Adaptive Redirection: A Context-Aware Redirected Walking Meta-Strategy

Mahdi Azmandian, Rhys Yahata, Timofey Grechkin, *Member, IEEE*, and Evan Suma Rosenberg, *Senior Member, IEEE*

Abstract—Previous research has established redirected walking as a potential answer to exploring large virtual environments via natural locomotion within a limited physical space. However, much of the previous work has either focused on investigating human perception of redirected walking illusions or developing novel redirection techniques. In this paper, we take a broader look at the problem and formalize the concept of a complete redirected walking system. This work establishes the theoretical foundations for combining multiple redirection strategies into a unified framework known as *adaptive redirection*. This meta-strategy adapts based on the context, switching between a suite of strategies with a priori knowledge of their performance under the various circumstances. This paper also introduces a novel static planning strategy that optimizes gain parameters for a predetermined virtual path, known as the Combinatorially Optimized Pre-Planned Exploration Redirector (COPPER). We conducted a simulation-based experiment that demonstrates how adaptation rules can be determined empirically using machine learning, which involves partitioning the spectrum of contexts into regions according to the redirection strategy that performs best. Adaptive redirection provides a foundation for making redirected walking work in practice and can be extended to improve performance in the future as new techniques are integrated into the framework.

Index Terms—Virtual reality, redirected walking, locomotion, combinatorial optimization

1 Introduction

Many virtual reality applications require users to navigate within large-scale virtual worlds. Allowing the users to walk naturally provides benefits such as efficient navigation [45, 52], improved cognitive maps of the environment [46], and an enhanced sense of presence [57]. However, since virtual environments may easily be larger than the available physical space, providing these types of experiences becomes a difficult challenge. Redirected walking offers a potential solution to this problem that alters the mapping between real and virtual movements, thereby allowing the user to traverse a virtual path that does not fit within the physical space [43]. Translation and curvature gains can be used to manipulate the user's path through the environment, and rotation gains can be used to adjust their orientation during turns.

In the context of redirected walking, one of the key questions is when and to what degree each gain should be applied to maximize effectiveness. A variety of heuristics have been introduced to address this matter, the most common of which is to adjust gains such that the user is gradually steered towards the center of the physical space [42]. More sophisticated planning methods have also been proposed that involve evaluating different gain choices based on how they map the user's predicted virtual path to a real world trajectory and then selecting the gains with the highest utility [9, 34, 65]. The performance of these approaches can be compared in terms of how effective they are at keeping the user within the physical space. Although previous work has noted performance improvements when using planning approaches in specific circumstances, comprehensive selection criteria for different redirected walking strategies remains an open question. Furthermore, the notion of performance itself is still a topic of debate in the redirected walking literature, with various metrics used in different contexts to evaluate and compare redirection techniques.

In this paper, we formalize the concept of a complete *redirected walking system* and present a taxonomy that organizes its components in a hierarchical structure. Starting with the building blocks, we identify redirection *techniques*, how they are used in *heuristics*, together forming redirection *strategies*, and finally expand to create a redirection *system*. With these theoretical foundations established, we introduce *adaptive redirection*, a novel redirected walking system designed as a blueprint for creating a hybrid of redirection strategies. Adaptive redirection is a context-aware meta-strategy—it adapts to the context in which redirection is applied and sits above a collection of strategies, determining which should be used depending on the context. The central tenet of adaptive redirection is that no single redirection strategy is generally superior in all situations. Instead, the system should be able to dynamically shift from one strategy to another, potentially using multiple approaches throughout the course of a single experience.

To fully flesh out the design of the adaptive meta-strategy, we first introduce a categorization for redirection strategies, grouping them into *General*, *Dynamic Planning*, and *Static Planning*. This categorization reflects the fact that not all redirection strategies are always applicable, and this distinction is driven by requirements needed to successfully apply each strategy. By design, adaptive redirection takes a candidate from each category of strategies, and uses adaptation rules to determine which strategy should be activated at each point in time. However, given that previous static planning methods are custom-tailored and not automated, they are not suitable for integration in an adaptive system. To address this gap, we introduce the Combinatorially Optimized Pre-Planned Exploration Redirector (*COPPER*) as the first automated static planning redirection strategy.

To implement a successful meta-strategy, the specific factors that influence the effectiveness of individual redirection strategies (i.e. the context) must be identified, and the variance in their performance under different conditions must be investigated. To achieve this, we formalize the notion of *prediction graph* and describe the range of possible contexts affecting redirection. As the next step, we study the performance of each redirection strategy in a simulation-based experiment and use the results to develop *adaptation rules* that determine which strategy should be activated at a given point in time.

The contributions of this paper include: (1) establishing the theoretical foundations for a redirected walking system; (2) introducing

- *Mahdi Azmandian was with the Institute for Creative Technologies, University of Southern California.*
- *Rhys Yahata is with the Institute for Creative Technologies, University of Southern California.*
- *Timofey Grechkin was with the Institute for Creative Technologies, University of Southern California.*
- *Evan Suma Rosenberg is with the Department of Computer Science & Engineering, University of Minnesota. E-mail: suma@umn.edu*

Manuscript received 6 Sept. 2021; revised 3 Dec. 2021; accepted 7 Jan. 2022. Date of publication 17 Feb. 2022; date of current version 29 Mar. 2022. Digital Object Identifier no. 10.1109/TVCG.2022.3150500

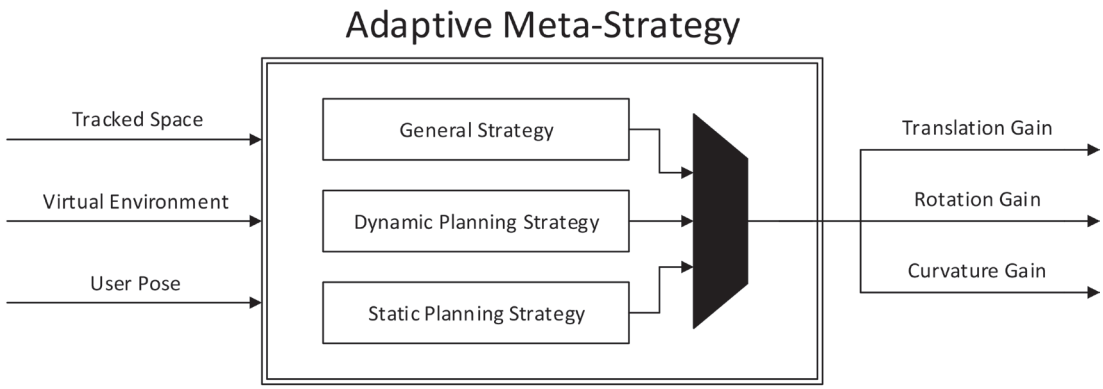

Fig. 1. Schematic for the meta-strategy in adaptive redirection. The system (represented by the black trapezoid) adapts to the context, dynamically selecting between multiple redirection strategies and activating the best choice among the options available.

the adaptive redirected walking system including a categorization of existing redirection strategies along with a definition of context; (3) presenting COPPER, a novel redirection strategy that achieves high performance for pre-determined virtual paths; (4) demonstrating how adaptation rules for the adaptive framework can be derived empirically using machine learning; and (5) analyzing the comparative performance of a representative strategy from each category, offering nuanced insights on their performance factors. The infrastructure presented in this work therefore encompasses many previous research efforts, identifies how they fit in the greater picture, and lays out a roadmap for the future development of redirected walking systems.

2 Background and Related Work

Redirected walking was initially proposed by Razzaque et al. [42] as a potential solution to physical space size limitations when exploring large virtual environments. This locomotion interface is made possible by perceptual illusions that exploit the fact that the human visual system dominates over vestibular cues. In fully immersive VR systems, it is possible to introduce subtle discrepancies between physical and visually perceived movements to "steer" unsuspecting users away from the boundaries of the physical space. In this section, we summarize the prior research most relevant to this work. More extensive literature reviews can be found in recent survey papers on redirected walking [37] and virtual locomotion [1, 38].

2.1 Self-Motion Gains and Perceptual Thresholds

Redirected walking works by adding discrepancies between the user's physical and virtual paths. The methods by which these discrepancies are introduced are called the self-motion gains. Traditionally, three self-motion gains have been used: rotation gain, translation gain, and curvature gain. Translation gains scale virtual translations relative to the real world movement, resulting in faster or slower displacement in the virtual world. Rotation gains apply scaling to rotations, effectively increasing or decreasing the amount of virtual rotation relative to a user's real-world movement. Finally, curvature gain injects a virtual rotation when the user is walking (i.e. primarily translating) in the real environment, which alters the user's trajectory in the virtual world. In order to maintain the desired trajectory, the user will physically rotate in the opposite direction, resulting in a curved real-world trajectory. The original implementations of redirected walking only used rotation and curvature gains [42, 43], and translation gains (which were separately implemented in other locomotion techniques such as the Seven League Boots by Interrante et al. [25]) were later incorporated. More recently, other self-motion techniques have also been proposed. Bending gains combine curvature and rotation gains to make better use of virtual curved paths [26]. Strafing gains are another self-correcting gain in which the user's virtual view is shifted to their left or right as they translate forward [63]. Similar to curvature gain, the user will shift their body in the opposite direction to maintain their desired trajectory, resulting in an angled real world path. Bending and strafing gains are not yet commonly incorporated in redirected walking implementations.

To ensure that the user remains unaware of the manipulation, the gains need to be within the limits of corresponding perceptual detection thresholds. These thresholds are unique for each gain and have been estimated through several empirical studies. The seminal study that is most commonly referenced was performed by Steinicke et al. [49]. Other studies have looked at how changing translation speed affects the thresholds, how combining gains affect the thresholds [18], how field of view affects the thresholds [2, 12, 61], and the relationship between thresholds and gender [61]. It is also widely accepted that these detection thresholds vary considerably among individuals, and Hutton et al. developed a method for quick individualized calibration of a user's detection thresholds for rotation gains [24]. There is currently a lack of research regarding a user's sensitivity to changing gain levels. Congdon et al. is the first to formally explore this concept, and they determined that gain smoothing is a necessity in RDW applications [14], but more research is needed to obtain a better understanding. Schmitz et al. proposed "threshold of limited immersion" (TLI) as an alternative to the traditional gain detection threshold [47]. TLI measures when user's lose their sense of immersion due to gain levels, instead of merely noticing them. The results suggest that RDW becomes more effective when TLI is used instead of traditional detection thresholds. However, further research noted that experiences that utilize TLI caused a measurable increase in the user's cybersickness, which usually subsided shortly after exposure [20].

A variety of methods to augment redirected walking have been introduced over the last decade. Sound has been investigated to enhance, or even replace, traditional redirected walking techniques [16, 44, 48]. Researchers have also studied techniques to increase redirection gains during eye blinks or saccade movements [13, 28, 36, 53]. Other techniques such as using haptic cues [31] or transcranial direct-current stimulation [27] have also been proposed. Additionally, a distinct but related technique for redirecting user vertically, known as redirected jumping, has also recently emerged as a new research topic [19, 30, 33, 62]. However, our proposed adaptive redirection framework is focused on redirected walking along the horizontal plane, and vertical movement is beyond the scope of this paper.

2.2 Boundary Collision Recovery

Because the application of self-motion gains is limited by the detection thresholds, there will inevitably be moments when the system cannot keep the user's path within the physical tracked space. When this occurs, the system must intervene to keep the user from exiting the boundaries of the physical space. One approach involves temporarily exceeding the detection thresholds until the risk of exiting the tracked space has passed [11, 39, 51]. This will likely cause the user to notice the redirection and may even result in negative experiences such as cybersickness, although the effects on the user's experience are

largely unstudied. More commonly, collisions with the boundary of the physical space are prevented by using reorientation events [41]. This approach is essentially a fail-safe mechanism that stops the user from leaving the boundary, requiring them to perform a reorientation task before continuing to walk. Resets are a type of reorientation event that often involve instructing the user to perform an in-place virtual rotation that is scaled with rotation gains [58]. For example, the *face-center* reset scales the rotation such that when the reset task is complete, the user faces towards the center of the physical space. Reorientation events usually pause the experience while the reorientation task is underway, which can be disruptive for the user. However, it is possible to construct the reorientation events in such a way that they are contextually relevant to the larger experience, resulting in a much lower chance that the user's sense of presence will be broken [17, 40].

2.3 Steering Algorithms

A core component of the taxonomy proposed in this paper is a *redirection strategy*. The closest equivalent to this idea in the existing literature is a steering algorithm. It is the job of the steering algorithm to control the magnitude of the gains applied to the user's movements. Ideally, the steering algorithm will produce a virtual-physical path pair that is more spatially efficient than walking without redirection.

The most widely used redirected walking algorithms in the literature are Steer-to-Center (S2C) and Steer-to-Orbit (S2O), both proposed in the original redirected walking paper [43]. Both of these techniques are essentially greedy approaches that rely only on the current user's physical location and direction of travel within physical space to determine the best choice of gains at each calculation step. As the system can only react to the user's current state, these algorithms are generally referred to as reactive algorithms. S2C uses rotation and curvature gains to steer the user toward the center of the physical space, while S2O aims to steer the user along an elliptical trajectory around the center. In recent years, two new forms of reactive algorithms have been introduced, both of which are focused on complex physical environments that may contain non-convex boundaries or obstacles such as furniture. The first category uses artificial potential fields to provide a better heuristic [11, 32, 55] and the second uses machine learning techniques to provide optimized solutions for a given physical environment [15, 29, 50].

Predictive algorithms have also been proposed that leverage information of the user's possible future path in the virtual environment, such as FORCE [65] and MPCRed [34]. These approaches involve investigating the outcome of the user's future movements to select gains according to some utility function. Possible actions are typically represented as a *prediction graph* that expresses the user's trajectory options in the near term. Constructing the graph often relies on a representation of the virtual environment's layout in the form of a bidirectional *layout graph*. This annotation provides a high-level abstract description of possible paths that a user can take. Edges of such a graph represent the general expected direction of travel for possible paths in a particular area of the virtual environment, while nodes correspond to decision points where potential paths diverge. The prediction graph may need to be created manually by the experience designer; alternatively, algorithms have also been introduced to automate the creation of the layout and prediction graphs [6, 64]. Recently, Williams et al. introduced a new algorithm, ARC, which has properties of both reactive and predictive algorithms [59]. ARC seeks to align the virtual and physical geometries instead of generating a prediction graph.

2.4 Performance Evaluation

The performance of a redirected walking system depends on a variety of interacting factors including user behavior, physical space dimensions, the structure of the virtual environment and the type of virtual path, as well as the internal parameters such as perceptual thresholds and the reorientation method. This complexity makes comparative evaluation of these algorithms a non-trivial problem. Azmandian et al. [5] introduced a systematic method of evaluation that controlled for the most salient factors impacting performance and proposed using a simulated user to enable experiments with large numbers of lengthy trials, both of which cannot be feasibly accomplished via user studies. A key characteristic of this system was accounting for the effect of the physical space and boundary collisions on the simulated user, requiring a reset action instead of permitting simulated users to exceed these limits. Furthermore, the frequency of these interruptions were used as the primary measure of performance instead of the previously proposed method of computing the minimum physical space dimensions that could fully contain a redirected user without ever reaching a boundary.

One of the fundamental questions that remains largely unanswered is how different redirection strategies perform across the various conditions that may be encountered in a virtual reality experience. Although subsets of this problem have been investigated in isolation [5, 22, 32, 55], our conceptual framework considers the broad spectrum of approaches, also factoring in the characteristics of prediction graphs. This work therefore establishes the foundations for understanding how the performance of redirection strategies will be influenced by external factors, which approach works best for a given set of conditions, and how an overall redirection walking system can be expected to perform.

3 Design Hierarchy of a Redirected Walking System

In this section, we provide the theoretical foundations for organizing a redirected walking system into a hierarchical structure. At the very top, we have a redirection *system*, which can make use of one or more *strategies*. A strategy would then make use of one or more *heuristics*, which in turn each make use of a set of *techniques*. We will now introduce each of these concepts in order to formalize the abstract structure of a redirected walking system.

3.1 Redirection Techniques

A redirection technique is a mechanism for manipulating the user's trajectory. Individual techniques are essentially the atomic building blocks for redirected walking. The three most common techniques are translation gains, rotation gains, and curvature gains, which have been already defined in section 2.

3.2 Redirection Heuristics

A heuristic is any mechanism or algorithm that uses one or more redirection techniques to fulfill a certain task or move towards an objective. The two most common forms of heuristics within the literature are *redirection* heuristics and *reorientation* heuristics. As their names suggest, a redirection heuristic aims to keep users within the physical space, while reorientation heuristics use techniques to reorient the user away from the boundary. With this formulation, the steer-to-center "steering algorithm" would be considered a redirection heuristic that uses rotation and curvature gain techniques to meet the objective of steering the user towards the center of the space. Similarly, the face-center reset is a reorientation heuristic that uses the rotation gain technique to reorient the user away from the boundary to face the center of the space.

3.3 Redirection Strategies

A strategy is a mechanism that uses one or more heuristics in order to ensure users remain within the boundaries of the physical space. More specifically, a *redirection strategy* is a function that takes in a set of arguments about the current state, selects and applies redirection heuristics, and returns a set of values for gains (rotations and translations) to be applied immediately on the next computation frame (Figure 2).

3.3.1 Inputs

The input of a redirection strategy can be grouped into either *spatial* input or *user* input. Spatial input comprises the information describing the state of the user, physical space, and virtual environment at a given moment. The user input, on the other hand, describes the set of preferences that tailor the experience to best suit the user. Note that not all these inputs are required for a redirection strategy to function.

Spatial Input

(a) **physical space dimensions:** exact dimensions of the physical space (e.g., 5×5 meters for an HTC Vive tracking space).

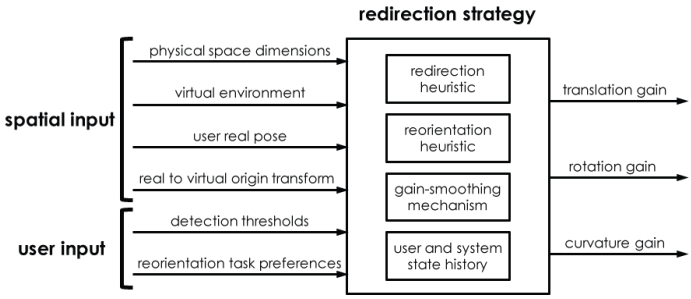

Fig. 2. The components of a redirection strategy. Inputs are grouped into spatial and user inputs which are used to determine the output gains.

(b) **virtual environment:** 3D model of the virtual world along with relevant data such as specific targets or experience restrictions.
(c) **user real pose:** the user's position and rotation in the real world coordinate system (and potentially additional information such as eye-tracking data and feet position).
(d) **real to virtual origin transform:** the transformation matrix mapping the origin of the real world to the virtual world origin.

User Input

(a) **detection thresholds:** the extreme values for each type of gain that could be applied [49]. Conceivably, this could include more complicated input that would account for velocity-dependent thresholds or other factors [18, 35]).
(b) **reorientation task preferences:** specific reorientation methods preferred by the user (e.g., text/voice prompt, distractor, etc.).

3.3.2 Outputs

The outputs of a redirection strategy include:

(a) **redirection gains:** the translation, rotation, and curvature gain values to inject in the next computation frame.
(b) **reorientation signal:** discrete event that triggers or disables a reorientation task.

It is worth mentioning that the gains from the output influence the *real to virtual origin transform*, which will again be fed back into the function in the next computation frame.

3.3.3 Internal Components of a Redirection Strategy

The components of a redirection strategy may include:

(a) **redirection heuristic:** the main component that guides the decision-making for applying gains.
(b) **reorientation heuristic:** the method for executing the reorientation task using gains.
(c) **gain-smoothing mechanism:** a component that ensures gain values change gradually to prevent detection and user discomfort.
(d) **user and system state history:** a summary of relevant information from the recent computation frames.

3.4 Redirected Walking System

A redirected walking system defines a complete locomotion interface that enables natural walking in a virtual environment within a smaller physical space by using one or more redirection strategies. In its simplest form, a redirected walking system can be entirely composed of a single redirection strategy, but in more advanced cases, multiple redirection strategies can be integrated into a *meta-strategy*. A meta-strategy resides above other redirection strategies and governs how they can be integrated in order to determine the final output. The paradigm presented in this work is an example of a redirected walking system and, to the best of our knowledge, the first ever to combine multiple redirection strategies using an *adaptive meta-strategy*. Measures based on reset frequency, such as the average distance walked between resets, are the primary metrics for evaluating the performance of a redirected walking system.

4 The Adaptive Redirection Meta-Strategy

The Adaptive Redirection Meta-Strategy is a function that determines the most effective redirection strategy for a given physical space based on the context, which we define using a prediction graph measured at the user's location. In this section we will explore the domain, range, and mapping of inputs to outputs (i.e. adaptation rules) for this function.

4.1 A Taxonomy of Redirection Strategies

Not all redirection strategies can be applied in all circumstances, and their applicability is subject to the availability and properties of the prediction graph. We can therefore, categorize redirection strategies according to the range of circumstances (i.e. subset of contexts) in which they can be applied.

General strategies can be applied to any scenario and do not depend on user predictability. Therefore, they neither rely on a prediction graph nor are able to leverage one even if available. These strategies utilize only current state information, sometimes employing only a single heuristic that is akin to a greedy algorithm. For example, common steering algorithms such as steer-to-center and steer-to-orbit are examples of both a redirection heuristic and a general strategy.

Dynamic planning strategies such as FORCE [65] and MPCRed [34] rely on the availability of a prediction graph. They use the prediction graph to dynamically plan a short-term redirection strategy. This plan is periodically updated as the user explores the environment. These strategies typically require some level of user predictability, but do not strictly require a linear path through the virtual environment (i.e., they can support branching pathways).

Static planning strategies have a very strict dependency on user predictability—they require a prediction graph that does not contain any branches. This means the user's virtual path is completely predictable. This feature allows the strategy to plan the entire route in advance, thus choosing the best set of gains. Any strategy that is designed or tailored for a specific path can be considered a static planning strategy. Razzaque's zig-zag fire drill demonstration from the original redirected walking paper is an example of this approach [43].

When designing adaptive redirection, we assume at least one candidate strategy is selected from each category. Ideally, the chosen strategies would be determined based on one of more factors like performance, efficiency, or ease of implementation. Adaptive redirection determines a set of rules to switch between these strategies such that the most effective one is selected at any given time. This does not mean a single strategy is selected for each virtual reality experience. Instead, within a single experience, the activated strategy will not necessary stay constant and may change multiple times as the user moves throughout the virtual environment. The adaptive framework is therefore flexible, and new redirection strategies can be readily integrated without changing the high-level system architecture.

4.2 Prediction Graph Properties

The input for an adaptive redirection meta-strategy is a prediction graph. This graph is used to first determine which strategies are applicable. If multiple strategies can be used, it is further examined to determine which of the contending strategies would perform best. Given the proposed classification of strategies, three distinct scenarios can be identified: 1) no prediction graph is available, in which case the general redirection strategy will be selected by default, 2) the prediction graph contains branches, in which case both general and dynamic planning strategies are viable, and 3) the prediction graph has no branches, in which case all three approaches are applicable. The hypothesis is that the properties of the prediction graph are sufficient for determining the most effective approach in scenarios 2 and 3. Therefore, it is essential to express the prediction graph properties with variables that capture its characteristics. In this work, we characterize prediction graphs by the distribution of the *segment length*, *turn angle*, and *branching factor*.

A segment is a portion of the complete virtual path which starts at one waypoint and ends at another. Consequently, the segment length is the distance between the start and end waypoint. The segment length distribution for a given prediction graph is a continuous probability

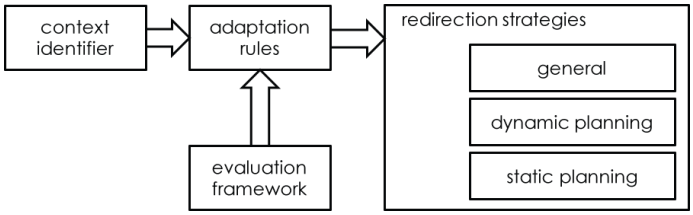

Fig. 3. The pipeline for developing the adaptive redirected walking system. Adaptation rules are constructed by training on data provided by the evaluation framework (i.e. simulations) such that the appropriate redirection strategy can be selected for the given context.

distribution for the segment lengths of that graph. When the user finishes traversing a path segment, they turn in place to face the next waypoint. The amount of rotation is known as the turn angle. Similar to the segment length distribution, the turn angle distribution is a continuous probability distribution for the turn angles of a given graph. The branching factor for a given waypoint is defined as the number of waypoints that it connects to. The distribution assigns a probability value to each possible non-zero integer branching factor.

4.3 From Prediction Graph to Most Effective Strategy

The final component of the adaptive redirection meta-strategy is the mapping from the inputted prediction graph to the most effective redirection strategy. This problem can be formulated as a classification of the prediction graph space: labeling regions with the strategy that best performs within it. In order to accomplish this, we sample the domain, label each sample with the strategy that performs the best, and use machine learning to train a classifier that partitions the prediction graph space based on the samples' labels. Additionally, a rule-based classifier will be used so that the partitioning is expressed as a set of rules that can be easily adapted to programming logic. The pipeline of the adaptive redirected walking system can be seen in Figure 3.

5 COPPER

In this section, we introduce the Combinatorially Optimized Path-Planned Exploration Redirector (COPPER). COPPER is a novel automated approach to static redirection planning that is applicable to any deterministic virtual path. COPPER assumes the user's virtual path consists of going from one point of interest to another in succession, without any detours or major deviations. The algorithm consists of two main components: *planning* and *execution*. The planning component is an offline search for the optimal mapping of the user's virtual path to a trajectory within the physical space. The execution component of COPPER is performed at runtime and aims to ensure that its plan is executed as expected by dynamically adjusting gains to compensate for the user's locomotion behavior, keeping the redirection results similar to the expected planned path. The online component of COPPER is based on the counter-deviation technique introduced by Azmandian et al. [3, 4]; therefore, this paper focuses on the offline planning component.

Static planning takes a given virtual path, expressed as a series of waypoints, and calculates the series of gains that result in the corresponding optimal real trajectory (i,e., incurs minimal resets). The simple approach to solving this problem would be to enumerate over all possible real trajectories that are created by combinatorially applying every possible gain value to each virtual path segment. This brute force strategy would then pick the gain values that yield the best trajectory. Therefore a sequence of gain values that would minimize the number of incurred resets would be considered optimal. Since the set of gain values is continuous, there are an infinite number of possible combinations that the brute force approach would have to compare, making this computationally infeasible. To solve this issue, COPPER reduces the number of compared trajectories by pruning the search space. We hypothesize that these simplifications will make this problem computationally tractable and would still allow COPPER to find an optimal or near-optimal solution.

COPPER's approach to finding the best choice of gains is to explore all possible gain choices at the same time with an algorithm similar to a breadth-first-search. First, we break down the virtual path into smaller segment of turns and walks. Then we begin to process these segments one at a time. For each segment, we entertain all possible gain choices (causing branches in the search tree) which may be in the form of rotation, translation, or curvature gains and also possibly resets if a boundary is reached. Therefore, as each segment is consumed, more and more possibilities are created, equating to all the possible gains choices being considered. COPPER also performs pruning in a manner that aims to keep the problem manageable without discarding elements that would preclude it from finding an optimal solution.

5.1 Simplifications and Assumptions

COPPER makes two assumptions while searching for the optimal set of gain values. First, it is assumed that the user will walk from one waypoint to the next in a straight line, while directly facing the destination waypoint. Additionally, when the end waypoint is reached, the user will rotate in place to face the subsequent waypoint before walking towards it. This simplification allows for easy calculation of the predicted user's real trajectory under the influence of redirection. Furthermore, the online execution component of COPPER, which includes a counter-deviation mechanism [3], will help ensure that the user's locomotion behavior will not cause a divergence from the predicted outcome despite this simplification.

The other simplification is made to the range of gain values and reset angles. Rather than using a continuous range, COPPER selects a finite subset of discrete values. While this may cause COPPER to find a solution that is sub-optimal, we can improve the outcome by increasing the set of values sampled from each range. In this work, we select 3 values for each of the different types of gains, and 5 values for the reset angle. For instance with translation gains we only consider the 3 values for minimum, maximum, and no gain applied. As for resets, the target user orientations we consider angles 0, $\pm$30, $\pm$60 deg. with the boundary normal (e.g. 0 deg. requires the user to face perpendicular to the wall, facing into the physical space).

5.2 Data Structures

While searching for the optimal trajectory, COPPER uses two data structures to keep track of contending potential solutions. The first is an *action*, representing a decision made by the planning strategy containing parameters used at a portion of the virtual path. There are 3 types of actions, each corresponding to a specific movement required to traverse the virtual path. A *turn action* is selected for the portions of the virtual path when the user reaches a waypoint and has to rotate in place to face the next waypoint. Therefore, the turn action would contain the rotation gain applied during the turn. *Walk actions* correspond to the walking segments from one waypoint to another. They contain the fixed translation and curvature gain that is applied by the static planning redirection strategy while the user moves along the virtual path segment. If COPPER predicts that the user is unable to reach the end waypoint without crossing the boundary of the physical space, it will create a *reset action* to determine the reset angle that the reorientation task should use when the user needs to be reset.

In order to keep track of all the decisions its made, COPPER uses a data structure called a *search node*. A search node encapsulates the information required to process virtual path segments at each search iteration. It includes the user's current pose (position and orientation) in the real world, the *score* of the planned real trajectory, and the sequence of actions taken up until this iteration.

5.3 Performing the Search

Before performing the search, the virtual path is converted from a list of consecutive waypoints to an ordered sequence of *moves*. These describe the type of movement a user will perform when traversing the virtual path, consisting of *rotate-to-face* and *walk-to-next* moves.

A rotate-to-face move is the movement a user has to perform when turning in place to face the next waypoint. This move maps directly to a turn action. Therefore, when processing a rotate-to-face move,

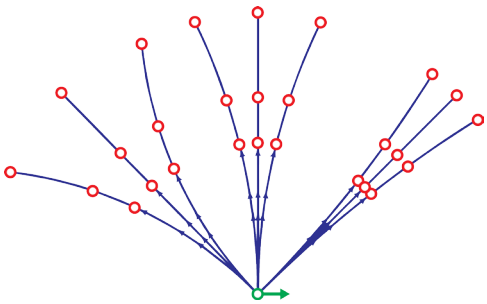

Fig. 4. Illustration of how COPPER expands a node when processing the virtual path. In this example, the user (green) makes a 90-degree left turn and walks forward in the virtual environment. Based on different combinations of translation, rotation, and curvature gains, the user's trajectories in the physical world (blue) result in 27 different possible end positions (red circles).

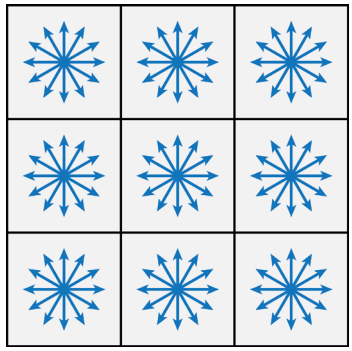

Fig. 5. A simplified example of configuration classes used to discard similar nodes. The node's real world position is discretized by a grid in the physical space. The orientation is also discretized by partitioning the space of possible angles. By retaining at most one node from each class, we ensure the number of nodes throughout the search is bounded (by the number of classes). In our experiment, a 10x10 grid was used with orientation intervals of 15 degrees.

COPPER will generate 3 search nodes, each containing a turn action with a unique value from the set of rotation gains.

The other movement the user will perform is walking from one waypoint to the next in a straight line, which is considered a walk-to-next move. This move will always map to at least 9 walk actions, one for each of the unique translation and curvature gain value combinations (Figure 4). If the user's predicted trajectory does not fit within the physical space, reset actions will be added as needed, thus increasing the number of created search nodes by a factor of 5; the five-fold increase is due to the number of possible reset angles. COPPER models the user's reaction to a reset prompt by assuming the user will continue to walk for 0.75 meters before coming to a complete stop. That reaction time is factored into the remaining path segment before creating additional walk actions. However, if it is calculated that the user will reach the destination before responding to the reset, the action is ignored. Therefore, for a virtual path segment that requires one reset, COPPER will create up to $9 \times 5 \times 9$ search nodes,

When processing a move, COPPER selects an appropriate action and applies it to all the current search nodes. As each search node is evaluated, new child nodes are created for all the possible values of the action. The planning strategy calculates the user's trajectory and its score for each action value. The set of produced search nodes is then pruned before repeating the search process on the next move.

5.4 Pruning

As each move is processed, the number of nodes at each iteration rapidly increases. To keep the node list manageable, search nodes that are similar need to be discarded. In order to achieve this, we classify nodes such that they can be split into groups with similar traits. The intuition behind the pruning is that any two nodes with similar ending poses and scores are equivalent, regardless of the sequence of gains (i.e. the real trajectory traversed) that lead them to the end pose. Furthermore, if an optimal trajectory can be created by expanding one of these nodes, it can also be found by expanding the other. Therefore, retaining only one is sufficient for finding an optimal solution.

To achieve this pruning, as moves are processed, their search nodes are assigned to a *configuration class* based on the calculated user pose in the physical space. This essentially discretizes the set of possible user poses by creating bins. Therefore, a configuration class can be represented as a tuple which contains a section of the physical space along with a continuous range of user orientations. The number of configuration classes depends on the granularity of discretization. For this experiment, our 10×10 meter physical space is converted to a grid of 1×1 meter cells, and the orientation range is divided into 15 degree intervals (Figure 5). This yields a total of $10 \times 10 \times 24 = 2400$ configuration classes.

Additionally, if the physical space shape is symmetric, further reduction can be applied to the search nodes. For example, a rectangular physical space can be divided into 4 equal quadrants. Any arbitrary configuration class in one quadrant can be transformed to a similar one in another quadrant by rotating it around the center of the rectangle by a multiple of 90 degrees. Therefore, in this example, this bijective mapping allows COPPER to reduce the number of configuration classifications by 75% (a total of 600 classes for this experiment).

After each search node is grouped into its respective configuration class, only the node with the best score is kept from each category. This puts an upper bound on the total number of search nodes at each search iteration—there can only be as many search nodes as there are unique configuration classifications (Figure 6). Although it may seem like potential optimal solutions are discarded, the rationale behind this is that nodes belonging to the same configuration classification offer the same spatial advantages for redirection. We assume that if an optimal path can be found by using one of the disposed nodes, a similar optimal path can also be obtained using the retained representative. Since the planning component of COPPER is run offline, execution speed is not a concern, thus the granularity for all the discretizations can be adjusted to find more possible solutions, increasing the likelihood of including an optimal solution, provided that the system has sufficient memory. Configuration classes are a trade-off between calculation complexity/feasibility and optimality.

5.5 Utility Function

The utility function assigns a score to each search node based on its current and previous actions. This score is a metric for how (un)desirable the planned trajectory is. In this implementation of COPPER, the only factors that affect the score are the number of resets encountered along the path and the number of near-reset situations. A near-reset situation is when the user's real world end position after a virtual segment is in one of the configuration classes that represents a cell on the border of the physical space. The score of a search node is defined recursively as the sum of the parent node's score, the number of resets encountered for this move, and an $\varepsilon \ll 1$ penalty if the planned trajectory results in a near-reset. Since a lower score indicates a more desirable trajectory, after all the moves have been processed, the planning strategy will select the remaining search node with the lowest score. Although our implementation only scores search nodes based primarily on resets, other factors like total amount of redirection could also be factored in.

5.6 Offline Pre-Calculations

Static planning solves the problem of finding an optimal redirection strategy when both the user's starting pose in the real world and the virtual path to be traversed are known in advance. However, COPPER can also be extended to operate in conditions where either of these elements are not known. Examples of this include: a) Designing a walkthrough without a fixed initial initial pose (this flexibility can

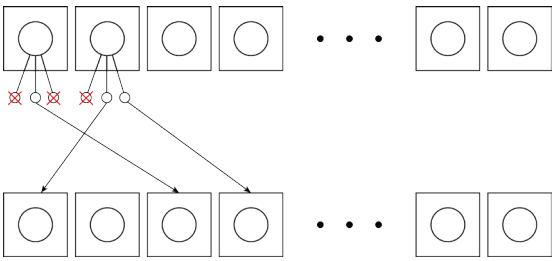

Fig. 6. Configuration class representatives in consecutive iterations. The top row represents the input configurations. Each circle is a representative of a configuration class. Once a portion of the virtual path is processed and the tree is expanded, only the best representative leaf node for each output class is retained for the next iteration. Note that each class would have at most (and possibly no) representatives.

allow finding a more optimal path); b) Recovering from excessive user deviation by rerunning COPPER's planning; and c) Using COPPER as part of adaptive redirection which would require instantly switching to COPPER at runtime.

The solution to dealing with this uncertainty is to pre-calculate the optimal strategy for all possible scenarios. By having a lookup table, we can match the specific user pose and virtual path at runtime and quickly access the pertinent optimal strategy. The table contains one entry for a representative of each configuration class, and at runtime the user's actual pose is matched to the representative of its class. As for the virtual paths, the table has entries for all sub-sequences in the environment's virtual path that have no branches, and at runtime the matching sequence is used to find the optimal strategy.

6 Experiment: Constructing Adaptation Rules

The goal of this experiment is to establish a methodology to derive the rules for selecting the best redirection strategy in adaptive redirection. This is achieved by using machine learning to create a classifier that maps prediction graphs to corresponding best-performing strategies. The training data for the classifier was generated from a simulated user.

6.1 User Simulation

To measure the performance for each condition, we performed simulated experiments using a modified version of the simulated user functionality included in the open-source Redirected Walking Toolkit [7]. In this design, a walking user was simulated by an autonomous agent. The agent was programmed to traverse the virtual path by walking toward the next waypoint with a constant linear velocity of 1 m/s while maintaining its heading toward the waypoint. Upon reaching a waypoint, the simulated user would stop and turn in place with an angular velocity of 90 deg/s to face the next waypoint. When the user's distance from a boundary dropped below 1 meter, a reset would be triggered. At this point, the user would stop and rotate in place at 90 deg/s until the reset task was complete.

For this experiment, we added the functionality to simulate a user's delayed response to a reset prompt. This was modelled as a 0.5 second delay in reacting to a reset, and a constant linear deceleration that caused the user to come to a full stop in 0.5 seconds. If the waypoint was followed by a branch in the virtual path, the simulated user would randomly select one of the possible path options. For this study, no noise was introduced to the simulated user's translation and rotation, thus guaranteeing that the simulated user would walk exactly along the virtual path. The simulation platform emulated a framerate of 60Hz and was implemented in the Unity game engine [56].

6.2 Procedure

The redirection strategies investigated were steer-to-center (S2C), FORCE, and COPPER, all of which were paired with the face-center reset technique. We also measured the performance for a baseline "No Redirection" strategy. In this condition, no redirection gains were applied and only the reset would prevent exceeding the boundary limits.

Factor	Analysis	None	S2C	FORCE
seg. length	$F(3,3341)$	1183.84*	11.76*	279.33*
turn angle	$F(3,3341)$	1917.34*	798.32*	46.22*
branch prob.	$F(10,3341)$	1.10	1.29	213.66*

Table 1. Statistical results (F-values) of the linear regression analyses for the No Redirection, Steer-to-Center, and FORCE strategies. $^*p < .001$

To generate prediction graphs, all three defining characteristics were varied. The segment lengths (in meters) were uniformly sampled from one of the following ranges: $(0.01, 2.5)$, $(0.01, 5)$, $(0.01, 7.5)$, and $(0.01, 10)$. The absolute turn angle distributions were uniformly sample from ranges: $(0°, 45°)$, $(0°, 90°)$, $(0°, 135°)$ and $(0°, 180°)$. The branching likelihood ranged from 0 to 1 in increments of 0.1. The number of the waypoints the simulated user cleared was adjusted based on the segment length distribution such that the expected virtual path would be 500 meters. For instance, a segment length distribution of $[0.01, 2.5]$ would have the user traverse a path of 400 waypoints.

For each trial of this experiment, the simulation began by placing the user at the center of a $10m \times 10m$ physical space facing the positive Z direction. The user's prediction path would dynamically expand after each waypoint was cleared to provide a 3-waypoint-deep horizon of navigation options. The choice of 3 levels deep was selected based on the FORCE algorithm functionality. During each trial, the number of resets were tallied and the overall virtual distance travelled was logged. Each condition was repeated a total of 20 times.

To gauge performance we use *reset-per-distance*, which is calculated by dividing the reset count by the virtual distance traveled. This is essentially the reset count normalized by virtual path length because not all randomly generated paths have the exact same length. Therefore, a lower reset-per-distance corresponds to superior performance.

7 Results

The state of the prediction graph dictates the applicability of each strategy and in turn determines what options are available for adaptation. Based on this, we identify three cases: a) the prediction graph is not available, for which S2C would be the only option, b) the prediction graph is available but contains branches, in which case we must select between S2C and FORCE, c) the prediction graph has no branches, making all strategies viable. Case *a* requires no adaptation rules, so we examine cases *b* and *c* in this section.

7.1 Branching Prediction Graph

We first examine the performance of the redirection strategies across all prediction graph properties (Figure 7). To simplify the analysis, we modeled effects on each redirection strategy separately. The goal was to broadly explore how reset counts per unit distance were affected by path characteristics, and to understand relative importance of each factor rather then derive a precise functional relationship. Therefore, we modelled the relationship between reset count and explanatory variables as main effects, ignoring any possible interactions.

First, we constructed a linear model for the No Redirection strategy, with reset per distance as a continuous response variable and max segment length, max turn angle and branching probability as explanatory main factors (see Table 1). Max segment length and max turn angle were significant predictors of reset per distance, while branching probability was not a significant predictor. The general trend was for the number of resets per unit distance to decrease with max segment length; for example, for $10m$ length, max segment the reset per distance decreased by 28.9% vs. $2.5m$ max segment length case ($p < .001$). The reset per distance also increased with max turn angle. For example when comparing the 45° and 180° max turn angle situations, the number of increased by as much as 57.5% in the latter case ($p < .001$).

For the S2C strategy, a linear model with reset per distance as the response variable and max segment length, max turn angle and branching probability as main effects explanatory variable revealed max segment length and max turn angle as significant variables while branching probability was not a significant predictor. The general trend was for

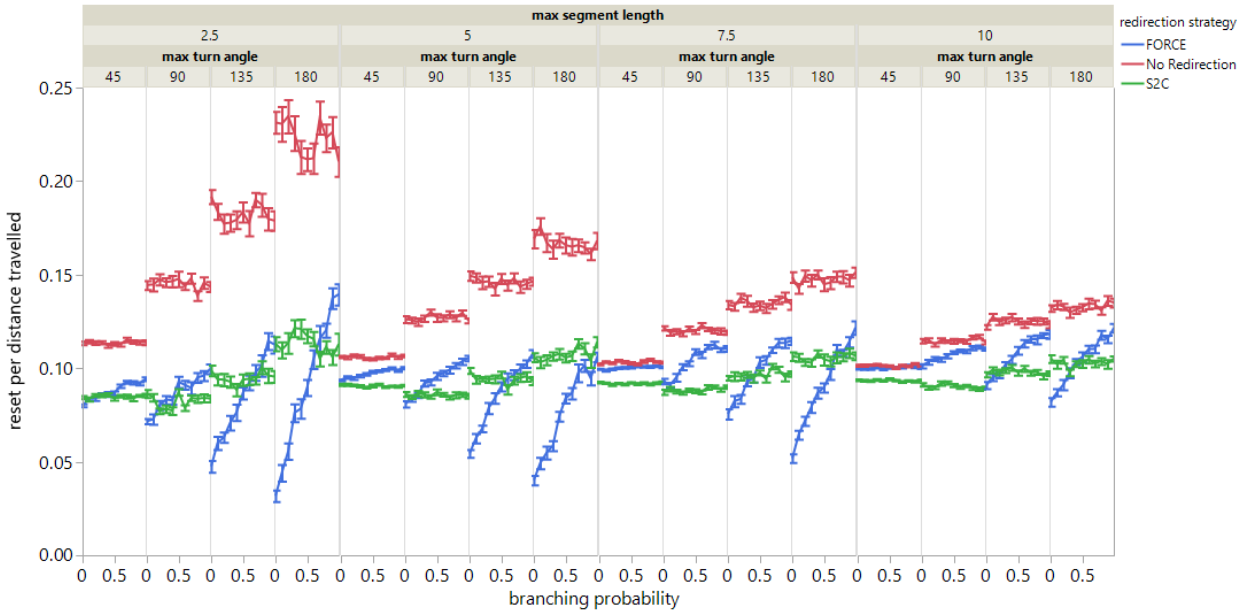

Fig. 7. Performance comparison of contending redirection strategies across prediction graph properties for the branching prediction graph scenario. The Y axis shows normalized reset values, therefore lower values indicate better performance. Error bars show standard error. We can see how each redirection strategy interacts differently with the factors, resulting in S2C (green) being superior in some cases and FORCE (blue) in others. The No Redirection (red) results show how the prediction graph can influence performance even when no redirection is used.

the number of resets per unit distance to increase with max segment length; for example for $10m$ length max segment the reset per distance increased by 2.5% vs. $2.5m$ max segment length case ($p < .001$). For the max turn angle, reset per distance decreased by 4.4% from the 45° to 90° ($p < .001$), but from 90° to 180° it consistently increased up to a factor of 24.4% ($p < .001$).

Finally, for the FORCE strategy, a linear model with reset per distance as the response variable and max segment length, max turn angle and branching probability as main effects explanatory variable revealed that max segment length, max turn angle, and branching probability were all significant predictors. The reset per distance measure increased by 19.5%, with max segment length changing from 2.5 to 10 ($p < .001$). The reset per distance measure decreased with max turn angle. For example, when comparing the 45° and 180° max turn angle situations, the number of decreased by as much as 6.3% in the latter case ($p < .001$). For branching factor, reset per distance increased by 47.6% as branching probability increased from 0 to 1 ($p < .001$).

In order to determine adaptation rules for the branching prediction graph case, we labeled each condition according to the strategy that performed best. This was achieved by measuring the mean of reset-per-distance of each strategy for each tuple of branching factor, segment length, and turn angle, and labeling the tuple with the strategy with the lowest mean reset-per-distance. We then used a PARTS rule-based classifier with a 10-fold cross-validation to train and assess the accuracy of our model. PARTS outperformed other rule-based classifiers with an accuracy of 98.9%. However, using a Random Tree classifier can also fully fit the model and achieve a 100% accuracy using a tree with 47 nodes. The rules derived by PARTS can be seen in Table 2, identifying in for each region of the prediction graph space what strategy should be used. These rules are well in line with the intersection points between performance graphs in Figure 7.

7.2 Non-Branching Prediction Graph

We now focus on cases where the prediction graph has no branches. In this case, all strategies are applicable. The comparative performance of each strategy can be seen in Figure 8.

Similar to the previous analysis, we first investigated the effect of each prediction graph property on COPPER. To achieve this, we constructed a linear model with reset per distance as a continuous response variable and max segment length and max turn angle as explanatory main factors. Max segment length ($F(3,304) = 299.93, p < .001$) and max turn angle ($F(3,304) = 1450.77, p < .001$) were both significant predictors of reset per distance. The general trend was for the number of resets per unit distance to increase with max segment length. For example, for the $10m$ length max segment, the reset per distance increased by 155.0% vs. $2.5m$ max segment length case ($p < .001$). The reset per distance also increased with max turn angle. For example, when comparing the 45° and 180° max turn angle situations, the number of decreased by as much as 88.2% in the latter case ($p < .001$).

To directly compare redirection strategies while accounting for the properties of the path, we constructed a linear model with reset per distance as an explanatory variable. The model included max turn angle, max segment length, and redirection strategy as predictors. We considered only main effects, and interaction terms were suppressed. The model revealed that redirection strategy was the most powerful predictor of reset count per unit distance ($F(3,1215) = 1170.25, p < .001$), followed by max segment length ($F(3,1215) = 2.99, p = .029$). Max turn angle was not a significant predictor of performance ($F(3,1215) = 2.09, p = 0.1$). The post-hoc analysis using Tukey's HSD shows that after accounting for the effects of max turn angle and max segment length, the number of resets per unit distance for COPPER strategy (0.026 ± 0.001) was significantly lower compared to that for FORCE ($0.075 \pm 0.001, t(1269) = 25.42, p < .001$), S2C ($0.096 \pm 0.001, t(1269) = 36.38, p < .001$), and the No Redirection control condition (0.138 ± 0.001, $t(1269) = 58.17, p < .001$). Therefore, the rule for this case is simply to use COPPER when the branching likelihood is 0.

8 Discussion

Prediction Graph Properties. The performance of the baseline condition (No Redirection) helps with understanding whether performance can be attributed to the redirection strategy or a property of the prediction graph. For the max segment length property, the baseline indicates that longer segments result in improved performance. However, in the presence of redirection, performance declines as the segment length becomes larger, and the magnitude of this effect becomes greater as we advance from S2C, to FORCE, to COPPER. This can be explained as follows: strategies with greater predictive accuracy

	Rule	Strategy
1	$\theta <= 90 \wedge P(b) > .2$	S2C
2	$P(b) > .6 \wedge \theta <= 135$	S2C
3	$l <= 5 \wedge \theta > 90 \wedge P(b) <= .5$	FORCE
4	$\theta <= 45 \wedge l > 2.5$	S2C
5	$P(b) <= .2 \wedge \theta > 90$	FORCE
6	$l > 7.5 \wedge \theta <= 135$	S2C
7	$l > 5 \wedge P(b) > .6$	S2C
8	$l > 7.5 \wedge P(b) > .3$	S2C
9	$l > 2.5 \wedge \theta > 135$	FORCE
10	$P(b) > .6$	S2C
11	$l <= 5 \wedge P(b) = 0$	FORCE
12	$l > 5 \wedge \theta <= 90$	S2C
13	$P(b) <= .4$	FORCE
14	$l > 2.5$	S2C
15	–	FORCE

Table 2. Derived adaptation rules for determining the most effective strategy in the branching prediction graph scenario. With these 15 rules, we can programmatically determine how to switch between strategies at runtime with minimal computational overhead. $P(b)$ is the branching probability, l is the max segment length, and θ is the max turn angle.

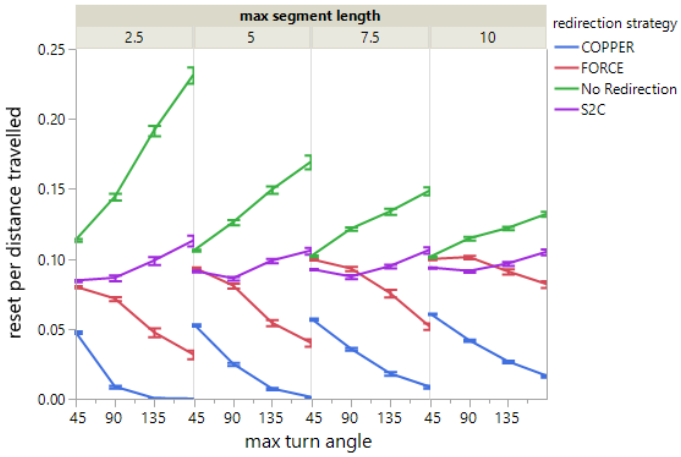

Fig. 8. Performance comparison of contending redirection strategies across prediction graph properties for non-branching prediction graphs. The Y axis shows normalized reset values, therefore lower values indicate better performance. Error bars show standard error. We can see how each redirection strategy interacts differently with the factors, but in all cases COPPER consistently dominates other strategies.

can make use of smaller path segments by using redirection to contain the path. The reduced performance of our baseline with smaller segments can also be explained by the lack of redirection, which causes smaller segments to get stuck near a boundary and experience frequent resets. However, longer path segments increase the odds of escaping a near-boundary region.

For the max turn angle property, we see that the baseline and S2C mostly experience a reduction in performance. However, for the planning strategies, performance improves with increased turn angles (especially for COPPER). This can be explained by the fact that increasing the turn angles allows for planning strategies to have more flexibility with the range of directions they can steer the user to keep the path contained within the physical space. The reduced performance of the baseline condition can similarly be explained by the fact that larger turn angles can cause the user to keep turning back towards the physical space boundary after a reset, effectively undoing the reset task. And similarly, when the turn angles are smaller, the user is more likely to escape the boundary area after a reset.

As expected, branching has no significant effect on non-planning strategies. However, the FORCE results were consistent with our expectations for dynamic planning strategies, where virtual path uncertainty (i.e. user unpredictability) reduces performance.

Limitations and Future Work. The empirical results reported in this paper were computed using a simulation framework. Simulation is commonly employed by researchers to evaluate redirected walking algorithms in studies that are impractical or impossible to conduct with real users (e.g. [5, 8, 11, 21, 23, 32, 54, 55, 60, 65]), and recent work has been conducted to validate this approach [10]. Although simulation is useful, ultimately these strategies need to be deployed and evaluated with human participants to understand their effects on the user experience. Furthermore, in a live VR system, multiple redirection techniques such as rotation and translation gains are often combined; however, the impact of adaptively switching between different redirection strategies on the user experience has not yet been studied.

Adaptive redirection was conceived to be extensible, and a multitude of strategies could be implemented within this framework. However, one limitation is that the adaptation rules for switching between them need to be derived for a specific physical space and a virtual environment that can be represented using a prediction graph. Because the performance of adaptive redirection is by definition context dependent, evaluations should also be conducted using a wider variety of physical configurations and virtual scenes. The methodology presented in this paper can serve as a proof-of-concept to guide future experiments that integrate new strategies into the adaptive framework.

In this paper, COPPER was compared with representative strategies from the general and dynamic planning categories in a rectangular physical space free of obstacles. However, over the past several years, novel redirected walking algorithms have been introduced to handle more complex physical configurations by applying concepts such as alignment [59], artificial potential fields [11, 32, 55], and machine learning [15, 29, 50]. In future work, COPPER can also be extended to support arbitrary physical spaces and compared with these modern approaches to redirected walking. For a linear VR experience with a non-branching prediction graph, we would expect COPPER to perform similarly or better than other strategies because it can analytically determine a near-optimal solution. However, it would still be valuable to empirically demonstrate the performance differences between these methods under varying conditions.

9 Conclusion

In this paper, we formally defined a redirected walking system and established the theoretical foundations necessary for engineering complex redirection strategies. The framework presented in this paper contributes to a better understanding of how previous work in the field fits together within a unified scope, and also serves as a blueprint for future advancements in the redirected walking landscape. To advance the state-of-the-art for static planning strategies, we introduced the COPPER algorithm, which can compute a near-optimal solution for linear, choreographed redirected walking experiences. We then presented the adaptive redirection meta-strategy, which dynamically chooses the most appropriate redirection strategy based on the current situation. Finally, we demonstrated how performance data could be used to develop adaptation rules, which involves partitioning the spectrum of contexts into regions according to the strategy that performs best. In summary, the adaptive redirection meta-strategy provides a foundation for making redirected walking work in practice and can be extended to improve performance in the future as new redirection strategies are introduced and integrated into the framework.

Acknowledgments

The authors would like to thank Jerald Thomas for his assistance with the paper. This work was sponsored by the U.S. Army Research Laboratory under contract number W911NF-14-D-0005. Statements and opinions expressed do not necessarily reflect the position or the policy of the Government, and no official endorsement should be inferred.

REFERENCES

[1] M. Al Zayer, P. MacNeilage, and E. Folmer. Virtual locomotion: a survey. *IEEE transactions on visualization and computer graphics*, 26(6):2315–2334, 2018.

[2] F. Argelaguet. Influence of dynamic field of view restrictions on rotation gain perception in virtual environments. In *Virtual Reality and Augmented Reality: 17th EuroVR International Conference, EuroVR 2020, Valencia, Spain, November 25-27, 2020, Proceedings*, vol. 12499, p. 20. Springer Nature, 2020.

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

[4] M. Azmandian, M. Bolas, and E. Suma. Countering user deviation during redirected walking. In *Proceedings of the ACM Symposium on Applied Perception*, pp. 129–129. ACM, 2014.

[5] M. Azmandian, T. Grechkin, M. Bolas, and E. Suma. Physical space requirements for redirected walking: how size and shape affect performance. In *Proceedings of the 25th International Conference on Artificial Reality and Telexistence and 20th Eurographics Symposium on Virtual Environments*, pp. 93–100. Eurographics Association, 2015.

[6] M. Azmandian, T. Grechkin, M. Bolas, and E. Suma. Automated path prediction for redirected walking using navigation meshes. In *3D User Interfaces (3DUI), 2016 IEEE Symposium on*, pp. 63–66. IEEE, 2016.

[7] 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 *Everyday VR Workshop, IEEE VR*, 2016.

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

[9] M. Azmandian, R. Yahata, M. Bolas, and E. Suma. An enhanced steering algorithm for redirected walking in virtual environments. In *Virtual Reality (VR), 2014 iEEE*, pp. 65–66. IEEE, 2014.

[10] M. Azmandian, R. Yahata, T. Grechkin, J. Thomas, and E. Suma Rosenberg. Validating simulation-based evaluation of redirected walking systems. *IEEE Transactions on Visualization and Computer Graphics*, to appear.

[11] 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.

[12] R. Boulic, N. M. Thalmann, and D. Thalmann. A global human walking model with real-time kinematic personification. *The visual computer*, 6(6):344–358, 1990.

[13] G. Bruder and E. Langbehn. Subliminal rotations during eye blinks for redirected walking. *Journal of Vision*, 17(10):1266–1266, 2017.

[14] B. J. Congdon and A. Steed. Sensitivity to rate of change in gains applied by redirected walking. In *25th ACM Symposium on Virtual Reality Software and Technology*, pp. 1–9, 2019.

[15] 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.

[16] P. Gao, K. Matsumoto, T. Narumi, and M. 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)*, pp. 639–648. IEEE, 2020.

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

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

[19] D. Hayashi, K. Fujita, K. Takashima, R. W. Lindeman, and Y. Kitamura. Redirected jumping: Imperceptibly manipulating jump motions in virtual reality. In *2019 IEEE Conference on Virtual Reality and 3D User Interfaces (VR)*, pp. 386–394. IEEE, 2019.

[20] J. Hildebrandt, P. Schmitz, A. C. Valdez, L. Kobbelt, and M. Ziefle. Get well soon! human factors' influence on cybersickness after redirected walking exposure in virtual reality. In *International Conference on Virtual, Augmented and Mixed Reality*, pp. 82–101. Springer, 2018.

[21] 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.

[22] E. Hodgson, E. Bachmann, and T. Thrash. Performance of redirected walking algorithms in a constrained virtual world. *IEEE transactions on visualization and computer graphics*, 20(4):579–587, 2014.

[23] J. E. Holm. *Collision Prediction and Prevention in a Simultaneous Multi-User Immersive Virtual Environment*. PhD thesis, Miami University, 2012.

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

[25] V. Interrante, B. Ries, and L. 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.

[26] E. Langbehn, P. Lubos, G. Bruder, and F. 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.

[27] E. Langbehn, F. Steinicke, P. Koo-Poeggel, L. Marshall, and G. Bruder. Stimulating the brain in vr: Effects of transcranial direct-current stimulation on redirected walking. In *ACM Symposium on Applied Perception 2019*, pp. 1–9, 2019.

[28] E. Langbehn, F. Steinicke, M. Lappe, G. F. Welch, and G. 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.

[29] 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.

[30] Y.-J. Li, D.-R. Jin, M. Wang, J.-L. Chen, F. Steinicke, S.-M. Hu, and Q. Zhao. Detection thresholds with joint horizontal and vertical gains in redirected jumping. In *2021 IEEE Virtual Reality and 3D User Interfaces (VR)*, pp. 95–102. IEEE, 2021.

[31] K. Matsumoto, Y. Ban, T. Narumi, T. Tanikawa, and M. Hirose. Curvature manipulation techniques in redirection using haptic cues. In *2016 IEEE Symposium on 3D User Interfaces (3DUI)*, pp. 105–108. IEEE, 2016.

[32] 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.

[33] R. Nagao, K. Matsumoto, T. Narumi, T. Tanikawa, and M. Hirose. Ascending and descending in virtual reality: Simple and safe system using passive haptics. *IEEE transactions on visualization and computer graphics*, 24(4):1584–1593, 2018.

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

[35] 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.

[36] A. Nguyen and A. Kunz. Discrete scene rotation during blinks and its effect on redirected walking algorithms. In *Proceedings of the 24th ACM Symposium on Virtual Reality Software and Technology*, pp. 1–10, 2018.

[37] 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.

[38] N. C. Nilsson, S. Serafin, F. Steinicke, and R. Nordahl. Natural walking in virtual reality: A review. *Computers in Entertainment (CIE)*, 16(2):1–22, 2018.

[39] N. Nitzsche, U. D. Hanebeck, and G. Schmidt. Motion compression for telepresent walking in large target environments. *Presence: Teleoperators & Virtual Environments*, 13(1):44–60, 2004.

[40] T. C. Peck, H. Fuchs, and M. C. Whitton. Evaluation of reorientation techniques and distractors for walking in large virtual environments. *IEEE transactions on visualization and computer graphics*, 15(3):383–394, 2009.

[41] T. C. Peck, H. Fuchs, and M. 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*, pp. 35–38. IEEE, 2010.

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

[43] S. Razzaque, Z. Kohn, and M. C. Whitton. Redirected walking. In *Proceedings of EUROGRAPHICS*, vol. 9, pp. 105–106. Citeseer, 2001.
[44] N. Rewkowski, A. Rungta, M. Whitton, and M. 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)*, pp. 395–404. IEEE, 2019.
[45] R. A. Ruddle and S. Lessels. The benefits of using a walking interface to navigate virtual environments. *ACM Transactions on Computer-Human Interaction (TOCHI)*, 16(1):5, 2009.
[46] R. A. Ruddle, E. Volkova, and H. H. Bülthoff. Walking improves your cognitive map in environments that are large-scale and large in extent. *ACM Transactions on Computer-Human Interaction (TOCHI)*, 18(2):10, 2011.
[47] P. Schmitz, J. Hildebrandt, A. C. Valdez, L. Kobbelt, and M. Ziefle. You spin my head right round: Threshold of limited immersion for rotation gains in redirected walking. *IEEE transactions on visualization and computer graphics*, 24(4):1623–1632, 2018.
[48] S. Serafin, N. C. Nilsson, E. Sikstrom, A. De Goetzen, and R. Nordahl. Estimation of detection thresholds for acoustic based redirected walking techniques. In *2013 IEEE Virtual Reality (VR)*, pp. 161–162. IEEE, 2013.
[49] 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.
[50] 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.
[51] J. Su. Motion compression for telepresence locomotion. *Presence: Teleoperators and Virtual Environments*, 16(4):385–398, 2007.
[52] E. Suma, S. Finkelstein, M. Reid, S. Babu, A. Ulinski, and L. 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, 2010.
[53] Q. Sun, A. Patney, L.-Y. Wei, O. Shapira, J. Lu, P. Asente, S. Zhu, M. McGuire, D. Luebke, and A. Kaufman. Towards virtual reality infinite walking: dynamic saccadic redirection. *ACM Transactions on Graphics (TOG)*, 37(4):1–13, 2018.
[54] 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.
[55] 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.
[56] Unity3D Game Engine. `https://unity3d.com/`.
[57] 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 *Proceedings of the 26th annual conference on Computer graphics and interactive techniques*, pp. 359–364. ACM Press/Addison-Wesley Publishing Co., 1999.
[58] 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 *Proceedings of the 4th symposium on Applied perception in graphics and visualization*, pp. 41–48. ACM, 2007.
[59] 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.
[60] 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*, 27:4267–4277, 2021.
[61] 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.
[62] D. Wolf, K. Rogers, C. Kunder, and E. Rukzio. Jumpvr: Jump-based locomotion augmentation for virtual reality. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*, pp. 1–12, 2020.
[63] C. You, E. Suma Rosenberg, and J. Thomas. Strafing gain: A novel redirected walking technique. In *ACM Symposium on Spatial User Interaction*, p. 26. ACM, 2019.
[64] M. Zank and A. Kunz. Optimized graph extraction and locomotion prediction for redirected walking. In *3D User Interfaces (3DUI), 2017 IEEE Symposium on*, pp. 120–129. IEEE, 2017.
[65] 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.