim 20 - 30$ minutes for adaptive methods [225]. Our motivation with this work is to investigate to what degree the strength of redirection gains is correlated with different physiological signals of the user, in hopes that such insights could be used to estimate a user's sensitivity to RDW in real time via online analysis of physiological signal patterns, thus bypassing the long psychophysical estimation process that has traditionally been used for RDW.

To achieve this, we ran a psychophysical study in which we estimated participants' RDW gain detection thresholds while also collecting different physiological signals (posture and gaze data). After collecting the data, we studied the correlations between redirection strength and different metrics derived from participants' physiological data. We limited our study to rotation gains since they allow for larger magnitudes of redirection to be applied [187] (giving us a wider range of gain values with which we can test for correlations). Additionally, we chose to study participants' posture data and gaze data since these are two biosignals that are readily available in most modern VR devices without requiring external devices, and since there is a large amount of prior work (Subsection 7.2.2) to suggest that redirection gains will be correlated with patterns in users' physiological data.

Note that in this work, we are *not* directly measuring a correlation between physiological signals and a user's subjective level of comfort (i.e., tolerance for redirection gains); we only measure the existence of any correlations between redirection gain strength and posture/gaze data. Since we are not aware of any prior work that has established a direct correlation between redirection gain strength and physiological signals, the first step to using physiological signals as an alternative to psychophysics is to confirm that the strength of redirection gains are indeed correlated with patterns in a user's biosignals. Once such a correlation has been established, follow-up works should study to what extent these signals can serve as a direct indicator of a user's subjective level of comfort during locomotion.

Additionally, in an attempt to better understand the different factors that may affect sensitivity to RDW, we measured users' sensitivity to rotation gains in both bright (photopic) and dark (mesopic) luminance conditions. Considering the gamut of different virtual experiences users may find themselves in VR, it is not unlikely that users will interact with virtual environments

that have different ranges of luminance values (e.g. visual content for horror games tend to have low luminance, while job simulators are more likely to have bright visual content akin to daytime luminance). Prior work in human perception has shown that observers can have differing sensitivity to visual motion depending on the luminance of the visual stimuli (Subsection 7.2.3). Considering these facts and the introduction of high dynamic range VR [129], we were motivated to study how sensitivity to redirection changes depending on the luminance of the virtual content.

7.3.1 Experiment Design & Stimuli

Our experiment was limited to rotation gains because rotation gain experiments require less physical space to conduct and they allow for larger redirection than curvature or translation gains [187]. We had two hypotheses that we aimed to test with our experiment:

H1 Rotation gains will be significantly correlated with patterns in participants' gaze and posture data.

H2 Rotation gain thresholds are different in mesopic viewing conditions compared to photopic viewing conditions.

For **H1**, we hypothesized that the strength of redirection would predict changes in participants' posture and gaze data since prior work showed that these biosignals are correlated with feelings of MS, and MS is frequently induced by prolonged exposure to RDW [140]. The intuition behind **H2** is that prior work in motion detection has shown that observers tend to have different sensitivity to visual motion depending on the luminance level of their surroundings (Subsection 7.2.3), and a large component of users' ability to detect RDW is their sensitivity to visual motion. Therefore, we hypothesized that the light level of the virtual content would have an effect on users' ability to detect the presence of RDW gains.

To measure RDW thresholds, we conducted an experiment that required users to repeatedly complete a 2-alternative forced choice (2AFC) task [67]. Participants were tasked with rotating their whole body in place where they stand, either left or right (indicated by an arrow near the top of the field of view), until they heard a beep tone that was their signal to stop rotating. After the participant stopped rotating and maintained their current orientation for 1 second, a green check mark appeared and the trial ended. If the user rotated too far or in the wrong direction at the start of the trial, a yellow arrow appeared that signaled them to turn in the other direction to complete the trial. The amount rotated in the virtual environment was always $90°$ ($\pm 5°$ so as to avoid requiring infeasibly-precise rotation from the participants), but the amount of physical rotation varied depending on the strength of the redirection gain applied. After the participant completed the rotation task, the view faded to black and the user was prompted to answer the following question [187, 230]: *"Was the virtual movement smaller or greater than the physical movement?"* (Response options: Smaller, Greater). The gain applied on each trial was randomly chosen from a set of nine gains $\{0.6, 0.7, 0.8, 0.9, 1.0, 1.1, 1.2, 1.3, 1.4\}$, and each gain was tested 12 times for a total of 108 trials. In order to elicit natural eye movements, the direction arrow disappeared shortly after the user began rotating and participants were not given any instruction on where they should look during the rotation trials.

An observer's ability to discern objects in their surroundings changes depending on the amount of light illuminating their environment. As the light level changes, the eye adapts and vision improves over time in a process called light adaptation. In particular, when shifting to a low-light environment, it can take up to two hours for the observer to fully adjust to the low-light conditions [26]. This required us to separate the light and dark luminance conditions into two separate blocks, so that the participants' eyes could adequately adjust to the light level of either

condition. If light and dark trials were interleaved within the same block, participants would not have enough time to meaningfully adjust their vision to properly perceive the virtual content. Blocks were separated by at least 12 hours and the order in which participants experienced them was counterbalanced. In the light condition, the average luminance of the virtual content in the HMD was about $25\ \frac{cd}{m^2}$ (photopic vision). In the dark condition, the average luminance was about $0.4\ \frac{cd}{m^2}$ (mesopic vision). The dark condition was implemented using a shader that uniformly lowered each pixel's brightness value to 2% of its original value. Pilot tests revealed that it was not feasible to change the luminance of the virtual content to a scotopic level while having the virtual content still visible to the user.

The virtual environment was an office with some pieces of office furniture placed throughout the environment. We used this environment for our experiment since it is a familiar setting that would have enough variation in color contrast and texture patterns to elicit sufficient optical flow for adequate motion perception [221]. An example of the stimuli users saw in the photopic condition is shown in Figure 7.2. After completing a block, simulator sickness symptoms were recorded using the Kennedy-Lane Simulator Sickness Questionnaire (SSQ) [104]. Participants received a $\$10$ or $\$40$ Amazon gift card after completing the first or second block, respectively.

7.3.2 Equipment & Participants

Our experiment was implemented using the Unity 2021.3.5f1 game engine and a Meta Quest Pro VR HMD. The HMD was tethered to a desktop computer (CPU: Xeon 2.2GHz, RAM: 64GB, GPU: dual GTX2080, operating system: Windows 10). We conducted the experiment in a private lab room with covered windows so that the light level in the room could be carefully controlled. In the mesopic condition, the lights in the room were dimmed to mitigate any

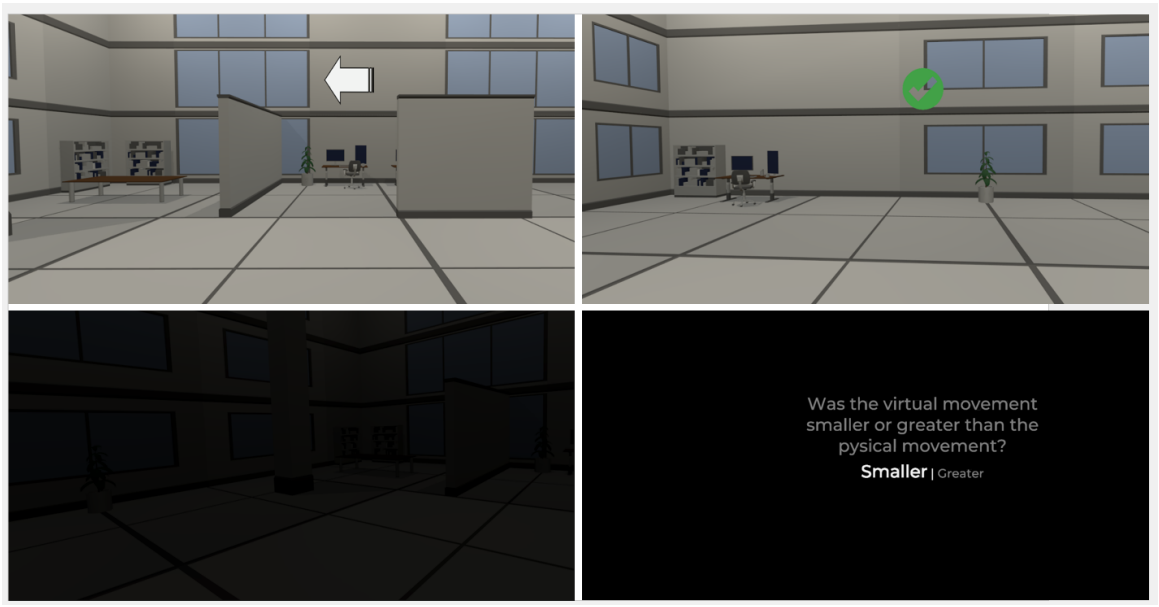


Figure 7.2: Screenshots of the virtual office environment used in our experiment (during the photopic condition). Ambient office sounds were played to help mitigate the viability of using sounds from the physical environment as a cue for the participant's orientation in the physical space. **(A)** The view of the environment that participants saw at the beginning of each trial. The white arrow indicated to the user which direction they should rotate, and this arrow disappeared after they rotated 5° from the starting position in the direction of the arrow. **(B)** An example view of the environment at the end of a trial. When the user rotated 90° in the virtual environment ($\pm 5^\circ$), a beep tone was played that indicated that the user should stop rotating and maintain their current orientation in the environment. After maintaining this orientation for 1 second, a green check mark appeared to indicate that they successfully completed the trial.

adaptation to scoptic light levels in case the participant took a break and removed the HMD during the experiment. The room could not be completely darkened because the Quest Pro uses inside-out tracking that requires enough ambient light to be able to track the relative motion of the physical surroundings. We recruited eight participants (four female, age $\mu = 23.625, \sigma = 2.504$) who successfully completed both blocks of the experiment. Note that for psychophysical experiments, it is common to have a single-digit number of participants since each participant provides a large amount of data and each individual participant is treated as an individual replication of the experiment [88, 180]. That is, the objective of psychophysics is to study the mechanics of the perceptual system, so average population effects are largely uninformative since hypotheses

are tested within the individual participants [88]. Participants had normal or corrected-to-normal vision and were at least 18 years old. The HMD was adjusted to match the participant's interpupillary distance and the eye tracker was calibrated for the user.

7.4 Results

7.4.1 Rotation Gain Thresholds

Rotation gain thresholds were computed by fitting a cumulative Gaussian distribution to participants' average likelihood of responding *"greater"* for a trial. Threshold estimates are shown in Table 7.1. To evaluate the effects of display luminance on perception thresholds, we conducted a 9 (Gain: $0.6 : 1.4 : 0.1$) $\times$ 2 (Luminance: photopic, mesopic) ANOVA. We found that gain had a significant effect on the likelihood of the participant to respond *"greater"* ($F(8, 42) = 10.405$, $p < .0001$, $\eta^2 = 0.39$), but found no effect of luminance on this likelihood of *"greater"* response ($F(1, 42) = 1.405$, $p = 0.238$, $\eta^2 = 0.00543$). Additionally, no significant interaction effects between threshold and luminance were found ($F(8, 42) = 0.497$, $p = 0.857$, $\eta^2 = 0.02$). These results do not support our first hypothesis that detection threshold values would be different in low-light and well-lit viewing conditions. Since we found no effects of luminance, the remaining analyses were conducted using the pooled mesopic and photopic trial data. A plot of the psychometric curves for mesopic and photopic conditions averaged across all participants is shown in Figure 7.3.

7.4.2 Simulator Sickness

Simulator sickness values for each participant are shown in Figure 7.4. In general, participants' SS values were not abnormal for a rotation gain threshold experiment. Participant 5 showed the highest SS levels (89.76 and 123.42 for the photopic and mesopic conditions, respectively) but

ID (luminance)	25%	μ (PSE)	75%	σ
1 (photopic)	0.595	0.900	1.206	1.454
2 (photopic)	1.189	1.217	1.245	1.041
4 (photopic)	0.849	1.099	1.350	1.371
5 (photopic)	0.846	1.133	1.421	1.427
6 (photopic)	0.729	1.282	1.835	1.820
7 (photopic)	0.551	0.863	1.174	1.462
8 (photopic)	0.580	1.227	1.874	1.959
9 (photopic)	1.021	1.137	1.252	1.171
Average	**0.795**	**1.107**	**1.420**	**1.463**

ID (luminance)	25%	μ (PSE)	75%	σ
1 (mesopic)	0.683	0.933	1.184	1.372
2 (mesopic)	1.107	1.192	1.277	1.127
4 (mesopic)	0.944	1.233	1.522	1.429
5 (mesopic)	0.847	1.127	1.408	1.416
6 (mesopic)	1.010	1.605	2.200	1.882
7 (mesopic)	-0.122	0.868	1.859	2.468
8 (mesopic)	0.668	1.214	1.759	1.809
9 (mesopic)	1.059	1.110	1.160	1.075
Average	**0.775**	**1.160**	**1.546**	**1.572**

Table 7.1: Individual psychometric fits for each participant in the photopic and mesopic light conditions. We show the point of subjective equality (PSE), the standard deviation of the Gaussian (σ), and the 25% and 75% detection threshold gains. We also show the average for each of these values at the bottom of each sub-table. In general, participants exhibited RDW detection thresholds that were within the ranges found in prior work [111], though there is significant inter-participant variability [92]. Our results showed no significant differences in detection thresholds between the two lighting conditions.

did not display any concerning symptoms during the experiment. Inspection of this participant's physiological data and threshold data (Table 7.1) did not reveal any noteworthy outlying data, so we included their data in our analyses. A Wilcoxon signed-rank test revealed no significant effect of luminance on SS values ($T = 9.0$, $p = 0.398$).

7.4.3 Postural Data

First, we compute the average postural stability for each trial. For each trial, we compute the centroid of the position of the HMD projected onto the floor (i.e., the HMD's 2D position ignoring height) across all trials. We use this centroid point as a proxy for the user's base position that they would stay in if there were no postural sway. Then, we compute the distance between this centroid point and the HMD's 2D position for each frame of the trial; this gives a measure of how far the user strays from the central position on any given frame. Averaging these distances yields our average postural sway metric, where a larger number indicates higher *instability* (more time spent at a distance far away from the centroid point). Some example plots of one participant's posture data over the course of a trial are shown in Figure 7.5. In this chapter, all postural sway values are reported in centimeter units.

To analyze the viability of using postural stability as a signal for RDW tolerance, we fitted a multilevel model to the pooled participant trial data using the `lme4` package and residualized maximum likelihood in R. By doing this we are able to measure the correlation between independent variables like rotation gain and physiological signals like postural stability. The model we fit was:

$$\begin{aligned} avgPosturalSway =& (1|ID) + gain + gender + trialNum \\ &+ trialDuration + gain \times trialDuration. \end{aligned} \tag{7.4.3.1}$$

Using this model, we treat the average postural sway as the dependent variable and model how different independent effects may be correlated with postural sway. Our independent variables are rotation gain, participant ID, participant gender, trial number, and trial duration. Participant ID is treated as a random variable, while all others are fixed variables.

Results of fixed effects estimates indicated that gain (slope $=$ 6.080, SE $=$ 0.00612,

$t = 9.936, p < 0.0001$) was significantly correlated with the average postural sway. Intuitively, this can be interpreted as meaning that as the gain value increases by 1, the average postural sway increases by 6.080 cm. No significant main effects of gender or trial duration were observed. Random effects analysis revealed very little variability between participants (variance $= 0.0218$, SD $= 0.0148$), which indicates that the results were not noticeably biased by any outlier participants who completed our experiment (postural sway varied by only 0.0218 cm from participant to participant, on average).

7.4.4 Gaze Data

In general, participants' gaze behavior during trials was characterized by the optokinetic reflex [121, 165] and vestibular nystagmus [96, 121]. Since vestibular nystagmus is known to be present in both light and dark lighting conditions [70], we did not separate the analyses based on lighting conditions. To study correlations between gaze behavior and the strength of redirected walking, we measured how a different gaze-related metrics changed over the course of our threshold experiment. In particular, we looked at average gaze velocity ($^\circ/s$), number of blinks, and number of saccades. We classified saccades as any eye movements that exceeded $30^\circ/s$ [46, 212]. Note that since we did not instruct users to fixate on a specific target, we could not directly measure gaze stability (in a manner similar to postural stability in Subsection 7.4.3). That is, without knowing exactly what target the user intends to fixate on, we cannot measure to what extent their gaze deviates from said target. Instead, we studied gaze velocity *only during fixations* as a proxy measure for gaze stability. Manual inspection of patterns in participants' vestibulo-ocular reflex (VOR) gain did not reveal any noteworthy patterns (the VOR gains during fixations were near 1, as expected) so we did not conduct any further VOR gain analyses.

We fitted a separate multilevel model to each of the three gaze metrics we studied (average velocity, number of blinks, and number of saccades) using the `lme4` package in R with residualized maximum likelihood. By modeling average gaze velocity per trial as the dependent variable and participant ID and gender, rotation gain, trial number, and trial duration as independent variables (Equation 7.4.4.1), we found that trial number (slope $= 0.007004$, SE $= 0.00139$, $t = 5.025$, $p < 0.0001$) and trial duration (slope $= -0.123$, SE $= 0.0204$, $t = -6.000$, $p < 0.0001$) were significantly correlated with average gaze velocity. Additionally, there was a trending, but not statistically significant, interaction effect between gain and trial duration (slope $= 0.0724$, SE $= 0.0372$, $t = 1.949$, $p = 0.0517$). Random effects analysis revealed that there was very little difference in average gaze velocity between participants (variance $= 0.451°/s$, SD $= 0.671°/s$).

$$\begin{aligned} avgGazeVelocity =& (1|ID) + gain + gender + trialNum \\ &+ trialDuration + gain \times trialDuration \end{aligned} \tag{7.4.4.1}$$

We fit the same model for the number of blinks and saccades per trial. For number of blinks, we found that trial number (slope $= 0.00524$, SE $= 0.00226$, $t = 2.315$, $p = 0.0208$), trial duration (slope $= 0.291$, SE $= 0.0280$, $t = 10.384$, $p < 0.0001$), and the interaction between gain and trial duration (slope $= -0.194$, SE $= 0.0758$, $t = -2.558$, $p = 0.0106$) were significantly correlated. The difference in blink frequency between participants was non-trivial but not particularly large (variance $= 2.645$, SD $= 1.626$). For number of saccades, the fitted model revealed a significant correlation with gain (slope $= 4.782$, SE $= 2.204$, $t = 2.169$, $p = 0.0302$) and trial duration (slope $= 2.792$, SE $= 0.0916$, $t = 30.469$, $p < 0.0001$). Individual participants had very different saccade frequencies, as indicated by the random effects

analysis on participant ID (variance $= 16.07$, SD $= 4.009$).

7.5 Discussion

7.5.1 Detection Thresholds and Sickness Scores

In general, the simulator sickness scores of our participants were similar to prior work (e.g., [230]). Additionally, we were able to fit a psychometric curve to all of our participants' data without convergence issues (Table 7.1). Note that some of the individual fitted thresholds are somewhat wide, though it is already well-known that sensitivity to RDW can vary widely from person to person [92, 138].

In our work, we did not find any significant effect of light level on detection thresholds. We believe this is likely due to the complex, multimodal nature of self-motion perception since participants' ability to detect rotation gains depends on both visual *and* non-visual cues of self-motion. In our experiment, users' had to determine if there were additional visual motion gains added to their virtual movement and *compare the differences* between their visual and non-visual signals of self-motion. Self-motion perception requires the brain to integrate motion signals from different sensory channels into a single, coherent "story" about how the body is moving [34, 168]. Furthermore, this integration process is known to change depending on the *reliability* of the sensory information [63], such that less reliable sensory inputs contribute less to the determination of body motion. Thus, it is possible that participants integrated their signals of self-motion differently for the photopic and mesopic conditions in our experiment, and that these different cue-combinations may have yielded similar performance for rotation perception in our experiment. Informal interviews with participants revealed that some participants employed a different strategy for the photopic and mesopic conditions. For example, one participant stated

that they found themself relying less on the visual information and more on the duration of each rotation trial in addition to their non-visual motion signals to try to detect the redirection.

Furthermore, we note that our experiment task was a rotation discrimination task. Prior work in vision science has shown that people are able to accurately perceive rotation magnitudes even in complete darkness [172], which may further complicate the nature of RDW detection in photopic and mesopic conditions. For these reasons, we believe that it is not necessarily surprising that we did not see any significant differences in detection thresholds between the photopic and mesopic conditions.

7.5.2 Gaze Data

Our results on gaze data during RDW showed that the participant's average gaze velocity, number of blinks, and number of saccades were all significantly correlated with the RDW gain or time spent being redirected. We used average gaze velocity as a proxy for gaze stability since participants were not tasked with fixating on any target in particular, so gaze stability with respect to a visual target could not be measured.

As the RDW gain increased (the same physical rotation yielded a larger virtual rotation), we found that participants had higher average gaze velocity. This make sense when we consider the rotation task and how gaze behavior works during head and body rotations. In healthy people, nystagmus is induced by body or head rotations as a mechanism for maintaining a stable view of the observer's surroundings. Optokinetic nystagmus occurs when the observer perceives a large moving field during rotation (as in the photopic condition of our experiment), while vestibular nystagmus occurs during rotations even in darkness [1]. Gaze position profiles during nystagmus are characterized by a "saw tooth" shape as the eye slowly drifts in the direction opposite to the

head rotation and then quickly jumps back in the direction of rotation. This pattern can be seen in one user's data in Figure 7.6. The frequency of nystagmus is a function of the speed of the head rotation and/or the perceived visual field motion. Therefore, as the rotation gain increases, the speed of the rotation of the virtual environment increases, which creates more nystagmus-induced saccades that raise the average gaze velocity of the trial. That is, the larger the redirection gain, the faster the virtual world rotates around the user, which triggers more saccades that increase the average gaze velocity for each trial.

Interestingly, we found a *negative* correlation between trial duration and gaze velocity (Subsection 7.4.4). A closer inspection of the data revealed two patterns that we believe explain this negative correlation. First, as the participant reaches the end of a trial, their rotation velocity decreases as they try to avoid turning beyond 90° in the VE (see the plateau towards the end of the trial time in the orange curves in Figure 7.5 (right column) and Figure 7.7). As participants slow down their body rotation to complete the trial, their gaze velocity also decreases (see the tail of the blue curve in Figure 7.7) due to the decreased scene motion (fewer nystagmus movements and more smooth vestibulo–ocular reflex movements). We found that longer trials tend to have longer gaze velocity tails as the participants made small adjustments to their virtual heading in order to complete the trial. The second factor that we believe contributes to the negative correlation between trial duration and gaze velocity is that of previously-documented slow and fast compensatory phases of eye movements during vestibular stimulation [1, 203]. When a person's head motion transitions from still to rotating, there is an initial step change in head rotation velocity. This sudden movement initiates ocular nystagmus that alternates between slow and fast phases: gaze velocity quickly reaches a peak and then slowly decreases exponentially and approaches $0^\circ/\text{s}$ [203]. Indeed, this is the pattern seen in the gaze velocity curve (blue)

in Figure 7.7—gaze velocity starts off slow ($0 - 0.4$ s), quickly peaks to a maximal value ($0.4 - 0.6$ s) shortly after the user begins turning, and is followed by periodic, smaller spikes in gaze velocity ($0.6 - 2.5$ s) which gradually approach $0°/s$ ($2.5 - 4.5$ s). Thus, as trial duration increases, it is logical that the participant's average gaze velocity decreases as their gaze velocity profile decreases exponentially.

We found a positive correlation between gain and blink and saccade frequency, which may indicate an increase in simulator sickness symptoms as users spent more time in the experiment [15, 20, 50, 77]. Finally, gaze velocity and blink frequency were correlated with time spent being redirected in VR (trial number) and trial duration. However, these correlations were *very* small, so they may not be useful signals for users' perception of RDW gains.

7.5.3 Postural Data

In our experiment, we found that participants' postural stability was significantly correlated with the strength of redirection gain applied. The postural instability theory of motion sickness proposes that postural instability is the main cause of motion sickness [160]. Stoffogen et al. showed that postural instability (i.e., increased postural sway) is associated with symptoms of motion sickness [189]. This finding is in line with our result, which showed that as the redirection gain increased (and thus, the user's surroundings become more visually unstable due to additional rotations), participants' postural sway also increased. Stoffogen et al. found that postural sway preceding motion sickness ranged, on average, on the order of 10 cm or less [189], and we found that as the gain increases by one unit, postural sway increases on average by 6.080 cm. This adds support to the notion that postural sway can be used as a signal for feelings of discomfort for users during redirection.

7.5.4 Further Considerations

Although our results showed some correlations between RDW system factors and users' physiological signals, our study design limits the amount of correlations we were able to study. We chose to use the method of constant stimuli (MCS) for our experiment because this topic is somewhat understudied, so we preferred to have a uniform sampling of the RDW gain parameter space. This allowed us to study equally how physiological signals varied across the parameter space, as opposed to an adaptive sampling paradigm which will be biased near to the participant's detection threshold. Since MCS applies gains in a random order, we were unable to test how physiological signals behave near a user's detection threshold since this threshold value is only known after the experiment once we fit a psychometric curve. As such, we were only able to test for general correlations between RDW gain and physiological signals.

Furthermore, we chose a randomized order of gains to mitigate learning effects and to minimize the chance that participants feel debilitating levels of simulator sickness. As a result, it is possible that our data do not capture some patterns in users' physiological data that only manifest after long, continuous exposure to redirection. With a different study design (such as adaptive staircase or continuous psychophysics), one may be able to better study how physiological signals behave near the detection threshold and during uninterrupted periods of walking.

Prior work has shown differences in detection thresholds depending on gender [138, 230], but we did not see any such differences in our work. It is not clear why this is the case. Finally, we note that while we did see significant correlations, the effects were sometimes small; we believe this is due to the short trial durations and because most of the trials applied a gain that was not near the participant's detection threshold.

In this work, we only measured the correlation between redirection strength and gaze/posture data. This should not be mistaken as a correlation between *user comfort* and physiological data. Since we did not continually record any measure of user comfort during the course of the experiment (e.g. after every trial), we cannot claim that there is a direct correlation between physiological signal patterns and user comfort. Since the strength (and perceptibility) of RDW gains is related to users' feelings of comfort [161], it is likely such these correlations exist.

7.6 Conclusions, Limitations, & Future Work

In this work, we investigated correlations between physiological signals (gaze and posture) and RDW gains. We also studied the impact of photopic and mesopic lighting conditions on users' sensitivity to RDW. In line with our first hypothesis, we found that postural stability, gaze velocity, blink and saccade frequency were significantly correlated with various factors of the VR environment during redirection (RDW gain, trial duration, and trial number). Contrary to our second hypothesis, our results showed no significant effect of luminance on RDW thresholds. Our results contribute to the growing body of knowledge on RDW thresholds and factors that affect it, as well as potential signals for RDW tolerance in the form of physiological signals. Our work has some limitations. It is difficult to know exactly which patterns in the physiological data were due to RDW or due to symptoms of simulator sickness that participants will naturally feel as experiment progresses. However, it is common for RDW to increase users' sickness levels, so in practice it may not be important to disentangle the effects of RDW and simulator sickness on physiological signal patterns. Additionally, our participant pool was biased toward young people. Finally, the physiological signals we recorded and analyzed are only a subset of the potentially-available signals that one could record during VR. We limited our study to posture and gaze data

since these are biosignals that are easily obtainable in most modern HMDs, but other signals such as heart rate or skin conductance may be useful correlates as well. Future work should study correlations between RDW parameters and other signals such as galvanic skin response, pupilometry, and heart rate.

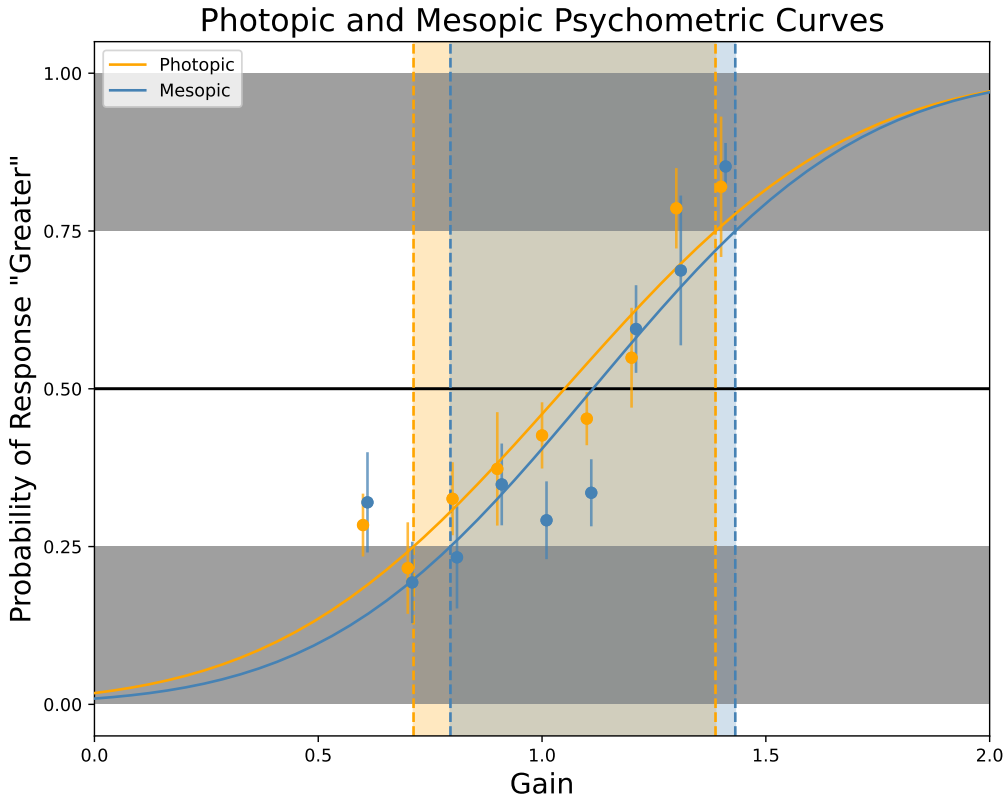


Figure 7.3: Psychometric curves fit to participants' pooled response data for the photopic (yellow) and mesopic (blue) conditions. The graph shows the average probability of responding *"greater"* to the post-trial question *"Was the virtual movement smaller or greater than the physical movement?"*. The yellow- and blue-shaded regions indicate the estimated range of rotation gains that are usually imperceptible to users (i.e., the 25% and 75% detection thresholds). Error bars for each data point denote the standard error. The pooled detection thresholds for photopic and mesopic conditions were similar to values found in prior work that used photopic stimuli, and there were no significant differences between the two conditions. The detection threshold gains shown here are not exactly the same as the average values shown in Table 7.1 since we computed the curves in this plot by fitting a psychometric curve to the *pooled* participant responses, while Table 7.1 computes the average of the curves fit to *individual* participants' responses for each conditions.

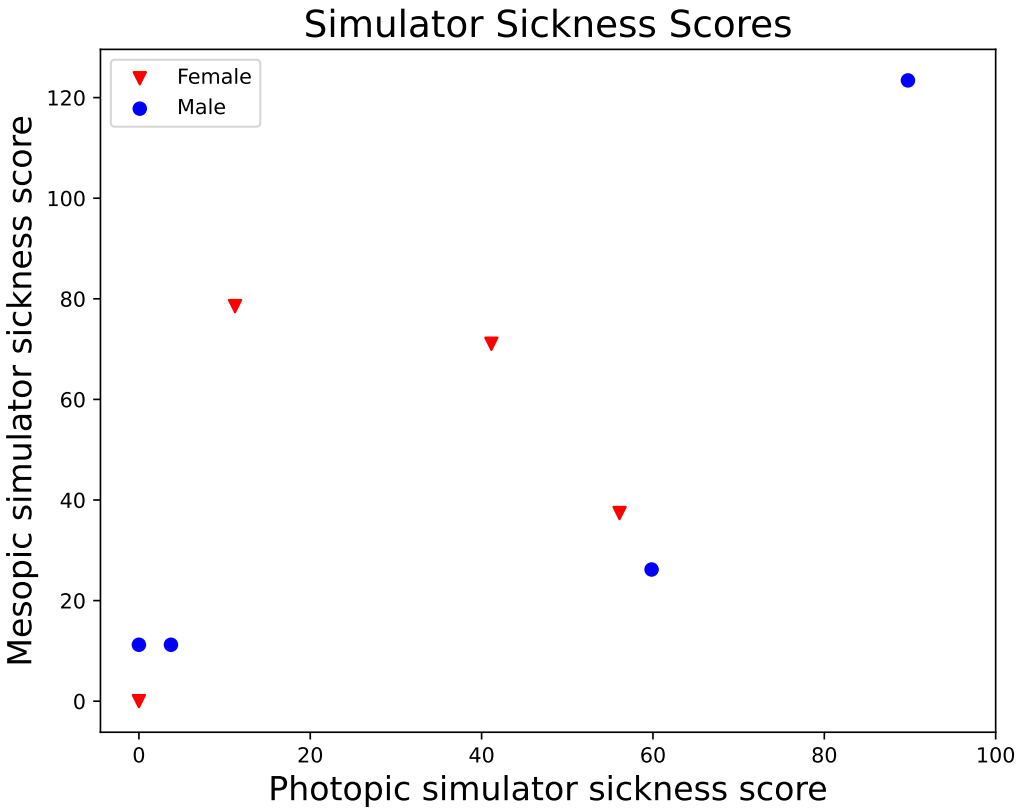


Figure 7.4: A scatter plot of users' SS scores after the light (photopic) and dark (mesopic) blocks of our experiment. In general, participants exhibited SS levels that are typical of RDW detection threshold experiments. The data belonging to the outlier male participant with the highest SS scores did not show any anomalous patterns, so their data are included in our analyses.

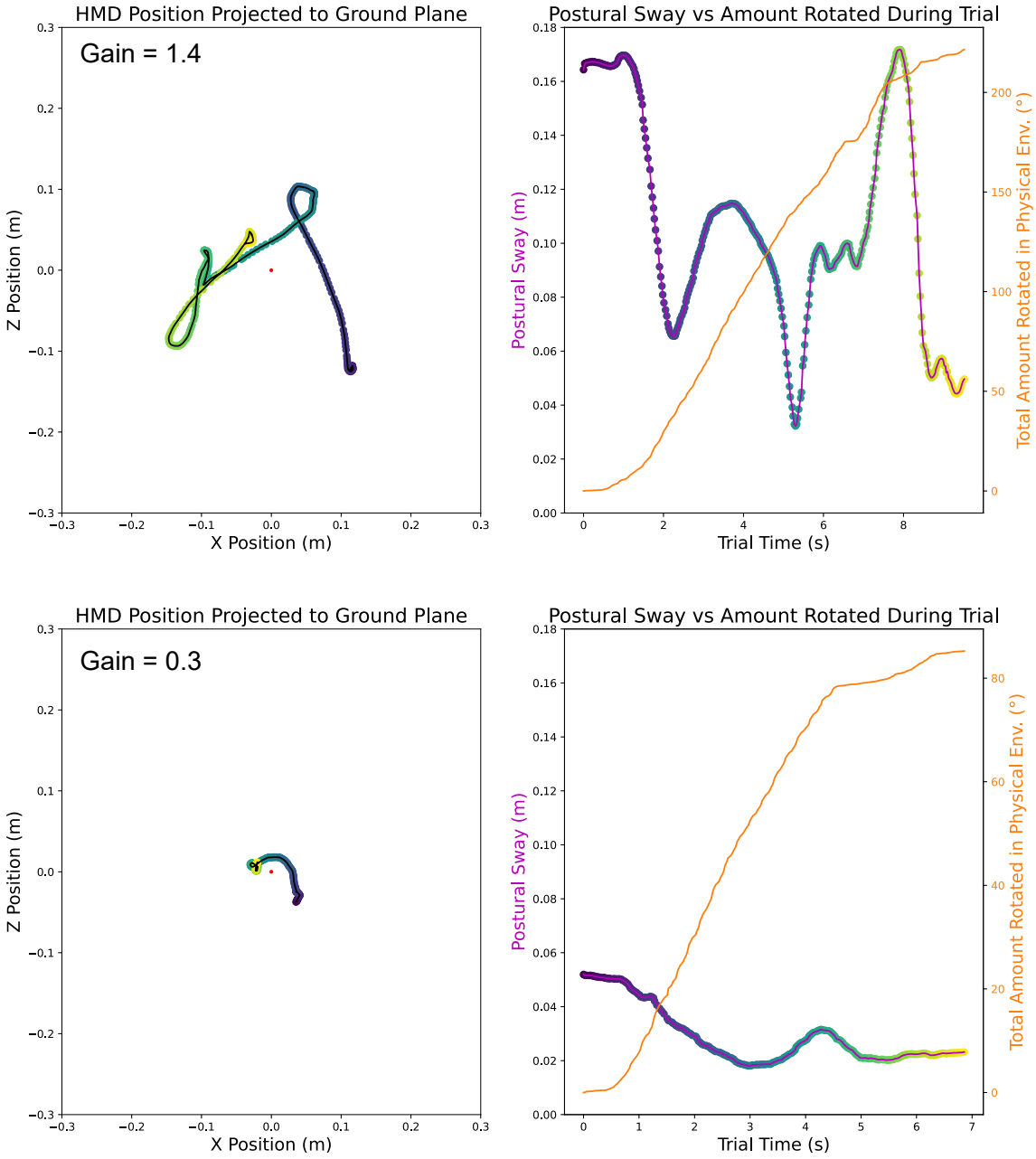


Figure 7.5: Examples of one participant's posture data for two different trials (each row corresponds to one trial). The left column shows the participant's head position projected onto the ground plane (black curve), with the centroid of their positions at the origin (red dot). For each trajectory point sampled, we compute a proxy for postural sway as the distance between the centroid and the sampled head position (i.e., the distance from each point to the origin). The right column shows the participant's postural sway (purple curve) and total amount rotated in the physical environment (orange curve) across the duration of the trial. The points along the trajectory curves and postural sway curves are colored according to the time in the trial (purple indicates the beginning of the trial, yellow indicates the end of the trial). These plots show that as the gain increases, participants' postural sway also increases—a correlation which was statistically significant (Subsection 7.4.3).

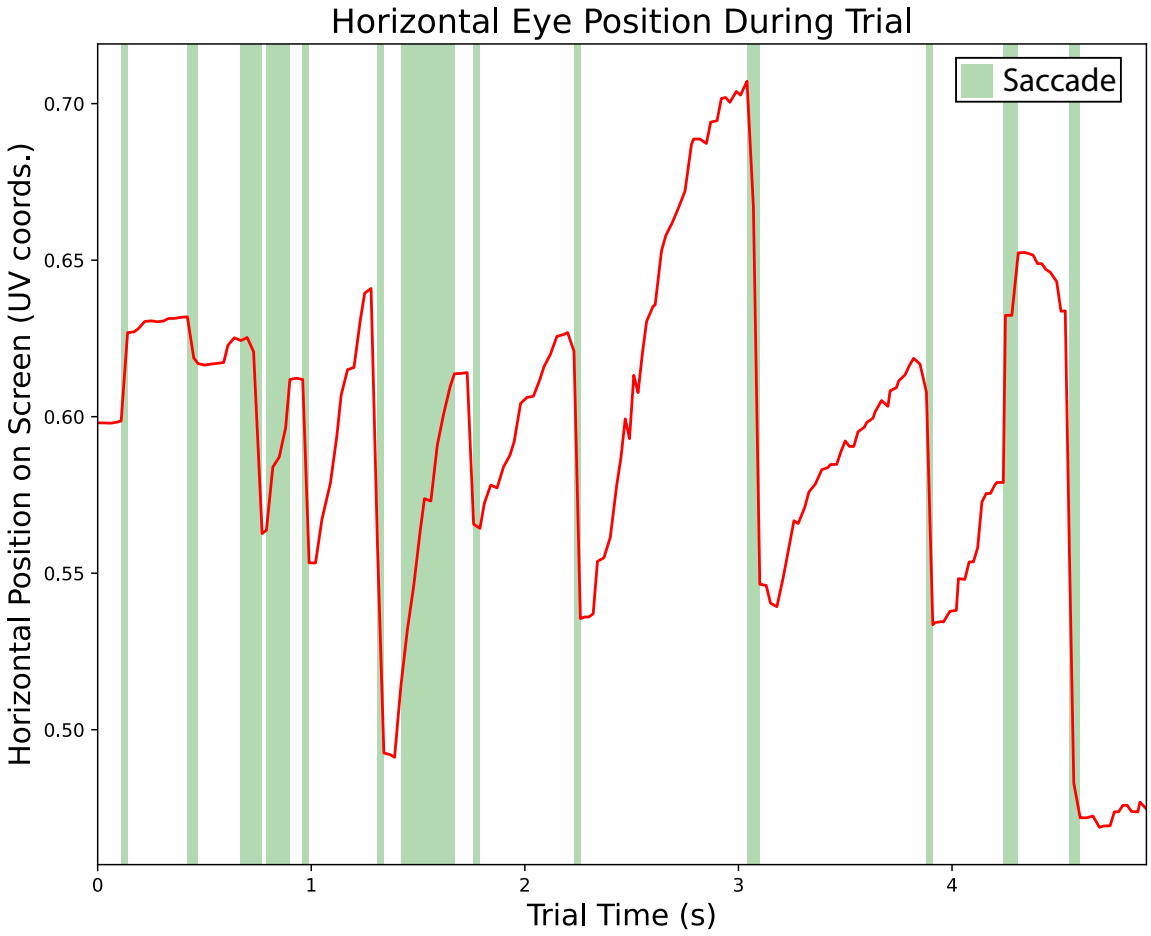


Figure 7.6: An example of one participant's horizontal eye position (red curve, in UV coordinates of the rendered image) during one trial. Green indicate saccades (gaze velocity above $30°$), which are also identifiable as a very steep slope in the red curve, denoting the eye's horizontal position. The data help to confirm that our participants' gaze behavior was free of abnormalities since this plot shows that gaze behavior was characterized by typical nystagmus responses that are expected in healthy observers during head rotation [1].

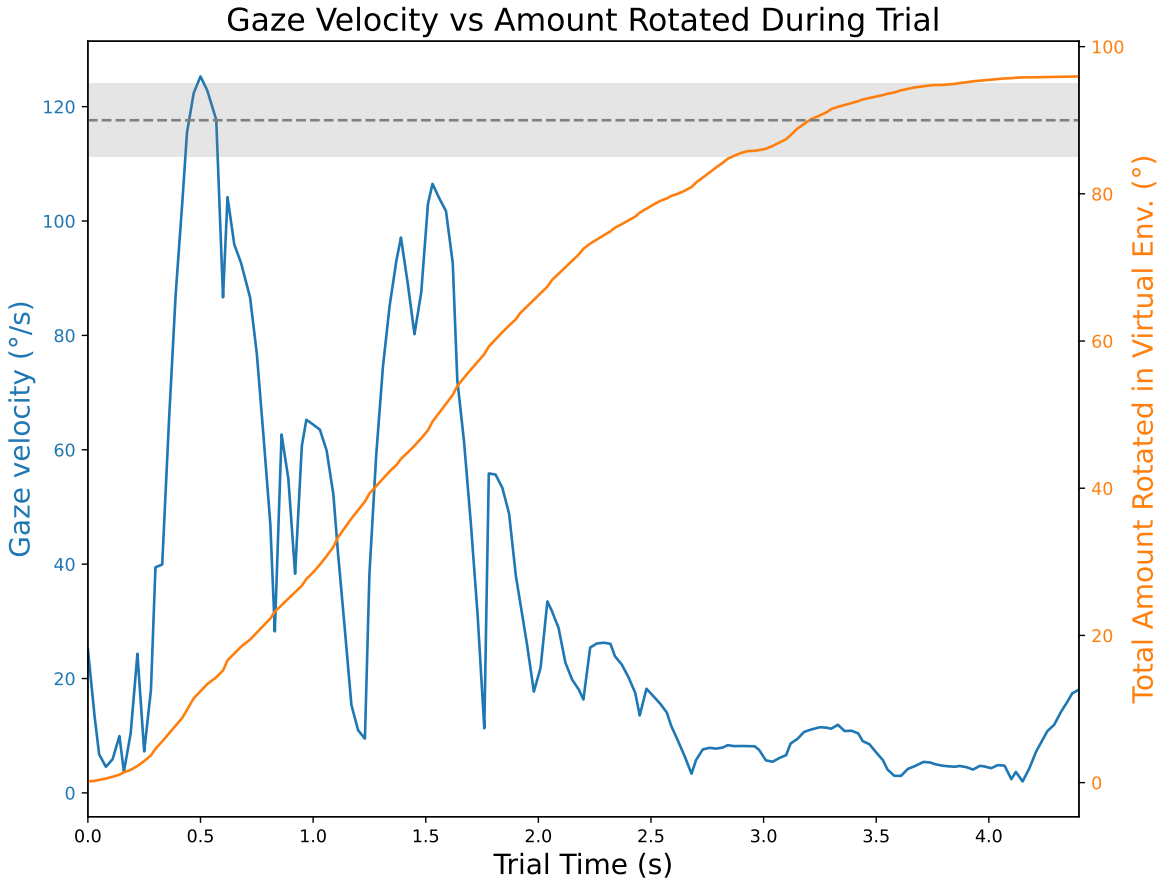


Figure 7.7: A graph showing a user's gaze velocity (blue) compared to the total amount they have rotated their body during one trial (orange). The gaze velocity curve is characterized by multiple saccades that arise from vestibular nystagmus and the optokinetic reflex. The body rotation curve increases from 0° to $\sim 95^\circ$ over the course of the trial. The grey shaded region is the range $85^\circ - 95^\circ$ (the trial ended if the orange curve lies within this region for 1 s). As the trial progresses, the user's gaze velocity gradually decreases to 0° as they wait for the trial to complete after rotating a sufficient amount.

Part IV

Conclusion, Limitations, and Future Work

Chapter 8: Conclusion, Limitations, & Future Work

8.1 Summary of Results

The work presented in this dissertation details new methods for understanding how people locomote naturally in VEs and new locomotion interfaces that enable people to explore large VEs with fewer collisions on average. The individual contributions of the work can be summarized as follows.

Alignment-based Redirection Controller [231]: In this project, we made the key observation that locomotion in VR is a problem primarily when the local structure of the physical and virtual environments are different. Building on this observation, we developed a redirected walking algorithm that steers the user in the physical space to minimize the differences in proximity to objects between the physical and virtual environments, yielding significantly fewer collisions on average. We also introduced a new metric, Complexity Ratio, that measures the relative complexity of a given physical and virtual environment.

Visibility-based Redirection Controller [232]: In this project, we used techniques from robot motion planning to formalize the redirection problem and develop a new redirection algorithm that achieves fewer collisions on average by leveraging visibility polygons to represent the structure of the user's immediate surroundings.

Distractor-based Redirection Controller [235]: In this project, we use motion planning techniques to design an algorithm that determines how to control the behavior of distractors in a virtual

experience such that the user travels on a collision-free path when interacting with the distractor. We showed that if certain information about the physical space is available to the system and the distractors fulfill certain constraints, it is possible to guarantee that the user will travel along a collision-free trajectory when engaging with the distractor in VR.

Environment Compatibility Metric [233]: We used visibility polygons and uniform random sampling techniques to describe the structure of a pair of environments on a local *and* a global scale. We then compare the shapes of visibility polygons between the two environments to quantify how easily navigable the two environments will be when they are used as a physical and virtual environment for a virtual experience.

Physiological Signals of Redirection Sensitivity [236]: In this project, we measured how sensitivity to redirected walking rotation gains changes as a function of photopic and mesopic light levels, and we found no significant differences in sensitivity based on light level. We also measured to what extent a user's gaze and postural data are correlated with the strength of rotation gains. Our results showed that as the rotation gain increases, the user's gaze stability and postural stability decreases.

8.2 Limitations

Our work is not without its limitations, however. First, some of the redirection algorithms presented in this thesis was evaluated only using simulations [231, 232]. Although there is recent work that points to the validity of simulation-based evaluation for redirected walking algorithms [10], there are still many subtle dynamics of human gait that are currently missing with our simulation frameworks. In addition to lacking the ability to simulate subtleties of human gait, our simulation framework also lacks the ability to separately and faithfully model

head and eye rotations (that is, we assume the user always looks directly ahead in the direction they are moving). Relatedly, another limitation of our work is that we did not extensively test the robustness of our algorithms in different environment layouts. Furthermore, the alignment metrics we used in our algorithms [231, 232] are quite simple and it is likely that more expressive similarity metrics would yield better results. Finally, our distractor-based algorithm [235] was only evaluated using one implementation of our framework, so the efficacy of our method with different distractor designs is not well-understood.

Our environment incompatibility metric [233] also has limitations. First, our method slow since it is essentially an exhaustive comparison of all pairs of sampled points in both environments, which is a naïve solution. It is possible that there exist more clever ways to select which points to compare to find the most-compatible pair of physical and virtual points, similar to importance sampling in ray tracing [216, 217].

Our work on physiological signals for redirection sensitivity also has limitations [236]. First, we only studied the relationship between physiological signals and rotation gains in a traditional psychophysical experiment, despite the existence of several other redirection gains [112, 187, 240]. Additionally, our ability to separate the effects of gain and simulator sickness on gaze and posture data is very limited, since we did not collect any continuous measure of users' simulator sickness symptoms during the course of the experiment (i.e., we only collected sickness data at the end of the experiment). Thus, it is possible that some of the observed patterns in gaze and posture data were caused by symptoms of simulator sickness and not directly because of the rotation gains applied.

8.3 Future Work

In this dissertation, we developed methods for analyzing and enabling collision-free natural walking in large virtual environments. Although our work has made contributions towards improving the viability of natural walking in VR, there are still many avenues for future work.

Future work should study the viability of combining multiple different techniques for natural walking. We studied alignment-based methods and distractor-based methods in this thesis. However, we know that different methods for natural walking in VR, such as redirection [158], dynamically-changing virtual spaces [215, 238], and distractors [149], are optimally effective in different environment configurations and virtual applications. For example, alignment-based methods work best when the physical and virtual environments have areas of locally-similar structure, while dynamically-changing virtual spaces are more amenable to unpredictable, dynamic physical environments. Algorithms that are able to combine multiple different natural walking interfaces based on the environment configurations will likely be more effective since they will be able to utilize the optimal technique based on the user's current situation. Additionally, future work should study different methods for manipulating distractor behavior to guide the user to safe physical locations. In our implementation [236], users very reliably interacted with the distractors since they were a core aspect of the virtual experience. However, one could imagine different kinds of distractors that vary in their ability to reliably capture the user's attention and elicit a specific behavior from the user. A locomotion interface that can make use of multiple types of distractors with varying levels of attentional capture would likely yield a more immersive experience for users.

There are many avenues for future work related to our environment incompatibility metric,

too. First, future work should develop new data-driven metrics that use machine learning to learn an environment compatibility metric based on large amounts of locomotion data. Computation of our ENI metric is quite slow, so a learned model that only needs to be trained once and can then be queried quickly for new, unseen environment pairs would be beneficial. A large-scale locomotion dataset would also be useful for fast evaluation of locomotion interfaces, too. Future work should also investigate the possibility of using additional factors besides environment geometry to estimate navigability (e.g., user task, user trajectory, environment complexity).

Additionally, future work should develop methods for measuring a user's tolerance to redirection that directly measure the amount of visual motion perceived by the user (e.g. via optic flow [163] or motion energy models [2]). In our experiment, we measured correlations between rotation gain strength and physiological signals. However, parameterizing motion by a rotation gain is not very descriptive of the motion perceived by the user since the perceived visual motion will depend on not only the gain but also the user's rotation speed as well as the structure (i.e., depth) and appearance (i.e., textures) of their virtual surroundings. Therefore, understanding a more direct relationship between physiological signals and perceived motion (parameterized by optic flow or a motion energy model) would yield an improved understanding of how sensitivity to redirection changes as a function of perceived motion in *any* virtual environment, allowing for greater generalizability and reliability of measures of redirection thresholds. Finally, future work on physiology for redirection sensitivity should test the viability of using physiological data as a real-time proxy for comfort and using the observed patterns in gaze and posture data to modulate the strength of redirection gains *during* the user's virtual experience without interrupting them (as is currently done in psychophysical threshold paradigms).

Bibliography

[1] Richard V Abadi. Mechanisms underlying nystagmus. *Journal of the Royal Society of Medicine*, 95(5):231–234, 2002.

[2] Edward H Adelson and James R Bergen. Spatiotemporal energy models for the perception of motion. *Josa a*, 2(2):284–299, 1985.

[3] Aaron D Ames, Samuel Coogan, Magnus Egerstedt, Gennaro Notomista, Koushil Sreenath, and Paulo Tabuada. Control barrier functions: Theory and applications. In *2019 18th European control conference (ECC)*, pages 3420–3431. IEEE, 2019.

[4] Gary T Anderson and Gang Yang. A proposed measure of environmental complexity for robotic applications. In *2007 IEEE International Conference on Systems, Man and Cybernetics*, pages 2461–2466. IEEE, 2007.

[5] Esther M Arkin, L Paul Chew, Daniel P Huttenlocher, Klara Kedem, and Joseph S Mitchell. An efficiently computable metric for comparing polygonal shapes. Technical report, CORNELL UNIV ITHACA NY, 1991.

[6] Mahdi Azmandian. *Design and evaluation of adaptive redirected walking systems*. PhD thesis, University of Southern California, 2018.

[7] Mahdi Azmandian, Timofey Grechkin, Mark T Bolas, and Evan A Suma. Physical space requirements for redirected walking: How size and shape affect performance. In *ICAT-EGVE*, pages 93–100, 2015.

[8] Mahdi Azmandian, Timofey Grechkin, and Evan Suma Rosenberg. An evaluation of strategies for two-user redirected walking in shared physical spaces. In *2017 IEEE Virtual Reality (VR)*, pages 91–98. IEEE, 2017.

[9] Mahdi Azmandian, Rhys Yahata, Timofey Grechkin, and Evan Suma Rosenberg. Adaptive redirection: A context-aware redirected walking meta-strategy. *IEEE Transactions on Visualization and Computer Graphics*, 28(5):2277–2287, 2022.

[10] Mahdi Azmandian, Rhys Yahata, Timofey Grechkin, Jerald Thomas, and Evan Suma Rosenberg. Validating simulation-based evaluation of redirected walking systems. *IEEE Transactions on Visualization and Computer Graphics*, 28(5):2288–2298, 2022.

[11] Eric R Bachmann, Eric Hodgson, Cole Hoffbauer, and Justin Messinger. Multi-user redirected walking and resetting using artificial potential fields. *IEEE transactions on visualization and computer graphics*, 25(5):2022–2031, 2019.

[12] Dennis R Baltzley, RS Kennedy, KS Berbaum, MG Lilienthal, and DW Gower. The time course of postflight simulator sickness symptoms. *Aviation, Space, and Environmental Medicine*, 60(11):1043–1048, 1989.

[13] Bernard Baumberger, Michelangelo Flückiger, and Martin Roland. Walking in an environment of moving ground texture. *Japanese Psychological Research*, 42(4):238–250, 2000.

[14] Michael L Benedikt. To take hold of space: isovists and isovist fields. *Environment and Planning B: Planning and design*, 6(1):47–65, 1979.

[15] Cebeci Berk, Celikcan Ufuk, and K Capin Tolga. A comprehensive study of the affective and physiological responses induced by dynamic virtual reality environments. comp anim virtual worlds 30 (3–4): e1893, 1893.

[16] Alain Berthoz. *The brain's sense of movement*, volume 10. Harvard University Press, 2000.

[17] Silvia Biasotti, Andrea Cerri, Alex Bronstein, and Michael Bronstein. Recent trends, applications, and perspectives in 3d shape similarity assessment. In *Computer Graphics Forum*, volume 35, pages 87–119. Wiley Online Library, 2016.

[18] Jutta Billino, Frank Bremmer, and Karl R Gegenfurtner. Motion processing at low light levels: Differential effects on the perception of specific motion types. *Journal of Vision*, 8 (3):14–14, 2008.

[19] Luke Bölling, Niklas Stein, Frank Steinicke, and Markus Lappe. Shrinking circles: Adaptation to increased curvature gain in redirected walking. *IEEE transactions on visualization and computer graphics*, 25(5):2032–2039, 2019.

[20] LJG Bouyer and DGD Watt. "torso rotation" experiments; 2: Gaze stability during voluntary head movements improves with adaptation to motion sickness. *Journal of Vestibular Research*, 6(5):377–385, 1996.

[21] Doug A Bowman, David Koller, and Larry F Hodges. A methodology for the evaluation of travel techniques for immersive virtual environments. *Virtual reality*, 3(2):120–131, 1998.

[22] Doug A Bowman, Elizabeth T Davis, Larry F Hodges, and Albert N Badre. Maintaining spatial orientation during travel in an immersive virtual environment. *Presence*, 8(6):618–631, 1999.

[23] Evren Bozgeyikli, Andrew Raij, Srinivas Katkoori, and Rajiv Dubey. Point & teleport locomotion technique for virtual reality. In *Proceedings of the 2016 annual symposium on computer-human interaction in play*, pages 205–216, 2016.

[24] Th Brandt, Johannes Dichgans, and Ellen Koenig. Differential effects of central versus peripheral vision on egocentric and exocentric motion perception. *Experimental brain research*, 16:476–491, 1973.

[25] Bruno Brito, Boaz Floor, Laura Ferranti, and Javier Alonso-Mora. Model predictive contouring control for collision avoidance in unstructured dynamic environments. *IEEE Robotics and Automation Letters*, 4(4):4459–4466, 2019.

[26] Vicki Bruce, Patrick R Green, and Mark A Georgeson. *Visual perception: Physiology, psychology, & ecology*. Psychology Press, 2003.

[27] Gerd Bruder, Frank Steinicke, and Klaus H Hinrichs. Arch-explore: A natural user interface for immersive architectural walkthroughs. 2009.

[28] Antonio Cardone, Satyandra K Gupta, and Mukul Karnik. A survey of shape similarity assessment algorithms for product design and manufacturing applications. *J. Comput. Inf. Sci. Eng.*, 3(2):109–118, 2003.

[29] Yuchen Chang, Keigo Matsumoto, Takuji Narumi, Tomohiro Tanikawa, and Michitaka Hirose. Redirection controller using reinforcement learning. *arXiv preprint arXiv:1909.09505*, 2019.

[30] Yuchen Chang, Keigo Matsumoto, Takuji Narumi, Tomohiro Tanikawa, and Michitaka Hirose. Redirection controller using reinforcement learning. *IEEE Access*, 9:145083–145097, 2021.

[31] Jean-Rémy Chardonnet, Mohammad Ali Mirzaei, and Frédéric Mérienne. Features of the postural sway signal as indicators to estimate and predict visually induced motion sickness in virtual reality. *International Journal of Human–Computer Interaction*, 33(10):771–785, 2017.

[32] Haiwei Chen and Henry Fuchs. Supporting free walking in a large virtual environment: imperceptible redirected walking with an immersive distractor. In *Proceedings of the Computer Graphics International Conference*, pages 1–6, 2017.

[33] Haiwei Chen, Samantha Chen, and Evan Suma Rosenberg. Redirected walking strategies in irregularly shaped and dynamic physical environments. In *25th IEEE Conference on Virtual Reality and 3D User Interfaces (VR 2018). Workshop on Everyday Virtual Reality*, 2018.

[34] Xiaoli Chen, Timothy P McNamara, Jonathan W Kelly, and Thomas Wolbers. Cue combination in human spatial navigation. *Cognitive Psychology*, 95:105–144, 2017.

[35] Lung-Pan Cheng, Eyal Ofek, Christian Holz, and Andrew D Wilson. Vroamer: generating on-the-fly vr experiences while walking inside large, unknown real-world building environments. In *2019 IEEE Conference on Virtual Reality and 3D User Interfaces (VR)*, pages 359–366. IEEE, 2019.

[36] Elizabeth R Chrastil and William H Warren. Active and passive contributions to spatial learning. *Psychonomic bulletin & review*, 19:1–23, 2012.

[37] Elizabeth R Chrastil and William H Warren. Active and passive spatial learning in human navigation: acquisition of survey knowledge. *Journal of experimental psychology: learning, memory, and cognition*, 39(5):1520, 2013.

[38] Mike Christenson. Registering visual permeability in architecture: isovists and occlusion maps in autolisp. *Environment and Planning B: Planning and Design*, 37(6):1128–1136, 2010.

[39] Claudiu Ciumedean, Cristian Patras, Mantas Cibulskis, Norbert Váradi, and Niels Christian Nilsson. Impossible open spaces: Exploring the effects of occlusion on the noticeability of self-overlapping virtual environments. In *2021 IEEE Conference on Virtual Reality and 3D User Interfaces Abstracts and Workshops (VRW)*, pages 389–390. IEEE, 2021.

[40] Susan Valerie Gray Cobb and Sarah Catherine Nichols. Static posture tests for the assessment of postural instability after virtual environment use. *Brain Research Bulletin*, 47(5):459–464, 1998.

[41] Ben J Congdon and Anthony Steed. Sensitivity to rate of change in gains applied by redirected walking. In *Proceedings of the 25th ACM Symposium on Virtual Reality Software and Technology*, pages 1–9, 2019.

[42] Ben J Congdon and Anthony Steed. Monte-carlo redirected walking: Gain selection through simulated walks. *IEEE Transactions on Visualization and Computer Graphics*, 2023.

[43] Ruth Alison Conroy. *Spatial navigation in immersive virtual environments*. PhD thesis, Citeseer, 2001.

[44] Robbe Cools and Adalberto L Simeone. Investigating the effect of distractor interactivity for redirected walking in virtual reality. In *Symposium on Spatial User Interaction*, pages 1–5, 2019.

[45] Jacob W Crandall. *Towards developing effective human-robot systems*. PhD thesis, Brigham Young University. Department of Computer Science, 2003.

[46] Weiwei Dai, Ivan Selesnick, John-Ross Rizzo, Janet Rucker, and Todd Hudson. Detection of normal and slow saccades using implicit piecewise polynomial approximation. *Journal of vision*, 21(6):8–8, 2021.

[47] Ruth Dalton, Christoph Hoelscher, TABITHA Peck, and VIJAY Pawar. Judgments of building complexity and navigability in virtual reality. *Spatial Cognition 2010*, 2010.

[48] Rudolph P Darken and Barry Peterson. Spatial orientation, wayfinding, and representation. In *Handbook of virtual environments*, pages 533–558. CRC Press, 2002.

[49] Mark De Berg, Marc Van Kreveld, Mark Overmars, and Otfried Schwarzkopf. Computational geometry. In *Computational geometry*, pages 1–17. Springer, 1997.

[50] Mark S Dennison, A Zachary Wisti, and Michael D'Zmura. Use of physiological signals to predict cybersickness. *Displays*, 44:42–52, 2016.

[51] Mark Stephen Dennison and Michael D'Zmura. Cybersickness without the wobble: Experimental results speak against postural instability theory. *Applied ergonomics*, 58: 215–223, 2017.

[52] Massimiliano Di Luca, Hasti Seifi, Simon Egan, and Mar Gonzalez-Franco. Locomotion vault: the extra mile in analyzing vr locomotion techniques. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*, pages 1–10, 2021.

[53] Paul DiZio and James R Lackner. Circumventing side effects of immersive virtual environments. In *HCI (2)*, pages 893–896, 1997.

[54] Tianyang Dong, Xianwei Chen, Yifan Song, Wenyuan Ying, and Jing Fan. Dynamic artificial potential fields for multi-user redirected walking. In *2020 IEEE Conference on Virtual Reality and 3D User Interfaces (VR)*, pages 146–154. IEEE, 2020.

[55] Zhi-Chao Dong, Xiao-Ming Fu, Chi Zhang, Kang Wu, and Ligang Liu. Smooth assembled mappings for large-scale real walking. *ACM Transactions on Graphics (TOG)*, 36(6):1–13, 2017.

[56] Zhi-Chao Dong, Xiao-Ming Fu, Zeshi Yang, and Ligang Liu. Redirected smooth mappings for multiuser real walking in virtual reality. *ACM Transactions on Graphics (TOG)*, 38(5): 1–17, 2019.

[57] Zhi-Chao Dong, Wenming Wu, Zenghao Xu, Qi Sun, Guanjie Yuan, Ligang Liu, and Xiao-Ming Fu. Tailored reality: Perception-aware scene restructuring for adaptive vr navigation. *ACM Transactions on Graphics (TOG)*, 40(5):1–15, 2021.

[58] Diderik Jan A Eikema, Jung Hung Chien, Nicholas Stergiou, Sara A Myers, Melissa M Scott-Pandorf, Jacob J Bloomberg, and Mukul Mukherjee. Optic flow improves adaptability of spatiotemporal characteristics during split-belt locomotor adaptation with tactile stimulation. *Experimental brain research*, 234:511–522, 2016.

[59] Haitham El-Hussieny, Samy FM Assal, and Mohamed Abdellatif. Robotic exploration: new heuristic backtracking algorithm, performance evaluation and complexity metric. *International Journal of Advanced Robotic Systems*, 12(4):33, 2015.

[60] Mohamed Elbanhawi and Milan Simic. Sampling-based robot motion planning: A review. *Ieee access*, 2:56–77, 2014.

[61] Paul Emrath. Spaces in new homes. *Natl. Assoc. Home Build. Available*, 2013.

[62] Ajoy S Fernandes and Steven K Feiner. Combating vr sickness through subtle dynamic field-of-view modification. In *3D User Interfaces (3DUI), 2016 IEEE Symposium on*, pages 201–210. IEEE, 2016.

[63] Christopher R Fetsch, Amanda H Turner, Gregory C DeAngelis, and Dora E Angelaki. Dynamic reweighting of visual and vestibular cues during self-motion perception. *Journal of Neuroscience*, 29(49):15601–15612, 2009.

[64] Paolo Fiorini and Zvi Shiller. Motion planning in dynamic environments using velocity obstacles. *The International Journal of Robotics Research*, 17(7):760–772, 1998.

[65] Gerald Franz and Jan M Wiener. Exploring isovist-based correlates of spatial behavior and experience. In *5th International Space Syntax Symposium*, pages 503–517. Techne Press, 2005.

[66] Peizhong Gao, Keigo Matsumoto, Takuji Narumi, and Michitaka Hirose. Visual-auditory redirection: Multimodal integration of incongruent visual and auditory cues for redirected walking. In *2020 IEEE international symposium on mixed and augmented reality (ISMAR)*, pages 639–648. IEEE, 2020.

[67] George A Gescheider. *Psychophysics: the fundamentals*. Psychology Press, 2013.

[68] Rhys Goldstein, Simon Breslav, Kean Walmsley, and Azam Khan. Spaceanalysis: A tool for pathfinding, visibility, and acoustics analyses in generative design workflows. *Proc. SimAUD*, 2020.

[69] Reginald G Golledge et al. *Wayfinding behavior: Cognitive mapping and other spatial processes*. JHU press, 1999.

[70] Ashton Graybiel, Brant Clark, Kenneth MacCorquodale, and Dorothy I Hupp. Role of vestibular nystagmus in the visual perception of a moving target in the dark. *The American Journal of Psychology*, pages 259–266, 1946.

[71] Timofey Grechkin, Jerald Thomas, Mahdi Azmandian, Mark Bolas, and Evan Suma. Revisiting detection thresholds for redirected walking: Combining translation and curvature gains. In *Proceedings of the ACM Symposium on Applied Perception*, pages 113–120, 2016.

[72] ED Grossman and R Blake. Perception of coherent motion, biological motion and form-from-motion under dim-light conditions. *Vision research*, 39(22):3721–3727, 1999.

[73] Leonidas J Guibas, Rajeev Motwani, and Prabhakar Raghavan. The robot localization problem. In *Proc. 1st Workshop on Algorithmic Foundations of Robotics*, pages 269–282. Citeseer, 1995.

[74] Xuanru Guo, Shinji Nakamura, Yoshitaka Fujii, Takeharu Seno, and Stephen Palmisano. Effects of luminance contrast, averaged luminance and spatial frequency on vection. *Experimental Brain Research*, 239:3507–3525, 2021.

[75] Saif Haq and Craig Zimring. Just down the road a piece: The development of topological knowledge of building layouts. *Environment and behavior*, 35(1):132–160, 2003.

[76] Peter E Hart, Nils J Nilsson, and Bertram Raphael. A formal basis for the heuristic determination of minimum cost paths. *IEEE transactions on Systems Science and Cybernetics*, 4(2):100–107, 1968.

[77] Wanja Hemmerich, Behrang Keshavarz, and Heiko Hecht. Visually induced motion sickness on the horizon. *Frontiers in Virtual Reality*, 1:582095, 2020.

[78] Lawrence J Hettinger, Kevin S Berbaum, Robert S Kennedy, William P Dunlap, and Margaret D Nolan. Vection and simulator sickness. *Military Psychology*, 2(3):171–181, 1990.

[79] Bill Hillier. Hanson the social logic of space, 1984.

[80] Bill Hillier, Adrian Leaman, Paul Stansall, and Michael Bedford. Space syntax. *Environment and Planning B: Planning and design*, 3(2):147–185, 1976.

[81] Christian Hirt, Yves Kompis, Christian Holz, and Andreas Kunz. The chaotic behavior of redirection–revisiting simulations in redirected walking. In *2022 IEEE Conference on Virtual Reality and 3D User Interfaces (VR)*, pages 524–533. IEEE, 2022.

[82] Rumi Hisakata and Ikuya Murakami. The effects of eccentricity and retinal illuminance on the illusory motion seen in a stationary luminance gradient. *Vision research*, 48(19): 1940–1948, 2008.

[83] Eric Hodgson and Eric Bachmann. Comparing four approaches to generalized redirected walking: Simulation and live user data. *IEEE transactions on visualization and computer graphics*, 19(4):634–643, 2013.

[84] Eric Hodgson, Eric Bachmann, and David Waller. Redirected walking to explore virtual environments: Assessing the potential for spatial interference. *ACM Transactions on Applied Perception (TAP)*, 8(4):1–22, 2008.

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

[86] John H Hollman, Robert H Brey, Tami J Bang, and Kenton R Kaufman. Does walking in a virtual environment induce unstable gait?: An examination of vertical ground reaction forces. *Gait & posture*, 26(2):289–294, 2007.

[87] Yukai Hoshikawa, Kazuyuki Fujita, Kazuki Takashima, Morten Fjeld, and Yoshifumi Kitamura. Redirecteddoors: Redirection while opening doors in virtual reality. In *2022 IEEE Conference on Virtual Reality and 3D User Interfaces (VR)*, pages 464–473. IEEE, 2022.

[88] Zien Huang and D Sam Schwarzkopf. A simple statistical framework for small sample studies. *bioRxiv*, pages 2023–09, 2023.

[89] J Hulk and F Rempt. Vertical optokinetic sensations by limited stimulation of the peripheral field of vision. *Ophthalmologica*, 186(2):97–103, 1983.

[90] Timothy Hurt, Eric Greenwald, Sara Allan, Matthew A Cannady, Ari Krakowski, Lauren Brodsky, Melissa A Collins, Ryan Montgomery, and Rena Dorph. The computational thinking for science (ct-s) framework: Operationalizing ct-s for k–12 science education researchers and educators. *International Journal of STEM Education*, 10(1):1, 2023.

[91] Daniel P Huttenlocher, Gregory A. Klanderman, and William J Rucklidge. Comparing images using the hausdorff distance. *IEEE Transactions on pattern analysis and machine intelligence*, 15(9):850–863, 1993.

[92] Courtney Hutton, Shelby Ziccardi, Julio Medina, and Evan Suma Rosenberg. Individualized calibration of rotation gain thresholds for redirected walking. In *ICAT-EGVE*, pages 61–64, 2018.

[93] Brent Edward Insko, M Meehan, M Whitton, and F Brooks. *Passive haptics significantly enhances virtual environments*. PhD thesis, University of North Carolina at Chapel Hill, 2001.

[94] Victoria Interrante, Brian Ries, and Lee Anderson. Seven league boots: A new metaphor for augmented locomotion through moderately large scale immersive virtual environments. In *2007 IEEE Symposium on 3D User interfaces*. IEEE, 2007.

[95] Atiqul Islam, Jinshuai Ma, Tom Gedeon, Md Zakir Hossain, and Ying-Hsang Liu. Measuring user responses to driving simulators: A galvanic skin response based study. In *2019 IEEE International Conference on Artificial Intelligence and Virtual Reality (AIVR)*, pages 33–337. IEEE, 2019.

[96] AC Ivy. The physiology of vestibular nystagmus. *Archives of Otolaryngology*, 9(2):123–134, 1929.

[97] Hiroo Iwata. The torus treadmill: Realizing locomotion in ves. *IEEE Computer Graphics and Applications*, 19(6):30–35, 1999.

[98] Hiroo Iwata, Hiroaki Yano, Hiroyuki Fukushima, and Haruo Noma. Circulafloor: A locomotion interface using circulation of movable tiles. In *null*, pages 223–230. IEEE, 2005.

[99] Hiroo Iwata, Hiroaki Yano, and Hiroshi Tomioka. Powered shoes. In *ACM SIGGRAPH 2006 Emerging technologies*, page 28. ACM, 2006.

[100] Omar Janeh, Eike Langbehn, Frank Steinicke, Gerd Bruder, Alessandro Gulberti, and Monika Poetter-Nerger. Walking in virtual reality: Effects of manipulated visual self-motion on walking biomechanics. *ACM Transactions on Applied Perception (TAP)*, 14(2): 1–15, 2017.

[101] Jason Jerald. *The VR book: Human-centered design for virtual reality*. Morgan & Claypool, 2015.

[102] Sertac Karaman and Emilio Frazzoli. Sampling-based algorithms for optimal motion planning. *The international journal of robotics research*, 30(7):846–894, 2011.

[103] Jonathan W Kelly, Bernhard Riecke, Jack M Loomis, and Andrew C Beall. Visual control of posture in real and virtual environments. *Perception & psychophysics*, 70:158–165, 2008.

[104] Robert S Kennedy, Norman E Lane, Kevin S Berbaum, and Michael G Lilienthal. Simulator sickness questionnaire: An enhanced method for quantifying simulator sickness. *The international journal of aviation psychology*, 3(3):203–220, 1993.

[105] Oussama Khatib. Real-time obstacle avoidance for manipulators and mobile robots. In *Autonomous robot vehicles*, pages 396–404. Springer, 1986.

[106] Dooyoung Kim, Jinwook Kim, Jae-eun Shin, Boram Yoon, Jeongmi Lee, and Woontack Woo. Effects of virtual room size and objects on relative translation gain thresholds in redirected walking. In *2022 IEEE conference on virtual reality and 3D user interfaces (VR)*, pages 379–388. IEEE, 2022.

[107] Luv Kohli, Eric Burns, Dorian Miller, and Henry Fuchs. Combining passive haptics with redirected walking. In *Proceedings of the 2005 international conference on Augmented tele-existence*, pages 253–254, 2005.

[108] Eugenia M Kolasinski. *Simulator sickness in virtual environments*, volume 1027. US Army Research Institute for the Behavioral and Social Sciences, 1995.

[109] Eric Krokos and Amitabh Varshney. Quantifying vr cybersickness using eeg. *Virtual Reality*, 26(1):77–89, 2022.

[110] Sandip D Kulkarni, Mark A Minor, Eric R Pardyjak, and John M Hollerbach. Combined wind speed and angle control in a virtual environment using a static observer. In *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1005–1010. IEEE, 2008.

[111] Eike Langbehn and Frank Steinicke. Redirected walking in virtual reality. *Encyclopedia of Computer Graphics and Games. Springer International Publishing*, 2018.

[112] Eike Langbehn, Paul Lubos, Gerd Bruder, and Frank Steinicke. 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, 2017.

[113] Eike Langbehn, Frank Steinicke, Markus Lappe, Gregory F Welch, and Gerd Bruder. In the blink of an eye: leveraging blink-induced suppression for imperceptible position and orientation redirection in virtual reality. *ACM Transactions on Graphics (TOG)*, 37(4): 1–11, 2018.

[114] Markus Lappe, Frank Bremmer, and Albert V van den Berg. Perception of self-motion from visual flow. *Trends in cognitive sciences*, 3(9):329–336, 1999.

[115] Steven M LaValle. *Planning algorithms*. Cambridge university press, 2006.

[116] Joseph J LaViola Jr, Ernst Kruijff, Ryan P McMahan, Doug Bowman, and Ivan P Poupyrev. *3D user interfaces: theory and practice*. Addison-Wesley Professional, 2017.

[117] Dang-Yang Lee, Yang-Hun Cho, and In-Kwan Lee. Real-time optimal planning for redirected walking using deep q-learning. In *2019 IEEE Conference on Virtual Reality and 3D User Interfaces (VR)*, pages 63–71. IEEE, 2019.

[118] David N Lee and JR Lishman. Visual proprioceptive control of stance. *Journal of human movement studies*, 1975.

[119] Dong-Yong Lee, Yong-Hun Cho, Dae-Hong Min, and In-Kwon Lee. Optimal planning for redirected walking based on reinforcement learning in multi-user environment with irregularly shaped physical space. In *2020 IEEE Conference on Virtual Reality and 3D User Interfaces (VR)*, pages 155–163. IEEE, 2020.

[120] Juyoung Lee, Sang Chul Ahn, and Jae-In Hwang. A walking-in-place method for virtual reality using position and orientation tracking. *Sensors*, 18(9):2832, 2018.

[121] Rebekka Lencer, Andreas Sprenger, and Peter Trillenberg. Smooth eye movements in humans: smooth pursuit, optokinetic nystagmus and vestibular ocular reflex. *Eye movement research: An introduction to its scientific foundations and applications*, pages 117–163, 2019.

[122] Yi-Jun Li, Frank Steinicke, and Miao Wang. A comprehensive review of redirected walking techniques: Taxonomy, methods, and future directions. *Journal of Computer Science and Technology*, 37(3):561–583, 2022.

[123] JJ-W Lin, Henry Been-Lirn Duh, Donald E Parker, Habib Abi-Rached, and Thomas A Furness. Effects of field of view on presence, enjoyment, memory, and simulator sickness in a virtual environment. In *Proceedings ieee virtual reality 2002*, pages 164–171. IEEE, 2002.

[124] Mingfeng Lin, Henry C Lucas Jr, and Galit Shmueli. Research commentary—too big to fail: large samples and the p-value problem. *Information Systems Research*, 24(4):906–917, 2013.

[125] Robert W Lindeman, John L Sibert, and James K Hahn. Hand-held windows: towards effective 2d interaction in immersive virtual environments. In *Proceedings IEEE Virtual Reality (Cat. No. 99CB36316)*, pages 205–212. IEEE, 1999.

[126] Massimiliano di Luca, Hasti Seifi, Simon Egan, and Mar Gonzalez Franco. Locomotion vault: the extra mile in analyzing vr locomotion techniques. In *ACM CHI*, May 2021. URL `https://www.microsoft.com/en-us/research/publication/locomotion-vault-the-extra-mile-in-analyzing-vr-locomotion-techniques`

[127] Hope Lutwak, T Scott Murdison, and Kevin W Rio. User self-motion modulates the perceptibility of jitter for world-locked objects in augmented reality. In *2023 IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, pages 346–355. IEEE, 2023.

[128] Patrick Mair and Rand Wilcox. Robust statistical methods in r using the wrs2 package. *Behavior research methods*, pages 1–25, 2019.

[129] Nathan Matsuda, Alex Chapiro, Yang Zhao, Clinton Smith, Romain Bachy, and Douglas Lanman. Realistic luminance in vr. In *SIGGRAPH Asia 2022 Conference Papers*, pages 1–8, 2022.

[130] Jonathan Samir Matthis, Karl S Muller, Kathryn L Bonnen, and Mary M Hayhoe. Retinal optic flow during natural locomotion. *PLOS Computational Biology*, 18(2):e1009575, 2022.

[131] Michael Meehan, Brent Insko, Mary Whitton, and Frederick P Brooks Jr. Physiological measures of presence in stressful virtual environments. *Acm transactions on graphics (tog)*, 21(3):645–652, 2002.

[132] Justin Messinger, Eric Hodoson, and Eric R Bachmann. Effects of tracking area shape and size on artificial potential field redirected walking. In *2019 IEEE Conference on Virtual Reality and 3D User Interfaces (VR)*, pages 72–80. IEEE, 2019.

[133] Daniel R Montello. *Navigation.* Cambridge University Press, 2005.

[134] Atsuo Murata. Effects of duration of immersion in a virtual reality environment on postural stability. *International Journal of Human-Computer Interaction*, 17(4):463–477, 2004.

[135] Shinji Nakamura, Takeharu Seno, Hiroyuki Ito, and Shoji Sunaga. Effects of dynamic luminance modulation on visually induced self-motion perception: observers' perception of illumination is important in perceiving self-motion. *Perception*, 42(2):153–162, 2013.

[136] Thomas Nescher, Ying-Yin Huang, and Andreas Kunz. Planning redirection techniques for optimal free walking experience using model predictive control. In *2014 IEEE Symposium on 3D User Interfaces (3DUI)*, pages 111–118. IEEE, 2014.

[137] Christian T Neth, Jan L Souman, David Engel, Uwe Kloos, Heinrich H Bulthoff, and Betty J Mohler. Velocity-dependent dynamic curvature gain for redirected walking. *IEEE transactions on visualization and computer graphics*, 18(7):1041–1052, 2012.

[138] Anh Nguyen, Yannick Rothacher, Bigna Lenggenhager, Peter Brugger, and Andreas Kunz. Individual differences and impact of gender on curvature redirection thresholds. In *Proceedings of the 15th ACM Symposium on Applied Perception*, page 5. ACM, 2018.

[139] Anh Nguyen, Yannick Rothacher, Bigna Lenggenhager, Peter Brugger, and Andreas Kunz. Effect of sense of embodiment on curvature redirected walking thresholds. In *ACM Symposium on Applied Perception 2020*, pages 1–5, 2020.

[140] Niels Christian Nilsson, Tabitha Peck, Gerd Bruder, Eri Hodgson, Stefania Serafin, Mary Whitton, Frank Steinicke, and Evan Suma Rosenberg. 15 years of research on redirected walking in immersive virtual environments. *IEEE computer graphics and applications*, 38 (2):44–56, 2018.

[141] Jason Orlosky, Brandon Huynh, and Tobias Hollerer. Using eye tracked virtual reality to classify understanding of vocabulary in recall tasks. In *2019 IEEE International Conference on Artificial Intelligence and Virtual Reality (AIVR)*, pages 66–667. IEEE, 2019.

[142] Robert Osada, Thomas Funkhouser, Bernard Chazelle, and David Dobkin. Shape distributions. *ACM Transactions on Graphics (TOG)*, 21(4):807–832, 2002.

[143] Lucy Owen, Jonathan Browder, Benjamin Letham, Gideon Stocek, Chase Tymms, and Michael Shvartsman. Adaptive nonparametric psychophysics. *arXiv preprint arXiv:2104.09549*, 2021.

[144] Chonhyon Park, Jia Pan, and Dinesh Manocha. Itomp: Incremental trajectory optimization for real-time replanning in dynamic environments. In *Twenty-Second International Conference on Automated Planning and Scheduling*, 2012.

[145] Aftab E Patla. Understanding the roles of vision in the control of human locomotion. *Gait & posture*, 5(1):54–69, 1997.

[146] Tabitha C Peck. *Redirected free exploration with distractors: A large-scale real-walking locomotion interface*. PhD thesis, The University of North Carolina at Chapel Hill, 2010.

[147] Tabitha C Peck, Henry Fuchs, and Mary C Whitton. Evaluation of reorientation techniques and distrators for walking in large virtual environments. *IEEE Transactions on Visualization and Computer Graphics*, 15(3):383, 2009.

[148] Tabitha C Peck, Henry Fuchs, and Mary C Whitton. Improved redirection with distractors: A large-scale-real-walking locomotion interface and its effect on navigation in virtual environments. In *Virtual Reality Conference (VR), 2010 IEEE*, pages 35–38. IEEE, 2010.

[149] Tabitha C Peck, Henry Fuchs, and Mary C Whitton. An evaluation of navigational ability comparing redirected free exploration with distractors to walking-in-place and joystick locomotion interfaces. In *Virtual Reality Conference (VR), 2011 IEEE*, pages 55–62. IEEE, 2011.

[150] Tabitha C Peck, Laura E Sockol, and Sarah M Hancock. Mind the gap: The underrepresentation of female participants and authors in virtual reality research. *IEEE transactions on visualization and computer graphics*, 26(5):1945–1954, 2020.

[151] John Peponis, Craig Zimring, and Yoon Kyung Choi. Finding the building in wayfinding. *Environment and behavior*, 22(5):555–590, 1990.

[152] Michael I Posner, Mary J Nissen, and Raymond M Klein. Visual dominance: an information-processing account of its origins and significance. *Psychological review*, 83 (2):157, 1976.

[153] Thomas Prokop, Martin Schubert, and Wiltrud Berger. Visual influence on human locomotion modulation to changes in optic flow: Modulation to changes in optic flow. *Experimental brain research*, 114:63–70, 1997.

[154] Eric D Ragan, Doug A Bowman, Regis Kopper, Cheryl Stinson, Siroberto Scerbo, and Ryan P McMahan. Effects of field of view and visual complexity on virtual reality training effectiveness for a visual scanning task. *IEEE transactions on visualization and computer graphics*, 21(7):794–807, 2015.

[155] Carolyn Ranti, Warren Jones, Ami Klin, and Sarah Shultz. Blink rate patterns provide a reliable measure of individual engagement with scene content. *Scientific reports*, 10(1): 8267, 2020.

[156] Nathan Ratliff, Matt Zucker, J Andrew Bagnell, and Siddhartha Srinivasa. Chomp: Gradient optimization techniques for efficient motion planning. In *2009 IEEE International Conference on Robotics and Automation*, pages 489–494. IEEE, 2009.

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

[158] Sharif Razzaque, Zachariah Kohn, and Mary C Whitton. Redirected walking. In *Proceedings of EUROGRAPHICS*, volume 9, pages 105–106. Citeseer, 2001.

[159] Nicholas Rewkowski, Atul Rungta, Mary Whitton, and Ming Lin. Evaluating the effectiveness of redirected walking with auditory distractors for navigation in virtual environments. In *2019 IEEE Conference on Virtual Reality and 3D User Interfaces (VR)*, pages 395–404. IEEE, 2019.

[160] Gary E Riccio and Thomas A Stoffregen. An ecological theory of motion sickness and postural instability. *Ecological psychology*, 3(3):195–240, 1991.

[161] Michael Rietzler, Jan Gugenheimer, Teresa Hirzle, Martin Deubzer, Eike Langbehn, and Enrico Rukzio. Rethinking redirected walking: On the use of curvature gains beyond perceptual limitations and revisiting bending gains. In *2018 IEEE international symposium on mixed and augmented reality (ISMAR)*, pages 115–122. IEEE, 2018.

[162] Roy A Ruddle and Simon Lessels. The benefits of using a walking interface to navigate virtual environments. *ACM Transactions on Computer-Human Interaction (TOCHI)*, 16 (1):1–18, 2009.

[163] Mohammad R Saeedpour-Parizi, Niall L Williams, Tim Wong, Phillip Guan, Dinesh Manocha, and Ian M Erkelens. Perceptual Thresholds for Radial Optic Flow Distortion in Near-Eye Stereoscopic Displays. *IEEE Transactions on Visualization and Computer Graphics*, 30(5):2570–2579, 2024.

[164] Giovagnoli Sara, Pansell Tony, Bolzani Roberto, Hellgren Kerstin, and Benassi Mariagrazia. The effect of luminance condition on form, form-from-motion and motion perception. *pathways*, 8:11, 2017.

[165] G Schweigart, T Mergner, I Evdokimidis, S Morand, and W Becker. Gaze stabilization by optokinetic reflex (okr) and vestibulo-ocular reflex (vor) during active head rotation in man. *Vision research*, 37(12):1643–1652, 1997.

[166] Lior Shapira and Daniel Freedman. Reality skins: Creating immersive and tactile virtual environments. In *2016 IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, pages 115–124. IEEE, 2016.

[167] Albulena Shaqiri, Maya Roinishvili, Lukasz Grzeczkowski, Eka Chkonia, Karin Pilz, Christine Mohr, Andreas Brand, Marina Kunchulia, and Michael H Herzog. Sex-related differences in vision are heterogeneous. *Scientific reports*, 8(1):1–10, 2018.

[168] Corey S Shayman, Jeanine K Stefanucci, Peter C Fino, and Sarah H Creem-Regehr. Multisensory cue combination during navigation: lessons learned from replication in real and virtual environments. In *2022 IEEE International Symposium on Mixed and Augmented Reality Adjunct (ISMAR-Adjunct)*, pages 276–277. IEEE, 2022.

[169] Dylan A Shell and Maja J Mataric. Human motion-based environment complexity measures for robotics. In *Proceedings 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2003)(Cat. No. 03CH37453)*, volume 3, pages 2559–2564. IEEE, 2003.

[170] Jonathan Richard Shewchuk. Triangle: Engineering a 2d quality mesh generator and delaunay triangulator. In *Workshop on Applied Computational Geometry*, pages 203–222. Springer, 1996.

[171] Angela B Shiflet and George W Shiflet. *Introduction to computational science: modeling and simulation for the sciences*. Princeton University Press, 2014.

[172] I Siegler, Isabelle Viaud-Delmon, Isabelle Israel, and Alain Berthoz. Self-motion perception during a sequence of whole-body rotations in darkness. *Experimental brain research*, 134(1):66–73, 2000.

[173] Adalberto L Simeone, Eduardo Velloso, and Hans Gellersen. Substitutional reality: Using the physical environment to design virtual reality experiences. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*, pages 3307–3316, 2015.

[174] Adalberto L Simeone, Niels Christian Nilsson, André Zenner, Marco Speicher, and Florian Daiber. The space bender: Supporting natural walking via overt manipulation of the virtual environment. In *2020 IEEE Conference on Virtual Reality and 3D User Interfaces (VR)*, pages 598–606. IEEE, 2020.

[175] Mark Simpson, Kai-Florian Richter, Jan Oliver Wallgrün, and Alexander Klippel. Quantifying space, understanding minds: A visual summary approach. *Journal of Spatial Information Science*, (14):95–136, 2017.

[176] Richard Skarbez, Frederick P Brooks, Jr, and Mary C Whitton. A survey of presence and related concepts. *ACM Computing Surveys (CSUR)*, 50(6):1–39, 2017.

[177] Richard T Skarbez. *Plausibility illusion in virtual environments*. PhD thesis, The University of North Carolina at Chapel Hill, 2016.

[178] Mel Slater and Sylvia Wilbur. A framework for immersive virtual environments (five): Speculations on the role of presence in virtual environments. *Presence: Teleoperators & Virtual Environments*, 6(6):603–616, 1997.

[179] Mel Slater, Martin Usoh, and Anthony Steed. Taking steps: the influence of a walking technique on presence in virtual reality. *ACM Transactions on Computer-Human Interaction (TOCHI)*, 2(3):201–219, 1995.

[180] Philip L Smith and Daniel R Little. Small is beautiful: In defense of the small-n design. *Psychonomic bulletin & review*, 25:2083–2101, 2018.

[181] Fabian Soffel, Markus Zank, and Andreas Kunz. Postural stability analysis in virtual reality using the htc vive. In *Proceedings of the 22nd ACM Conference on Virtual Reality Software and Technology*, pages 351–352, 2016.

[182] Misha Sra, Sergio Garrido-Jurado, and Pattie Maes. Oasis: Procedurally generated social virtual spaces from 3d scanned real spaces. *IEEE transactions on visualization and computer graphics*, 24(12):3174–3187, 2017.

[183] Misha Sra, Xuhai Xu, Aske Mottelson, and Pattie Maes. Vmotion: designing a seamless walking experience in vr. In *Proceedings of the 2018 Designing Interactive Systems Conference*, pages 59–70, 2018.

[184] Arthur E Stamps III. Mystery, complexity, legibility and coherence: A meta-analysis. *Journal of environmental psychology*, 24(1):1–16, 2004.

[185] Arthur E Stamps III. Isovists, enclosure, and permeability theory. *Environment and Planning B: Planning and Design*, 32(5):735–762, 2005.

[186] Barry E Stein, Terrence R Stanford, and Benjamin A Rowland. The neural basis of multisensory integration in the midbrain: its organization and maturation. *Hearing research*, 258(1-2):4–15, 2009.

[187] Frank Steinicke, Gerd Bruder, Jason Jerald, Harald Frenz, and Markus Lappe. Estimation of detection thresholds for redirected walking techniques. *IEEE transactions on visualization and computer graphics*, 16(1):17–27, 2009.

[188] Frank Steinicke, Yon Visell, Jennifer Campos, and Anatole Lécuyer. *Human walking in virtual environments*, volume 2. Springer, 2013.

[189] Thomas A Stoffregen and L James Smart Jr. Postural instability precedes motion sickness. *Brain research bulletin*, 47(5):437–448, 1998.

[190] Thomas A Stoffregen, Elise Faugloire, Ken Yoshida, Moira B Flanagan, and Omar Merhi. Motion sickness and postural sway in console video games. *Human factors*, 50(2):322–331, 2008.

[191] Thomas A Stoffregen, Ken Yoshida, Sebastien Villard, Lesley Scibora, and Benoît G Bardy. Stance width influences postural stability and motion sickness. *Ecological Psychology*, 22(3):169–191, 2010.

[192] Ryan R Strauss, Raghuram Ramanujan, Andrew Becker, and Tabitha C Peck. A steering algorithm for redirected walking using reinforcement learning. *IEEE Transactions on Visualization and Computer Graphics*, 26(5):1955–1963, 2020.

[193] Evan Suma, Samantha Finkelstein, Myra Reid, Sabarish Babu, Amy Ulinski, and Larry F Hodges. Evaluation of the cognitive effects of travel technique in complex real and virtual environments. *IEEE Transactions on Visualization and Computer Graphics*, 16(4):690–702, 2009.

[194] Evan A Suma, Seth Clark, David Krum, Samantha Finkelstein, Mark Bolas, and Zachary Warte. Leveraging change blindness for redirection in virtual environments. In *2011 IEEE Virtual Reality Conference*, pages 159–166. IEEE, 2011.

[195] Evan A Suma, Gerd Bruder, Frank Steinicke, David M Krum, and Mark Bolas. A taxonomy for deploying redirection techniques in immersive virtual environments. In *2012 IEEE Virtual Reality Workshops (VRW)*, pages 43–46. IEEE, 2012.

[196] Evan A Suma, Zachary Lipps, Samantha Finkelstein, David M Krum, and Mark Bolas. Impossible spaces: Maximizing natural walking in virtual environments with self-overlapping architecture. *IEEE Transactions on Visualization and Computer Graphics*, 18(4):555–564, 2012.

[197] Qi Sun, Li-Yi Wei, and Arie Kaufman. Mapping virtual and physical reality. *ACM Transactions on Graphics (TOG)*, 35(4):1–12, 2016.

[198] Qi Sun, Anjul Patney, Li-Yi Wei, Omer Shapira, Jingwan Lu, Paul Asente, Suwen Zhu, Morgan McGuire, David Luebke, and Arie Kaufman. Towards virtual reality infinite walking: dynamic saccadic redirection. *ACM Transactions on Graphics (TOG)*, 37(4): 1–13, 2018.

[199] Subhash Suri and Joseph O'Rourke. Worst-case optimal algorithms for constructing visibility polygons with holes. In *Proceedings of the second annual symposium on Computational geometry*, pages 14–23, 1986.

[200] Takafumi Taketomi, Hideaki Uchiyama, and Sei Ikeda. Visual slam algorithms: A survey from 2010 to 2016. *IPSJ Transactions on Computer Vision and Applications*, 9(1):1–11, 2017.

[201] Tatsuto Takeuchi and Karen K De Valois. Velocity discrimination in scotopic vision. *Vision research*, 40(15):2011–2024, 2000.

[202] Shigehito Tanahashi, Hiroyasu Ujike, Ryo Kozawa, and Kazuhiko Ukai. Effects of visually simulated roll motion on vection and postural stabilization. *Journal of neuroengineering and rehabilitation*, 4:1–11, 2007.

[203] Sébastien Tanguy, Gaëlle Quarck, Olivier Etard, Antoine Gauthier, and Pierre Denise. Vestibulo-ocular reflex and motion sickness in figure skaters. *European journal of applied physiology*, 104:1031–1037, 2008.

[204] MiM Taylor and C Douglas Creelman. Pest: Efficient estimates on probability functions. *The Journal of the Acoustical Society of America*, 41(4A):782–787, 1967.

[205] Jerald Thomas and Evan Suma Rosenberg. A general reactive algorithm for redirected walking using artificial potential functions. In *2019 IEEE Conference on Virtual Reality and 3D User Interfaces (VR)*, pages 56–62. IEEE, 2019.

[206] Jerald Thomas and Evan Suma Rosenberg. Reactive alignment of virtual and physical environments using redirected walking. In *2020 IEEE Conference on Virtual Reality and 3D User Interfaces Abstracts and Workshops (VRW)*, pages 317–323. IEEE, 2020.

[207] Jerald Thomas, Courtney Hutton Pospick, and Evan Suma Rosenberg. Towards physically interactive virtual environments: Reactive alignment with redirected walking. In *26th ACM Symposium on Virtual Reality Software and Technology*, pages 1–10, 2020.

[208] H Ujike, T Yokoi, and S Saida. Effects of virtual body motion on visually-induced motion sickness. In *The 26th Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, volume 1, pages 2399–2402. IEEE, 2004.

[209] Martin Usoh, Kevin Arthur, Mary C Whitton, Rui Bastos, Anthony Steed, Mel Slater, and Frederick P Brooks Jr. Walking $>$ walking-in-place $>$ flying, in virtual environments. In *Proceedings of the 26th annual conference on Computer graphics and interactive techniques*, pages 359–364. ACM Press/Addison-Wesley Publishing Co., 1999.

[210] Martin Usoh, Ernest Catena, Sima Arman, and Mel Slater. Using presence questionnaires in reality. *Presence*, 9(5):497–503, 2000.

[211] Jur Van Den Berg, Stephen J Guy, Ming Lin, and Dinesh Manocha. Reciprocal n-body collision avoidance. In *Robotics research*, pages 3–19. Springer, 2011.

[212] Ralf Van der Lans, Michel Wedel, and Rik Pieters. Defining eye-fixation sequences across individuals and tasks: the binocular-individual threshold (bit) algorithm. *Behavior research methods*, 43:239–257, 2011.

[213] Oliver Van Kaick, Hao Zhang, Ghassan Hamarneh, and Daniel Cohen-Or. A survey on shape correspondence. In *Computer graphics forum*, volume 30, pages 1681–1707. Wiley Online Library, 2011.

[214] Khrystyna Vasylevska and Hannes Kaufmann. Towards efficient spatial compression in self-overlapping virtual environments. In *2017 IEEE Symposium on 3D User Interfaces (3DUI)*, pages 12–21. IEEE, 2017.

[215] Khrystyna Vasylevska, Hannes Kaufmann, Mark Bolas, and Evan A Suma. Flexible spaces: Dynamic layout generation for infinite walking in virtual environments. In *2013 IEEE Symposium on 3D User Interfaces (3DUI)*, pages 39–42. IEEE, 2013.

[216] Eric Veach. *Robust Monte Carlo methods for light transport simulation*. Stanford University, 1998.

[217] Eric Veach and Leonidas J Guibas. Optimally combining sampling techniques for monte carlo rendering. In *Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*, pages 419–428, 1995.

[218] Remco C Veltkamp and Michiel Hagedoorn. Shape similarity measures, properties and constructions. In *International Conference on Advances in Visual Information Systems*, pages 467–476. Springer, 2000.

[219] Weizhuo Wang, Michael Raitor, Steve Collins, C Karen Liu, and Monroe Kennedy. Trajectory and sway prediction towards fall prevention. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 10483–10489. IEEE, 2023.

[220] William H Warren and Kenneth J Kurtz. The role of central and peripheral vision in perceiving the direction of self-motion. *Perception & Psychophysics*, 51(5):443–454, 1992.

[221] William H Warren, Bruce A Kay, Wendy D Zosh, Andrew P Duchon, and Stephanie Sahuc. Optic flow is used to control human walking. *Nature neuroscience*, 4(2):213–216, 2001.

[222] LA1 Warwick-Evans, N Symons, T Fitch, and L Burrows. Evaluating sensory conflict and postural instability. theories of motion sickness. *Brain research bulletin*, 47(5):465–469, 1998.

[223] Andrew B Watson and Denis G Pelli. Quest: A bayesian adaptive psychometric method. *Perception & psychophysics*, 33(2):113–120, 1983.

[224] Nicholas A Webb and Michael J Griffin. Eye movement, vection, and motion sickness with foveal and peripheral vision. *Aviation, space, and environmental medicine*, 74(6): 622–625, 2003.

[225] Felix A Wichmann and N Jeremy Hill. The psychometric function: Ii. bootstrap-based confidence intervals and sampling. *Perception & psychophysics*, 63:1314–1329, 2001.

[226] Felix A Wichmann and N Jeremy Hill. The psychometric function: I. Fitting, sampling, and goodness of fit. *Perception & psychophysics*, 63(8):1293–1313, 2001.

[227] Felix A Wichmann and N Jeremy Hill. The psychometric function: II. Bootstrap-based confidence intervals and sampling. *Perception & psychophysics*, 63:1314–1329, 2001.

[228] Jan M Wiener, Gerald Franz, Nicole Rossmanith, Andreas Reichelt, Hanspeter A Mallot, and Heinrich H Bülthoff. Isovist analysis captures properties of space relevant for locomotion and experience. *Perception*, 36(7):1066–1083, 2007.

[229] Betsy Williams, Gayathri Narasimham, Bjoern Rump, Timothy P McNamara, Thomas H Carr, John Rieser, and Bobby Bodenheimer. 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*, pages 41–48, 2007.

[230] Niall L Williams and Tabitha C Peck. Estimation of rotation gain thresholds considering fov, gender, and distractors. *IEEE transactions on visualization and computer graphics*, 25(11):3158–3168, 2019.

[231] Niall L Williams, Aniket Bera, and Dinesh Manocha. Arc: Alignment-based redirection controller for redirected walking in complex environments. *IEEE Transactions on Visualization & Computer Graphics*, 27(05):2535–2544, 2021.

[232] Niall L Williams, Aniket Bera, and Dinesh Manocha. Redirected Walking in Static and Dynamic Scenes Using Visibility Polygons. *IEEE Transactions on Visualization & Computer Graphics*, 27(11):4267–4277, 2021.

[233] Niall L Williams, Aniket Bera, and Dinesh Manocha. ENI: Quantifying Environment Compatibility for Natural Walking in Virtual Reality. In *2022 IEEE Virtual Reality and 3D User Interfaces (VR)*, pages 419–427. IEEE, 2022.

[234] Niall L Williams, Nicholas Rewkowski, Jiasheng Li, and Ming C Lin. A framework for active haptic guidance using robotic haptic proxies. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 12478–12485. IEEE, 2023.

[235] Niall L Williams, Geonsun Lee, Aniket Bera, and Dinesh Manocha. Persistent distractors for natural locomotion in virtual reality. *In submission*, 2024.

[236] Niall L Williams, Logan C Stevens, Aniket Bera, and Dinesh Manocha. Sensitivity to redirected walking considering gaze, posture, and luminance. *In submission*, 2024.

[237] Jeremy M Wolfe, Keith R Kluender, Dennis M Levi, Linda M Bartoshuk, Rachel S Herz, Roberta L Klatzky, Susan J Lederman, and Daniel M Merfeld. *Sensation & perception*. Sinauer Sunderland, MA, 2006.

[238] Jackie Yang, Christian Holz, Eyal Ofek, and Andrew D Wilson. Dreamwalker: Substituting real-world walking experiences with a virtual reality. In *Proceedings of the 32nd Annual ACM Symposium on User Interface Software and Technology*, pages 1093–1107, 2019.

[239] Sanae Yoshimoto, Katsunori Okajima, and Tatsuto Takeuchi. Motion perception under mesopic vision. *Journal of Vision*, 16(1):16–16, 2016.

[240] Christopher You, Brett Benda, Evan Suma Rosenberg, Eric Ragan, Benjamin Lok, and Jerald Thomas. Strafing gain: Redirecting users one diagonal step at a time. In *2022 IEEE international symposium on mixed and augmented reality (ISMAR)*, pages 603–611. IEEE, 2022.

[241] Run Yu, Zachary Duer, Todd Ogle, Doug A Bowman, Thomas Tucker, David Hicks, Dongsoo Choi, Zach Bush, Huy Ngo, Phat Nguyen, et al. Experiencing an invisible world war i battlefield through narrative-driven redirected walking in virtual reality. In *2018 IEEE Conference on Virtual Reality and 3D User Interfaces (VR)*, pages 313–319. IEEE, 2018.

[242] Markus Zank and Andreas Kunz. Optimized graph extraction and locomotion prediction for redirected walking. In *2017 IEEE Symposium on 3D User Interfaces (3DUI)*, pages 120–129. IEEE, 2017.

[243] Michael A Zmuda, Joshua L Wonser, Eric R Bachmann, and Eric Hodgson. Optimizing constrained-environment redirected walking instructions using search techniques. *IEEE transactions on visualization and computer graphics*, 19(11):1872–1884, 2013.