This is a preprint version to appear in the proceedings of the 21st IEEE International Symposium on Mixed and Augmented Reality (ISMAR).

Inverse Kinematics Assistance for the Creation of Redirected Walking Paths

Jerald Thomas* Virginia Tech
Seraphina Yong† University of Minnesota
Evan Suma Rosenberg‡ University of Minnesota

ABSTRACT

Virtual reality interactions that require a specific relationship between the virtual and physical coordinate systems, such as passive haptic interactions, are not possible with locomotion techniques using redirected walking. To address this limitation, recent research has introduced environmental alignment, which is the notion of using redirected walking techniques to align the virtual and physical coordinate systems such that these interactions are possible. However, the previous research has only implemented environmental alignment in a reactive manner, and the authors posited that better results could be achieved if a predictive algorithm was instead used. In this work, we introduce a novel way to model the environmental alignment problem as a version of the inverse kinematics problem which can be incorporated into several existing predictive algorithms, as well as a simple proof-of-concept implementation. An exploratory human subject study (N=17) was conducted to evaluate this implementation's usability as a tool for authoring planned path redirected walking scenarios that incorporate physical interactivity. To our knowledge, this is the first study to evaluate redirected walking experience design tools and provides a possible framework for future studies. Our qualitative analysis of the results generated both guidance for integrating automatic solvers and broad recommendations for designing redirected path authoring tools.

Index Terms: Human-centered computing—Human computer interaction (HCI)—Interaction paradigms—Virtual reality; Computing methodologies—Computer graphics—Graphics systems and interfaces—Virtual reality;

1 INTRODUCTION

The various methods that allow virtual reality (VR) users to move throughout a virtual environment are considered locomotion techniques. Due to advances in display and tracking technology, large-scale VR experiences are becoming more readily available for average consumers, and the larger scale of these experiences allows developers to better leverage natural locomotion. Natural locomotion, where the user physically walks and rotates to translate and rotate their virtual representation, has been shown to have several benefits over other locomotion methods, including increased spatial awareness [37] and sense of presence [43]. However, a notable limitation of natural locomotion is that physical obstacles and the extent of the tracking system constrain the user's movement. Typically, the size of the available physical environment is smaller than the virtual environment the user is experiencing, requiring techniques that augment natural locomotion. Redirected walking (RDW) is a common technique that augments natural locomotion to effectively increase the size of the physical walking space [35].

When RDW is applied, the mapping between the user's virtual and physical coordinate systems no longer remains static as it does with traditional natural locomotion [41], which in turn makes it difficult for VR developers to integrate physical interactions into environments that use RDW. This limits the amount of immersion these environments can provide because interactivity with the physical environment, particularly passive haptic feedback, has been shown to significantly enhance the user's experience [21, 23]. The combination of RDW and physical interaction, therefore, represents a particularly compelling opportunity to create immersive virtual reality experiences that transcend the limitations of either the purely virtual or completely real world. Accomplishing this is the main goal of environmental alignment, a concept introduced by Thomas et al. in 2020 [41]. Environmental alignment uses RDW techniques to align the user's virtual and physical coordinate systems, thereby enabling interactivity with the physical environment. However, the prior work only evaluated environmental alignment using reactive RDW algorithms that do not predict where the user will move, and as such, provided sub-optimal results.

Despite nearly two decades of research, redirected walking techniques have had only a modest real world impact, and the complexity of integrating RDW in virtual reality applications remains a major practical barrier for developers. Although some open-source implementations of RDW algorithms have been developed, such as the Redirected Walking Toolkit [4] and the OpenRDW platform [26], there has been a notable lack of attention on authoring tools for RDW experiences. The addition of physical interaction further increases the complexity of the design space, and to our knowledge no prior work has ever developed practical solutions for authoring VR experiences that integrate redirected walking with passive haptics.

In this paper, we address the gap in the literature by presenting a proof-of-concept inverse kinematics based algorithm that assists developers by calculating the RDW trajectories required to achieve alignment between the physical and virtual environments, advancing the state-of-the-art by incorporating movement prediction into environmental alignment. We also present an exploratory user study that compares this semi-automated approach with manual configuration of RDW parameters. This user study is the first to formally evaluate RDW authoring tools, and serves as an example for similar future research.

2 BACKGROUND AND RELATED WORKS

2.1 Redirected Walking

RDW is a method of VR locomotion that allows the user to use natural locomotion to control their representation in the virtual world. First introduced by Sharif Razzaque, RDW uses subtle perceptual illusions to effectively increase the size of the tracked physical environment [36]. These perceptual illusions, known as self-motion gains, introduce discrepancies between the user's physical and perceived virtual movements. If applied correctly, the discrepancies can go unnoticed by the user because the visual system tends to dominate when different stimuli are presented to the visual and vestibular systems. However, this dominance is only reliable to a certain degree. The value at which a particular self-motion gain will begin to be noticeable by the average user is its detection threshold. Several human-subject studies have empirically determined the detection thresholds for each of the major self-motion gains, and these values are used in the majority of RDW literature [38].

*e-mail: jeraldlt@vt.edu
†e-mail: yong0021@umn.edu
‡e-mail: suma@umn.edu

However, more recent research suggests that the detection thresholds can have numerous factors, including biological sex, HMD field of view, cognitive load, and other individual differences [22, 31, 32, 47]. The majority of these works evaluate the user's detection threshold for a specific self-motion gain in isolation, but it is possible to combine self-motion gains and doing so has been shown to change their individual perception thresholds [17, 30].

There are situations where the magnitude of the gains required to prevent a user from colliding with a boundary exceeds the detection thresholds. At this point, the system can employ one of two methods. The first, and more traditional method, is a reorientation event. The most common form of reorientation events are called resets [44]. Resets work by pausing the VR experience prior to the user exiting the physical boundaries. Once paused, the experience instructs the user to physically turn to face an advantageous position. In most literature, this advantageous position is the center of the physical environment, though some research has explored other "reset strategies" [42]. The experience then continues when the reset concludes. Resets, as described, have a very negative impact on the user's experience, and considerable effort is put in mitigating their use. However, prior work has shown that it is possible to incorporate resets into the experience narrative, allowing for resets without the need to pause the experience [16, 34]. For the second method, the RDW gains are temporarily set to values that exceed the detection threshold limits [9]. This allows the system to use stronger, but noticeable, RDW gains to prevent the user from colliding with a boundary. This method is relatively new, and there is no good empirical understanding of its effects on the user. Both methods have their pros and cons, and it is up to the experience designer to determine which is more appropriate for their application.

A publication providing a more in-depth overview of the first 15 years of RDW research was published in 2018 by Nilsson et al. [33]. Recently, most research has focused on developing more effective RDW algorithms. RDW algorithms are the mechanisms that decide how much of which gain to apply at when. Generally speaking, there are two main categories of RDW algorithms: reactive and predictive. Reactive algorithms do not know where the user intends to move in the virtual environment and must make decisions based on immediately available information. As a result, reactive algorithms are very generalizable in use but often only provide very localized optimizations. On the other hand, predictive algorithms have at least some knowledge of the user's intended path and can create a more optimized solution. However, predictive algorithms are less generalizable due to unreliable prediction in more open virtual environments.

Redirected Walking Algorithms

The first introduced RDW algorithms, Steer to Center (S2C) and Steer to Orbit (S2O), are both reactive algorithms [36]. Until recently, S2C was considered the best performing reactive algorithm [5, 20]. However, two new classes of reactive algorithms have emerged in recent years that generally perform equal to, or better than, the traditional reactive methods. The first uses artificial potential functions (APF-RDW) to calculate a better heuristic for determining per-frame gain values [9, 28, 42]. The second uses machine learning to optimize gain selection for a specific physical environment [11, 14, 25, 40].

Predictive algorithms consist of two sub-categories: dynamic and static [3]. Dynamic predictive algorithms, such as FORCE and MPCRed are constantly updating a prediction model based on the layout of the virtual environment, and the user's position within it [29, 52]. In most cases, if the algorithm can produce a reliable, accurate prediction model, it will perform better than a reactive algorithm. Static predictive algorithms, such as COPPER, work on the assumption that the user's entire virtual path is known ahead of time and that the user will not deviate from that path [3].

2.2 Interactions with the Physical Environment

Interactions with the physical environment have long been studied to enhance VR experiences, and a key way that they do so is by increasing the user's sense of presence [13]. A particularly useful technique is passive haptic feedback, where a physical proxy object is mapped to and used as a counterpart for a virtual object [27]. When implemented in a convincing way, passive haptic feedback can improve the user's sense of presence even more than other physical interaction techniques [21, 23].

Azmandian et al. showed that it is possible to subtly redirect a user's reaching motion in order to have the user interact with multiple virtual objects while interacting with only one physical object, which the authors termed "retargeting" [6]. Zenner et al. continued this work by determining the perception thresholds of the applied manipulations [51]. Where Azmandian et al. used consistent movement manipulations to retarget multiple virtual objects onto one physical object, Wilson et al. employed a change blindness illusion to achieve a similar effect [48].

Environmental Alignment

RDW works by adding discrepancies between the user's virtual and physical paths. A negative side-effect of the added discrepancies is a misalignment of the virtual and physical coordinate systems. With few exceptions, experiences that incorporate RDW cannot also incorporate interactions that require aligned virtual and physical coordinate systems such as passive haptic feedback. Steinicke et al. and Kohli et al. have explored solutions to this problem, but both required manual selection of the RDW gains to work [24, 39].

In an attempt to solve this problem in a more generalizable manner, Thomas et al. introduced the idea of environmental alignment [41]. Environmental alignment uses RDW gains to keep users from colliding with boundaries and obstacles, as with traditional RDW algorithms, and to steer users towards physical-virtual position pairs that result in an aligned physical and virtual coordinate system. The implementation was accomplished using an APF-RDW reactive algorithm, and while the results were promising, they were not always able to accomplish alignment. Williams et al. has also used the concept of alignment to introduce a new RDW algorithm, ARC [45]. Williams et al. further improved upon this method by incorporating the concept of visibility polygons in their algorithm [46]. Researchers have also implemented reactive alignment using machine learning techniques [12].

Recent work presented by Xu et al. discretizes the physical environment and attempts to find optimal paths for every pair of physical locations [50]. While this work was not proposed as a way to implement alignment, we believe that it could be used to implement reactive alignment with minimal effort.

Though reactive alignment is an area of active development and research, there is no work on implementing alignment in a predictive manner. In this paper, we extend the work performed by Thomas et al. by introducing a method to accomplish predictive environmental alignment.

3 Method

In this section we, describe how predictive RDW can be cast as an inverse kinematics problem. Several objects in the physical world can be modeled mechanically as a kinematic chain, a series of rigid bodies connected by joints. To calculate the position of a joint in the chain, we must know the length of the prior rigid body, the angle of the prior joint, and the position of the prior joint. Thus, if the position of the first joint (termed the "root"), the length of each rigid body, and the angle of each joint are known, it is possible to describe the entire chain completely. This process of calculating the position of joint i by first calculating the position of joint $i-1$ is called forward kinematics (FK). Often the goal of FK is to determine the position of the final joint in the chain (termed the "end effector").

In many cases, it is desirable to have the end effector be in a specific position. Typically the positions of the other joints are not as important as long as all constraints are met (most real-world joints have a limit to their rotation, for example). The problem of finding a satisfying set of joint positions resulting in the end effector being in the desired position is called inverse kinematics (IK).

Predictive RDW algorithms work by calculating the optimal gain values for each segment of a predicted virtual path. For this paper, we define a path as a series of waypoints connected by a series of segments. We also assume that a user will repeatedly stand at virtual waypoint $i-1$, rotate to face virtual waypoint i, and then walk from virtual waypoint $i-1$ to virtual waypoint i. This is repeated until the user reaches the final waypoint. Given the virtual path, the user's starting physical location and rotation, and a set of gains, it is possible to calculate the resulting physical path, assuming that the user follows the virtual path as expected. This process reduces to an FK problem where the position of physical waypoint $i-1$, the rotation at virtual waypoint $i-1$ multiplied by the corresponding rotation gain, and the length of the segment between virtual waypoints $i-1$ and i multiplied by the corresponding translation gain together determine the position of physical waypoint i. Equation 1 shows the FK equation for a traditional 2D kinematic chain where $P \in \mathbb{R}^2$ is the position of a joint, $L \in \mathbb{R}$ is the length of a rigid body, and $\theta \in \mathbb{R}$ is the rotation of a joint in the local coordinate system of P_{i-1}.

$$P_i = P_{i-1} + (L_{i-1}\ cos(\theta_{i-1}),\ L_{i-1}\ sin(\theta_{i-1})) \tag{1}$$

Similarly, Equation 2 shows the FK equation for a pre-determined RDW path where $P^P \in \mathbb{R}^2$ is the position of a physical waypoint, $P^V \in \mathbb{R}^2$ is the position of a virtual waypoint, $T \in \mathbb{R}$ is the amount of translation for a given segment (the length of the corresponding virtual path segment ($||P_i^V - P_{i-1}^V||$) multiplied by its translation gain (g_{i-1}^R)), and $R \in \mathbb{R}$ is the rotation of the corresponding virtual waypoint in the local coordinate system of P_{i-1}^V (θ_{i-1}) multiplied by its rotation gain (g_{i-1}^R).

$$P_i^P = P_{i-1}^P + (T_{i-1}\ cos(R_{i-1}),\ T_{i-1}\ sin(R_{i-1})) \tag{2}$$

We observe that rotation gains are equivalent to scaling the angle between two joints, and translation gains are equivalent to scaling the distance. In the robotics literature, the equivalent joints are known as 2D revolute and 2D prismatic joints, respectively.

Just as it is possible to determine the final position of a user's physical path by using FK, we propose that IK can calculate the gain values for a given virtual path resulting in the user's physical path ending at a specific physical position. This effectively solves the environmental alignment problem for planned-path predictive algorithms, and for this reason, we refer to this strategy as Environmental Alignment Inverse Kinematics (EA-IK).

3.1 Implementation

Our current implementation of EA-IK uses Cyclic Coordinate Descent (CCD) [49] as the IK solver algorithm. However, several decades of research have explored IK and IK solvers, and many more effective and efficient algorithms exist. We chose CCD in order to provide a proof-of-concept as it is simple to conceptually grasp and implement.

The difference between the classical IK problem and the EA-IK problem is the use of resets. Continuing the IK analogy, resets would be akin to dynamically adding joints to segments of the kinematic chain. To the best of our knowledge, no existing IK algorithms consider this ability as it breaks the assumptions of a classical kinematic chain. Our implementation takes a basic approach to solve this problem, though we propose a more advanced solution in Section 6.

The user's starting position and rotation within the physical environment will be the root (R), and after traversing the path, the user's position in the physical environment will be the end effector (E). The goal of the system is to get E to a target location (T) Every virtual waypoint between R and E represents a joint that can rotate between the smallest possible rotation (the waypoints's rotation times the minimum rotation gain) and the largest possible rotation (the waypoints's rotation times the maximum rotation gain). Similarly, every virtual path segment represents a rigid body which can vary in length between the smallest possible length (the virtual path segment length times the minimum translation gain) and the largest possible length (the virtual path segment length times the maximum translation gain).

The CCD algorithm works on a parameter space, an n-dimensional space where each dimension represents one parameter that the algorithm can manipulate. In traditional IK scenarios, this parameter space typically consists of a dimension for each degree of freedom of each joint in the kinematic chain. The EA-IK CCD parameter space consists of a translation gain dimension for each path segment and a rotation gain dimension for each waypoint.

A pass of the EA-IK CCD algorithm iterates over its parameter space. Each iteration sets the current gain value to its minimum and then increases it in steps. We found a step size of 0.001 worked well in our case. After each gain increase, the physical path and the corresponding error (the euclidean distance from E to the target T) is calculated. This process continues until either the gain reaches its maximum value or the calculated error reaches a minimum value.

To find an optimal solution, EA-IK CCD performs multiple passes until the final error reaches a minimum value. Once the algorithm determines a solution, it simulates a traversal of the resulting physical path starting at R. If it detects an intersection with the physical boundaries, it creates a reset. To create the reset, the physical path up to the intersection point is "locked in", and R is positioned at the intersection point and oriented to face the center of the physical environment. Then the whole EA-IK CCD algorithm is executed again, solving for the portion of the virtual path remaining after the reset and the new physical root R. This repeats until the entire path lies within the physical boundaries. A pseudo-code description of this algorithm can be found in Appendix B in the supplementary materials.

4 Experiment

The primary purpose of this experiment was to determine if using IK gain selection for planned path RDW led to a better designer experience. We presented participants with two gain selection modes: manual and automatic. Manual mode provided an interface where users had to select each segment's rotation and translation gain value manually. Automatic mode used the EA-IK algorithm described in Section 3.1 to select the gains automatically. The entire experiment was completed in the Unity3D Editor, and custom interfaces were developed for each mode.

For this experiment, we had three hypotheses:

H1: Automatic mode will result in faster task completion times. As there are fewer parameters for the user to change, we believe that the task will take less time when automatic mode is used.

H2: Manual mode will have fewer resets. The implemented version of EA-IK does not optimize for resets. We hypothesize that in manual mode, users will be able to tune parameters as they accomplish the task to reduce the number of resets.

H3: Automatic mode would be overall more usable. As there are fewer parameters for the user to keep track of, we believe that the automatic mode will produce a smoother and more usable experience.

4.1 Design

A two-by-three within-subjects study was designed and implemented. The two independent variables were the gain selection mode and

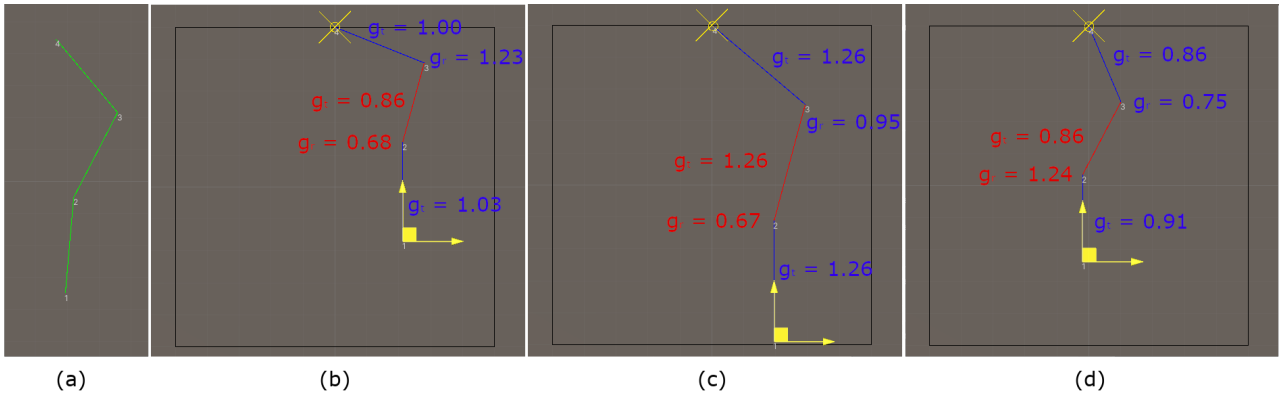

Figure 1: Three different feasible physical paths (b, c, and d) for the same virtual path (a). Images are captured as a user was moving the physical path starting location. During this time our algorithm automatically calculated the gains necessary for the end of the physical path to be positioned at a pre-determined target position (the yellow X). The calculated gain values were added to the images in post-production, are colored to match their corresponding physical path segment. Here g_R refers to rotation gain and g_T refers to translation gain.

the path difficulty. The gain selection mode was either manual or automatic as described above, and the path difficulties were either easy, medium, or hard. The number of path waypoints determined a path's difficulty; easy paths had four waypoints, medium paths had six waypoints, and hard paths had eight waypoints. Each set of conditions was presented once, for a total of six trials. The presentation order of experimental conditions was counter-balanced. There were two different paths for each difficulty to reduce the chance of a learning effect. The first path was randomly generated such that:

1. the total path length was equal to five meters multiplied by the number of path segments
2. each path segment length was a minimum of three meters
3. the total amount of absolute rotation was equal to the number of waypoints - 2 (there was no rotation at the first and last waypoints) multiplied by $\frac{3\pi}{8}$
4. the minimum absolute rotation at any of the waypoints with rotation was $\frac{\pi}{8}$

These values were chosen after pilot testing to provide a path that is similar to generated virtual paths in the RDW literature while allowing control of the difficulty with a single variable. The second path was a copy of the first, but mirrored about the Z-axis. This promoted equal difficulty across the two modes and mitigated the impacts of any learning effects.

4.2 Interface

We developed the experiment interface for Unity3D LTS version 2020.3.18f1. An "Experiment" menu added to the menu bar provided the functionality for the participant to start new trials when instructed by the experimenter. When a trial was started, a new scene was programmatically generated, consisting of two game objects; one representing the physical environment and one representing the virtual path. The physical environment game object had a child object representing the user's physical starting location and rotation. Unity3D "Gizmos" were used to draw the virtual path, the calculated physical path, the physical environment boundaries, and the physical target location.

One of two versions of a C# script was added as a component to the virtual path game object, depending on the gain selection mode (Figure 2 shows the Manual mode component). This script provided helpful information, such as distance to target and gain

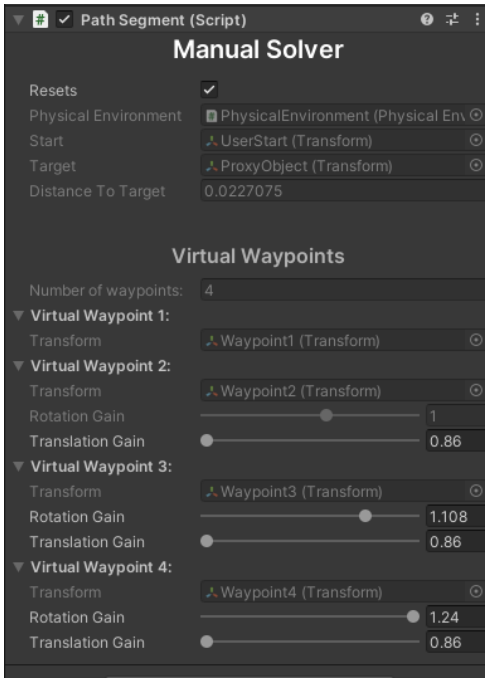

Figure 2: A screen-capture of the Unity3D C# component interface that controlled the RDW calculations. The interface for manual mode allowed for participants to select gain values for each segment using the sliders. The interface for automatic mode still had the sliders, but they were deactivated so that the participants could not manually change the gain value.

values, and allowed manipulation of the gains when in manual mode. It also provided the ability to temporarily allow the path to exit the bounds of the physical environment. This option helped to see how the gains affected the path without resets, but the task could not be completed if this option was activated. Figure 1 shows how by simply moving the physical path's starting location the automatic mode functionality would update the gain values.

4.3 Participants

17 participants (15 Male, 2 Female) between the ages of 22 and 50 (Mean=30.71, SD=9.18) volunteered for participation. Participants

were recruited via postings to online communities such as Reddit and the 3DUI mailing list as well as emails sent to colleagues of the authors. As the experiment was conducted during the COVID-19 pandemic, participants took part in the study remotely via Zoom. 15 participants had no prior experience with RDW, 1 participant had some prior experience with RDW, and 1 participant had moderate prior experience with RDW. All participants were provided with a $20 Amazon.com gift card. Our inclusion criteria, which all participants met, were that they are 18 years or older, able to converse in both written and spoken English, and had one more more years of experience creating VR applications using the Unity3D game engine.

4.4 Procedure and Metrics

At the start of the experiment session, the experimenter received oral confirmation that the participant met all of the inclusion criteria, reviewed the IRB-approved information sheet with the participant, and then received permission to record the session. The experimenter would then briefly introduce RDW, present the problem and task, and show the participant how to use the interface. Once the experimenter finished, they instructed the participant to complete four practice trials using the interface to solve two paths of trivial difficulty using manual and automatic modes. Participants were then allowed to repeat the practice trials until they felt sufficiently familiar with the interface, at which point they would begin the experimental trials.

For each trial, participants were provided with one of the immutable virtual paths generated (as described in 4.1). A simple representation of a physical environment with 10 meter by 10 meter boundaries was displayed, which contained the representation of the physical path. A game object that represented the user's starting physical location and rotation determined the physical path's position and orientation within the physical environment. The participant was allowed to rotate this game object and move it anywhere within the bounds of the represented physical environment. Doing so would automatically update the physical path. When calculating the physical path, resets were automatically accounted for so that after a reset, the physical path was pointing towards the center of the represented physical environment.

Participants were instructed that their primary goal for each trial was to find a combination of starting location, starting rotation, and RDW gains that resulted in a physical path with a final waypoint within 0.25 meters of the target. Their secondary goal was to try to reduce the number of resets. Participants were allowed to finish the task if the final waypoint of the physical path was within 0.25 meters of the target, regardless of the number of resets. However, they could choose to continue the task in order to achieve fewer resets. If a participant did not complete the task with 10 minutes, they were allowed to get the endpoint as close to the target as possible and finish the task.

In automatic mode, manipulating the user's starting location and rotation was the only manipulation the participant could make to complete the task. Every participant's change would trigger the EA-IK algorithm to execute and find a set of gains. In manual mode, the participant also had to set both the rotation and translation gains for each segment of the paths manually.

The experimental trials were broken down by mode into two blocks. The tasks to be completed in each block consisted of one easy path, one medium path, and one hard path, in that order. After each block, the experimenter instructed participants to fill out a questionnaire about the gain selection mode they had just used. These questionnaires consisted of the NASA Task Load Index (TLX) questionnaire [18], the System Usability Scale (SUS) questionnaire [10], and a prompt for the participant to detail any strategy they may have used during that block.

Once the participant finished both blocks and the subsequent questionnaires, the experimenter asked a series of nine semi-structured interview questions designed to elicit the user's thoughts on the individual components of the interface, as well as garner information on how it could be improved. The full list of questions can be found in Appendix A in the supplementary materials.

This experiment collected a series of both quantitative and qualitative metrics. Our quantitative metrics were the time it took to complete the task, the number of resets in the calculated physical path, the SUS questionnaire, and the NASA TLX questionnaire. Importantly, participants were not explicitly instructed to perform each trial as quickly as possible. Our qualitative metrics included questionnaire prompts to describe any strategy used for a given mode, sentiments and ideas from the think-aloud portion of the tasks, and sentiments and ideas from the semi-structured interview conducted at the end of the.

5 Results

5.1 Quantitative Results

Mode	Difficulty	Times (sec.)		Resets	
		Mean	SD	Med.	IQR
Automatic	Easy	51.13	39.96	0	0
	Medium	162.38	131.04	2	2
	Hard	315.57	139.95	3	0
Manual	Easy	154.50	107.79	0	0.25
	Medium	135.06	75.75	2	2
	Hard	241.56	109.74	3	1

Table 1: Table of descriptive statistics for the time to task completion and resets encountered quantitative metrics.

Descriptive statistics for all quantitative data are shown in Table 1. For data that were not normally distributed, non-parametric analyses were conducted and the results were reported using median and interquartile range. Statistical tests assumed a significance value of $a = .05$. Two participants were not able to complete the task for the Automatic-Hard condition, and one participant was not able to complete the task for the Automatic-Easy condition. All three instances were because the participants were not able to complete the task in the 10 minutes allotted. The corresponding data was left out of the quantitative analysis.

Completion Times. The completion time data were analyzed using a 3x2 repeated-measures ANOVA (difficulty x mode). Mauchly's test of sphericity indicated a possible violation, and so a Greenhouse-Geisser correction was applied. The results revealed a significant interaction, $F(1.78) = 12.54, p < .001, \eta^2 = .09$, and a significant main effect for difficulty, $F(1.83) = 22.70, p < .01, \eta^2 = .44$. The main effect for mode was not significant, $p = .41$. Post-hoc analysis was conducted using paired-sample t-tests with a Bonferroni correction for multiple comparisons. For the easy difficulty, participants completed the task faster using the automatic mode, $p < .001$. The completion times were not significantly different for the medium difficulty, $p > .99$. However, the trend was reversed for hard difficulty, with faster times in the manual mode, $p = .02$. These results only partially support H1.

Resets. Wilcoxon signed rank tests were conducted to compare modes for each difficulty level. For the easy difficulty, it was not possible to statistically analyze the data because all participants were able to completely avoid resets using the automatic mode. This resulted in a variance of 0, which causes the Z value if the test to be undefined. However, it should be noted that some participants still did encounter resets in easy trials using the manual mode. For the medium difficulty, the analysis was not significant, $p = .68$. However, the results were significant for the hard difficulty, $Z = 28.00, p = .02$. Although the medians for both modes in this

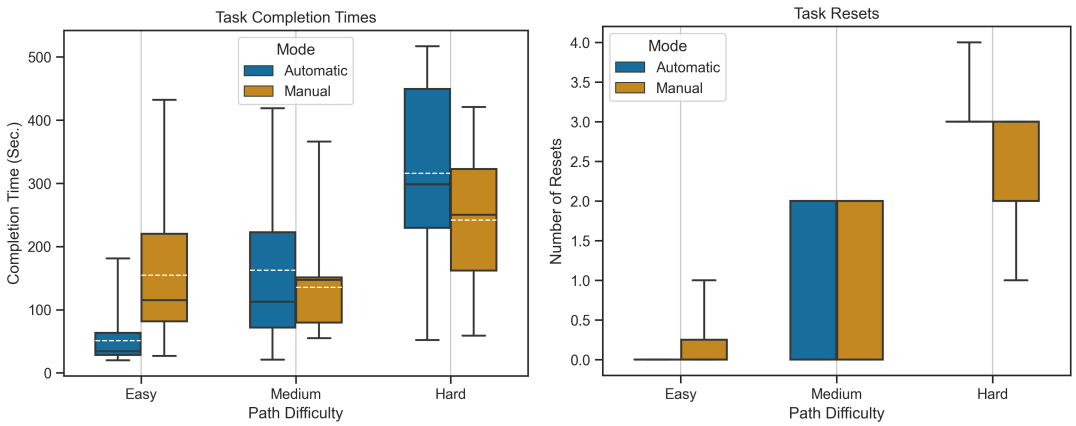

Figure 3: Box plots showing statistics for the time to task completion (left) and the resets encountered (right). The box extents represent IQR, the black line within a box is the median, the dashed white line within a box is the mean, and the whiskers represent total spread of the data.

difficulty were the same (3), the box plots show that some participants were able to achieve fewer resets using the manual mode (see Figure 3). Again, this only partially supports H2.

Usability and Workload. The overall SUS scores for each of the two modes were analyzed using a paired-samples t-test. The results were found no significant difference between the automatic mode ($M = 56.62$, $SD = 15.61$) and the manual mode ($M = 68.68$, $SD = 18.48$), $p = .08$. The six subscales of the NASA TLX were each analyzed using a Wilcoxon signed rank test. The manual mode ($Mdn = 5$, $IQR = 6$) was associated with less frustration as compared to the automatic mode ($Mdn = 12$, $IQR = 6$), $Z = 101.50, p = .02$. Manual mode ($Mdn = 5$, $IQR = 5$) was also associated with better performance as compared to the automatic mode ($Mdn = 5$, $IQR = 6$), $Z = 107.00, p = .05$. The following results were not significant: mental demand (Automatic $Mdn = 14$, $IQR = 8$; Manual $Mdn = 14$, $IQR = 7$), $p = .68$, physical demand (Automatic $Mdn = 4$, $IQR = 3$; Manual $Mdn = 4$, $IQR = 6$), $p = .09$, temporal demand (Automatic $Mdn = 5$, $IQR = 3$; Manual $Mdn = 5$, $IQR = 6$), $p = .33$, and effort (Automatic $Mdn = 13$, $IQR = 10$; Manual $Mdn = 13$, $IQR = 10$), $p = .67$. As the SUS scores were not found to be significantly different, we can neither accept nor reject H3.

5.2 Qualitative Results and Discussion

We performed qualitative analysis on the think-aloud and interview portions of the study. In this section, we discuss trends in participants' thoughts on the manual and automatic path selection interfaces, and detail design implications distilled from this analysis.

5.2.1 Giving Users Communication and Control

The presence or lack of control was an overarching theme in participants' responses, with almost all participants unanimously stating they felt they had more *"control"* in the manual mode, compared to the automatic mode. This is also reflected in the NASA TLX scores above, which showed less frustration and better performance than the automatic mode. The system behavior and communication style contributed to this perception. The great majority of participants expressed that the automatic solver made them feel lack of control, because the system *"would change things when I already had a plan on what to do next"*(P4), *"did too many steps for me"*(P8), and *"messed up how I created the path."* (P5). The automatic solver also did not communicate how or why it was making these changes. Participants perceived the automatic solver's behavior as *"random,"*(P10, P8, P5), *"chaotic,"*(P13), or *"unpredictable"*(P10, P14). This made them frustrated and dissatisfied with the task. Inability to understand how the system worked also made the participants unable to suggest improvements for the automatic solver.

The sentiments expressed here resonate with recent calls to improve explainability in automated interfaces [1]. Especially in contexts where users must respond to the system's behavior or collaborate with it, transparently communicating to users why the system is making a decision is critical. We saw above that participants felt negatively toward the system's obscure behavior. Their frustration about the lack of control also indicated desire for a more balanced power relationship with the system. Users who are expected to take responsibility for and do meaningful work in response to a system's behaviors have the right to know the reasons behind them. Knowing this lets users trust the system and also improves productivity [15]. Based on this understanding, we recommend that automated system elements must communicate the basis for all decisions (in this case, automated gain and path changes) and allow users control over the automation. For example, the automatic solver could visualize different path recommendations instead of directly changing the path, or only be activated at discretion of the user.

The minority of participants that demonstrated less frustration with the automatic solver shared the common approach of switching to a "hands-off" mindset, and stopped trying to think or work towards the goal. They described the automatic solver as *"just letting the algorithm do its thing,"*(P12), *"less thinking to do"*(P1), and *"much faster, I found myself unconsciously ignoring all the variables."*(P15). These participants gave up control of the system, which not everyone was able to do: P4 realized that they felt as if they *"[were] trying to reach the goals too much"* by having a plan while using the automatic solver. Participants' goals were also different – some participants felt forced to do this, while others wanted to put less effort into the tasks overall.

5.2.2 Flexibly Integrating Automation

Despite their frustrations with the automatic solver, almost all participants described various contexts in which they would like to make use of it. Their diverse preferences for integrating the automatic interface varied along dimensions of task complexity (number and length of path segments), task phase (beginning vs. end), and level of control desired.

Some participants felt that the automatic solver worked better sticking to easier tasks with less waypoints, while others wanted the automatic solver to complete the complex part of the path selection task for them. P8 explained, *"If the task is easy, the system can take more control, and just suggest some solutions. But if there are more waypoints I might want to first do more exploration of*

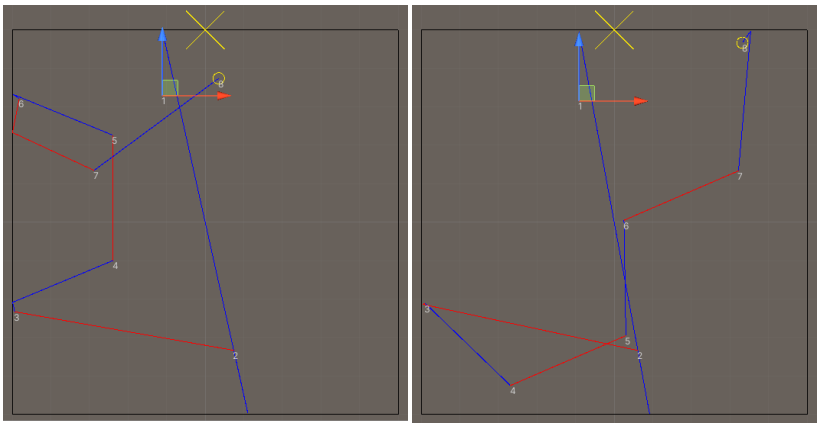

Figure 4: An example of how small changes (in this example, the starting position of the physical path) can drastically change the appearance of the entire physical path.

the starting points and figure out how to get the endpoint of the path close to the destination. If I can't do that, then I'd see if system has some suggestions for me." This is also reflected in the task completion times for different modes and difficulties in Figure 3, which show that easy tasks in automatic mode were completed more significantly more quickly than manual, but difficult tasks in automatic mode took significantly more time to complete. In contrast, P1 remarked *"If the difficulty is high, just make it automatic. Otherwise, make it manual."* We can see how this preference is moderated by participants' desire to understand the problem. They need the opportunity to work through more complex problems, in these cases they are not comfortable with the system jumping in. In contrast, participants who prefer the system to complete more difficult tasks for them do not express this desire to understand the context. Both approaches are valid in different situations, and both options should be supported.

Participants had divided preferences of whether to start or end with auto or manual modes, respectively. Participants P10, P4, and P7 stated preference for starting with manual, while P2, P15, P9, and P17 preferred ending with manual. P10 explained, *"I would definitely start with manual, fit it the best I can, and then use automatic at the end to make it more precise."* In contrast, P2 stated,*"You start off with automatic, and once you get close switch to manual so you can micro-manipulate the waypoint to finish it off."* The reasons behind similar rationale for different options was not evident, but there is clearly variation in individual preference for workflow ordering of the automatic and manual modes.

Many participants desired finer control over the automation, while others wanted it to be decided for them. Participants who desired finer granularity of control over the system also suggested the ability to interchange between automatic and manual mode on a segment-by-segment basis (P3, P9, P13, P5, P7, P14). P5 mentioned one possible way was to *"default to automatic, but have manual override over each individual segment."* Alternatively, P14 advised *"You could have an IK solver in between the vertices, and be able to control which segments use it."* Applying different solver modes to individual segments is an interesting suggestion, which would somewhat reduce the efficiency of the solver but affords much more user control. Other participants made broader suggestions: *"I'd rather have all manual or all algorithm. If I'm going to put my faith in the algorithm, that's that."* (P5)

The deviation of participant preferences on many levels indicates that such automated authoring tools should allow for presets that accommodate designers' motivations on multiple dimensions.

5.2.3 General Considerations for Redirected Path Authoring

In addition the above insights comparing the manual and automatic solvers, we also observed participant feedback on some characteristics inherent to interfaces for selecting RDW paths.

Especially in more complex paths, the high amount of resets made it difficult for participants to make sense of the path and adjust it. The basic design used in the study allowed for a more continuous workflow in which the path would change fluidly as the user manipulated it. Participants (P10, P8, P9, P5) mentioned that the constantly changing paths and reset locations this caused would make it challenging for them to plan out how to edit the path; this issue was present in both manual and automatic modes. P10 mentioned, *"It was hard to know exactly what effect the reset is going to have on the rest [of the path], so the visualization ends up being more complicated."* Providing a preview of the direction in which the path would change in response to a discrete edit by the user can help users understand these changes; this could take the form of temporally-incremental snapshots of the predicted path change. Separately visualizing the number of resets can also help users understand how their edits are altering the resets without having to individually count the resets appearing in the edit space.

Participants also (P8, P5) expressed being unsure of how effectively they were optimizing the path, and desired a form of dynamic score they could actively respond to. Optimality in RDW is subjective to the designer because it involves compromising between the gain and amount of resets. We can still give useful feedback by allowing designers to choose what factors they prioritize in a preset, and calculate a score customized to the designer's priority.

For both manual and automatic solvers, the way in which the resets forced the path to curl up inside the physical boundaries made it appear that all parts of a path would change in response to participants' adjustments of single segments or points, even though this was not the case. (see Figure 4). As a result, participants (P3, P8, P5, P17, P14) wanted to adjust the path incrementally in single segments or points without drastically changing the entire structure of the path. Given that this may cause violations of the gain threshold, we could address it by allowing users to make independent segment or point adjustments that violate the gain threshold, whilst making more noticeable indicators that the threshold has been violated (e.g. larger size, color, etc.). Another way to allow this kind of adjustment would be to make the reset direction editable for all resets.

On violating gain thresholds during the path selection process, participants provided distinct views. Though knowing that the gain thresholds are based on standard values, P12 and P5 both mentioned that they would like to be able to alter these thresholds to avoid

resets. Temporarily allowing violation of gain thresholds is a method explored in recent work, but there is not much evidence on how they affect user experience [9]. An interface that aims to give users more control over path experimentation can consider allowing the gain thresholds to be changed. P17 raised a point of how they felt less concerned about violating the gain thresholds because they were not experiencing the path while designing it: *"If I were in a headset, I'd be more cognizant of how each gain diminished the sense of presence; but I didn't experience this while manipulating the lines."* Future interfaces may want to consider how designers can better empathize with the end-user while making redirection decisions that could compromise the experience.

In general, path selection for RDW is a complex multi-step process with parameters that may differ based on individual designers' goals and mindset for the task. Therefore, the system should clearly communicate and facilitate step-by-step changes to the path, as well as create options for a diversity of designer contexts (e.g. hands-off vs. hands-on). We hope that these insights are able to assist with the design of redirected path authoring in the future.

6 Limitations and Future Work

Although the EA-IK implementation using CCS worked well as a proof-of-concept initiate exploratory analysis of assisted RDW experience generation, it is still a somewhat simplistic approach that only attempted to get the end of the path to the target. As a result, the algorithm is not able to optimize for the number of resets as in more traditional RDW algorithms. Determining the best way to optimize for both alignment error and the number of resets is a promising direction for future research. This algorithm also did not consider the user's orientation at the end of the path. The user's final orientation is critical for facilitating passive haptics. While a simple re-orientation event at the conclusion of the path can be used to align the user's orientation, further research on how to control the user's orientation with the IK algorithm is necessary.

There are situations in which the system may be unable to find a physical path that both reaches the target endpoint and maintains the perceptual thresholds. An elementary example would be if the distance between the start and goal positions is greater than the physical path length multiplied by the maximum translation gain. With the current implementation it is extremely difficult, if not impossible, to derive a closed form solution to determine if the combination of a virtual path, a starting physical pose, and a physical environment will successfully reach a target physical location without exceeding the user's perceptual thresholds. The only way to determine this is to perform the forward kinematics calculations while taking into account resets. Future research on classifying the outcome as either successful or unsuccessful based on the system inputs, without needing to calculate the full physical path, will make this work more usable in a wider range of applications.

We believe that a more advanced IK integration will produce not only more optimal results, but will have a smoother user experience. For example, we noticed in the experiment that participant computers with lower-end hardware tended to lag when solving for the longer paths, particularly with the CCD step size that was chosen. Increasing the step size could fix this problem, but utilizing a more effecient IK algorithm would be more ideal. Adopting the Forward And Backward Reaching Inverse Kinematics (FABRIK) [2] solver seems particularly promising, as it determines where the root needs to be for the end effector to reach the target. This information could be used to inform better starting locations for the user. This work also did not make use of curvature gains. Unlike translation and rotation gains, which are directly analogous to prismatic and revolute joints, there is no direct analog for curvature gain in the traditional IK literature. Adapting the model to also take advantage of curvature gains would likely have a positive effect on performance.

Additionally, in our implementation, resets were handled using a fairly naive approach. By locking the entire solved path before each reset occurrence, we kept the solver from finding a globally optimal solution. A more sophisticated way to approach this problem would be to allow the system to find a solution for the entire path that dynamically considers resets. One way to potentially do this is to break each physical path segment into smaller segments that prefer to be straight, but could bend if necessary. This would allow for "resets" to happen at locations other than the boundaries, which could improve the results.

When using either the manual or automatic solver, resets would often make small changes in the path parameters dramatically change the resulting physical path (Figure 4). Recent work by Hirt et al. also noticed the tendency for RDW paths to exhibit a "chaotic" nature, being heavily influenced by initial path conditions [19]. It might be possible that as a user traverses RDW paths human variation will invoke similar chaotic behavior. Azmandian et al. introduced a method for keeping users on a pre-planned path to solve a different problem, but it could be used in this situation as well [7,8]. However, further research on how the apparent chaotic nature of RDW paths affects the design and authorship of RDW experiences is necessary.

The main purpose of the presented technique is to produce a RDW path that ends in a specific physical location when provided with a known virtual path. As there are currently no other algorithms that accomplish this it is impossible to compare it alongside existing RDW algorithms, which have the singular goal of reducing the number of resets. However, as other methods with this goal are introduced, it will be imperative to do more direct comparisons.

The techniques developed in this paper were only initially evaluated in the case of a statically planned RDW scenario, where the entire virtual path was assumed to be known in advance. Future evaluations can consider use of the IK method for dynamic predictive RDW algorithms that provide a reliable prediction of the user's future path. The method demonstrated in this paper could could then be adapted to compute gains that would result in environment alignment. As IK solvers in general are not computationally expensive, it would even be possible to dynamically re-calculate them in real-time as the prediction changes.

Our user study was also limited in its simplistic interface design and participant sample size. The high level Unity3D abstractions led to some superficial usability issues such as lines being too thin to see well; creating a more polished user interface in the future would improve the authoring experience. The gender ratio of our participants was uneven, and while there are no expected gender effects for this study, a more balanced sample gives more representative results.

7 Conclusion

RDW allows VR developers more flexibility in the design of natural locomotion-based experiences. However, it also interferes with the ability to use passive haptic feedback and other forms of interactions with the physical environment. The introduction of EA-IK provides a solution that allows VR experiences to incorporate both physical interactivity and RDW. The presented user study is among the first to evaluate how designers actually create RDW experiences, and provides several useful insights to inform future development of these authoring tools. In the future, we believe that the ability to align the virtual and physical coordinate systems using predictive RDW algorithms will introduce new possibilities for both research and applications.

Acknowledgments

This work was supported in part by a US Department of Education GAANN Fellowship through the Department of Computer Science and Engineering, University of Minnesota and a Grant to Advance Graduate Education (GAGE) from the College of Science and Engineering, University of Minnesota. The authors would also like to thank Daniel Keyes for brainstorming some of the critical concepts.

REFERENCES

[1] A. Adadi and M. Berrada. Peeking inside the black-box: A survey on explainable artificial intelligence (xai). *IEEE Access*, 6:52138–52160, 2018. doi: 10.1109/ACCESS.2018.2870052

[2] A. Aristidou and J. Lasenby. Fabrik: A fast, iterative solver for the inverse kinematics problem. *Graphical Models*, 73(5):243–260, 2011.

[3] M. Azmandian. *Design and Evaluation of Adaptive Redirected Walking Systems*. PhD thesis, University of Southern California, 2018.

[4] M. Azmandian, T. Grechkin, M. Bolas, and E. Suma. The redirected walking toolkit: A unified development and deployment platform for exploring large virtual environments. In *IEEE VR Workshop on Everyday Virtual Reality*, 2016.

[5] M. Azmandian, T. Grechkin, M. T. Bolas, and E. A. Suma. Physical space requirements for redirected walking: How size and shape affect performance. In *ICAT-EGVE*, pp. 93–100, 2015.

[6] M. Azmandian, M. Hancock, H. Benko, E. Ofek, and A. D. Wilson. Haptic retargeting: Dynamic repurposing of passive haptics for enhanced virtual reality experiences. In *ACM CHI Conference on Human Factors in Computing Systems*, pp. 1968–1979. ACM, 2016.

[7] M. Azmandian, R. Yahata, T. Grechkin, and E. S. Rosenberg. Adaptive redirection: A context-aware redirected walking meta-strategy. *IEEE Transactions on Visualization and Computer Graphics*, 28(5):2277–2287, 2022.

[8] M. Azmandian, R. Yahata, T. Grechkin, J. Thomas, and E. S. Rosenberg. Validating simulation-based evaluation of redirected walking systems. *IEEE Transactions on Visualization and Computer Graphics*, 28(5):2288–2298, 2022.

[9] E. R. Bachmann, E. Hodgson, C. Hoffbauer, and J. Messinger. Multi-user redirected walking and resetting using artificial potential fields. *IEEE Transactions on Visualization and Computer Graphics*, 25(5):2022–2031, 2019.

[10] A. Bangor, P. T. Kortum, and J. T. Miller. An empirical evaluation of the system usability scale. *Intl. Journal of Human–Computer Interaction*, 24(6):574–594, 2008.

[11] Y. Chang, K. Matsumoto, T. Narumi, T. Tanikawa, and M. Hirose. Redirection controller using reinforcement learning. *IEEE Access*, 9:145083–145097, 2021. doi: 10.1109/ACCESS.2021.3118056

[12] Z.-Y. Chen, Y.-J. Li, M. Wang, F. Steinicke, and Q. Zhao. A reinforcement learning approach to redirected walking with passive haptic feedback. In *2021 IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, pp. 184–192. IEEE, 2021.

[13] L.-P. Cheng, T. Roumen, H. Rantzsch, S. Köhler, P. Schmidt, R. Kovacs, J. Jasper, J. Kemper, and P. Baudisch. Turkdeck: Physical virtual reality based on people. In *ACM Symposium on User Interface Software and Technology*, pp. 417–426. ACM, 2015.

[14] T. Dong, X. Chen, Y. Song, W. Ying, and J. Fan. Dynamic artificial potential fields for multi-user redirected walking. In *2020 IEEE Conference on Virtual Reality and 3D User Interfaces (VR)*, pp. 146–154. IEEE, 2020.

[15] U. Ehsan, Q. V. Liao, M. Muller, M. O. Riedl, and J. D. Weisz. *Expanding Explainability: Towards Social Transparency in AI Systems*. Association for Computing Machinery, New York, NY, USA, 2021.

[16] T. Grechkin, M. Azmandian, M. Bolas, and E. Suma. Towards context-sensitive reorientation for real walking in virtual reality. In *IEEE Conference on Virtual Reality*, pp. 185–186. IEEE, 2015.

[17] T. Grechkin, J. Thomas, M. Azmandian, M. Bolas, and E. Suma. Revisiting detection thresholds for redirected walking: combining translation and curvature gains. In *ACM Symposium on Applied Perception*, pp. 113–120. ACM, 2016.

[18] S. G. Hart. Nasa-task load index (nasa-tlx); 20 years later. In *Proceedings of the human factors and ergonomics society annual meeting*, vol. 50, pp. 904–908. Sage publications Sage CA: Los Angeles, CA, 2006.

[19] C. Hirt, Y. Kompis, C. Holz, and A. Kunz. The chaotic behavior of redirection–revisiting simulations in redirected walking. In *2022 IEEE Conference on Virtual Reality and 3D User Interfaces (VR)*, pp. 524–533. IEEE, 2022.

[20] E. Hodgson and E. 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.

[21] H. G. Hoffman. Physically touching virtual objects using tactile augmentation enhances the realism of virtual environments. In *IEEE Virtual Reality*, pp. 59–63. IEEE, 1998.

[22] C. Hutton, S. Ziccardi, J. Medina, and E. Suma Rosenberg. Individualized calibration of rotation gain thresholds for redirected walking. In *ICAT-EGVE*, 2018.

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

[24] L. Kohli, E. Burns, D. Miller, and H. Fuchs. Combining passive haptics with redirected walking. In *Proceedings of the 2005 international conference on Augmented tele-existence*, pp. 253–254. ACM, 2005.

[25] D.-Y. Lee, Y.-H. Cho, and I.-K. Lee. Real-time optimal planning for redirected walking using deep q-learning. In *2019 IEEE Conference on Virtual Reality and 3D User Interfaces (VR)*, pp. 63–71. IEEE, 2019.

[26] Y.-J. Li, M. Wang, F. Steinicke, and Q. Zhao. Openrdw: A redirected walking library and benchmark with multi-user, learning-based functionalities and state-of-the-art algorithms. In *2021 IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, pp. 21–30. IEEE, 2021.

[27] R. W. Lindeman, J. L. Sibert, and H. J. K. Hand-held windows: towards effective 2d interaction in immersive virtual environments. In *IEEE Conference on Virtual Reality*, pp. 205–212, 1999.

[28] J. Messinger, E. Hodgson, and E. R. Bachmann. Effects of tracking area shape and size on artificial potential field redirected walking. In *IEEE Conference on Virtual Reality and 3D User Interfaces*, 2019.

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

[30] C. T. Neth, J. L. Souman, D. Engel, U. Kloos, H. H. Bulthoff, and B. J. Mohler. Velocity-dependent dynamic curvature gain for redirected walking. *IEEE Transactions on Visualization and Computer Graphics*, 18(7):1041–1052, 2012.

[31] A. Nguyen, Y. Rothacher, E. Efthymiou, B. Lenggenhager, P. Brugger, L. Imbach, and A. Kunz. Effect of cognitive load on curvature redirected walking thresholds. In *26th ACM Symposium on Virtual Reality Software and Technology*, pp. 1–5, 2020.

[32] A. Nguyen, Y. Rothacher, B. Lenggenhager, P. Brugger, and A. Kunz. Individual differences and impact of gender on curvature redirection thresholds. In *Proceedings of the 15th acm symposium on applied perception*, pp. 1–4, 2018.

[33] N. C. Nilsson, T. Peck, G. Bruder, E. Hodgson, S. Serafin, M. Whitton, F. Steinicke, and E. S. Rosenberg. 15 years of research on redirected walking in immersive virtual environments. *IEEE Computer Graphics and Applications*, 38(2):44–56, 2018.

[34] T. C. Peck, H. Fuchs, and M. 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.

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

[36] S. Razzaque, Z. Kohn, and M. C. Whitton. Redirected walking. In *EUROGRAPHICS*, vol. 9, pp. 105–106. Citeseer, 2001.

[37] R. A. Ruddle. The effect of translational and rotational body-based information on navigation. In *Human walking in virtual environments*, pp. 99–112. Springer, 2013.

[38] F. Steinicke, G. Bruder, J. Jerald, H. Frenz, and M. Lappe. Estimation of detection thresholds for redirected walking techniques. *IEEE Transactions on Visualization and Computer Graphics*, 16(1):17–27, 2010.

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

[40] R. R. Strauss, R. Ramanujan, A. Becker, and T. C. Peck. A steering algorithm for redirected walking using reinforcement learning. *IEEE transactions on visualization and computer graphics*, 26(5):1955–1963, 2020.

[41] J. Thomas, C. Hutton Pospick, and E. Suma Rosenberg. Towards

physically interactive virtual environments: Reactive alignment with redirected walking. In *26th ACM Symposium on Virtual Reality Software and Technology*, pp. 1–10, 2020.

[42] J. Thomas and E. S. Rosenberg. A general reactive algorithm for redirected walking using artificial potential functions. In *IEEE Conference on Virtual Reality and 3D User Interfaces*, 2019.

[43] M. Usoh, K. Arthur, M. C. Whitton, R. Bastos, A. Steed, M. Slater, and F. P. Brooks Jr. Walking¿ walking-in-place¿ flying, in virtual environments. In *ACM SIGGRAPH*, pp. 359–364. ACM Press/Addison-Wesley Publishing Co., 1999.

[44] B. Williams, G. Narasimham, B. Rump, T. P. McNamara, T. H. Carr, J. Rieser, and B. Bodenheimer. Exploring large virtual environments with an hmd when physical space is limited. In *ACM Symposium on Applied Perception*, pp. 41–48, 2007.

[45] N. L. Williams, A. Bera, and D. Manocha. Arc: Alignment-based redirection controller for redirected walking in complex environments. *IEEE Transactions on Visualization and Computer Graphics*, 27(5):2535–2544, 2021.

[46] N. L. Williams, A. Bera, and D. Manocha. Redirected walking in static and dynamic scenes using visibility polygons. *IEEE Transactions on Visualization and Computer Graphics*, 27(11):4267–4277, 2021.

[47] N. L. Williams and T. C. Peck. Estimation of rotation gain thresholds considering fov, gender, and distractors. *IEEE transactions on visualization and computer graphics*, 25(11):3158–3168, 2019.

[48] G. Wilson, M. McGill, M. Jamieson, J. R. Williamson, and S. A. Brewster. Object manipulation in virtual reality under increasing levels of translational gain. In *ACM Conference on Human Factors in Computing Systems*, p. 99. ACM, 2018.

[49] S. J. Wright. Coordinate descent algorithms. *Mathematical Programming*, 151(1):3–34, 2015.

[50] S.-Z. Xu, T. Lv, G. He, C.-H. Chen, F.-L. Zhang, and S.-H. Zhang. Optimal pose guided redirected walking with pose score precomputation. In *2022 IEEE Conference on Virtual Reality and 3D User Interfaces (VR)*, pp. 655–663. IEEE, 2022.

[51] A. Zenner and A. Krüqer. Estimating detection thresholds for desktop-scale hand redirection in virtual reality. In *IEEE Virtual Reality*, pp. 47–55. IEEE, 2019.

[52] M. A. Zmuda, J. L. Wonser, E. R. Bachmann, and E. Hodgson. Optimizing constrained-environment redirected walking instructions using search techniques. *IEEE Transactions on Visualization and Computer Graphics*, 19(11):1872–1884, 2013.

A Semi-Structured Interview Questions

1. Before participating in this experiment, what has been your experience with designing VR applications?
2. Before participating in this experiment, what was your awareness of redirected walking?
3. What are your general impressions of using the manual gains method?
4. Do you have any feedback on how the manual gains method can be improved?
5. What are your general impressions of using the automatic gains method?
6. Do you have any feedback on how the automatic gains method can be improved?
7. How would you imagine a workflow in which you could switch between automatic and manual mode?
8. What kind of applications do you think a tool like this could be used for?
9. Do you have any other questions, suggestions, or comments?

B EA-IK CCD Algorithm

Algorithm 1 EA-IK Cyclic Coordinate Descent

Require: P_V : Virtual path
Require: R: Starting pose of the user
Require: T: Target position
Ensure: *gains*: A list of optimal gain values
  $gains \leftarrow [1.0] * 2 * len(P_v)$: List of translation and rotation gain values for path
  $lastError \leftarrow$ inf
  loop
    $gains \leftarrow coordinateDescent(gains, P_V, R)$
    $P_P \leftarrow forwardKinematics(gains, P_V, R)$: Calculated physical path
    $error \leftarrow ||P_P[N-1] - T||$

    if $error > lastError$ **then**
      $BREAK$
    end if
    $lastError \leftarrow error$
  end loop
  for segment $\in P_P$ **do**
    if outOfBounds(segment) **then**
      $P_I \leftarrow intersectionPoint(segment)$
      $P_V \leftarrow virtualPathRemainder(P_I, P_V)$
      $R \leftarrow P_I$
      $RECURSE$
    end if
  end for