Monte-Carlo Redirected Walking: Gain Selection Through Simulated Walks

Ben J. Congdon and Anthony Steed

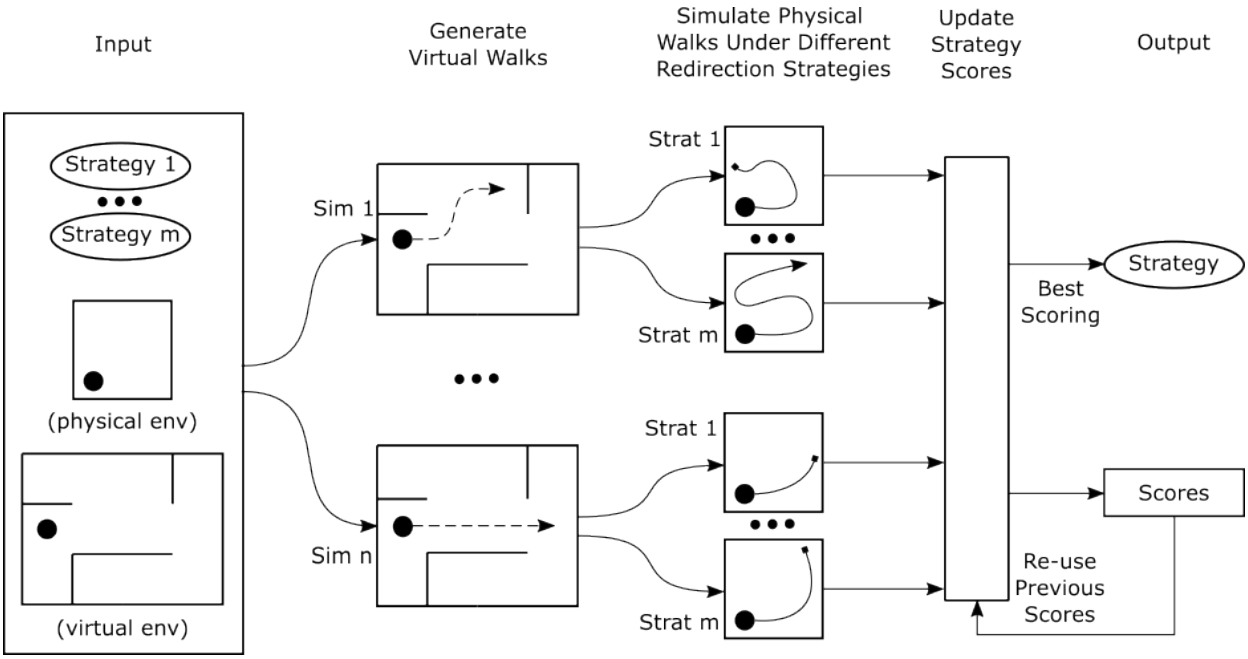


Fig. 1: An example calculation step for the Monte-Carlo Redirected Walking (MCRDW) gain selection algorithm. For a given virtual environment, a large number of virtual walks are generated. For each virtual walk, a physical walk is simulated under the effect of one of a set of redirection strategies. Strategies are combinations of different levels and directions of virtual gain, varying over the course of the virtual walk. Simulations are conducted by calculating the physical path a user must walk to follow the generated virtual path when under the effects of these different levels of gain. Calculated physical paths are then scored and the score for the corresponding strategy updated. Simulations can be conducted in parallel to a realtime virtual reality application. When required, the current best scoring strategy can be used to provide gain levels and directions for redirected walking.

Abstract—We present Monte-Carlo Redirected Walking (MCRDW), a gain selection algorithm for redirected walking. MCRDW applies the Monte-Carlo method to redirected walking by simulating a large number of simple virtual walks, then inversely applying redirection to the virtual paths. Different gain levels and directions are applied, producing differing physical paths. Each physical path is scored and the results used to select the best gain level and direction. We provide a simple example implementation and a simulation-based study for validation. In our study, when compared with the next best technique, MCRDW reduced incidence of boundary collisions by over 50% while reducing total rotation and position gain.

Index Terms—Virtual reality, human computer interaction, redirected walking.

1 Introduction

Virtual environments are three-dimensional spatial representations that respond to user movement and interaction in real time. Virtual reality (VR) systems allow a user to experience a virtual environment in a way that feels closer to reality, where display and interaction are closer to their physical counterparts. This involves combining display and tracking technologies to translate user movement in the physical space (or track space) into movement in the virtual environment.

Ben J. Congdon (ben.congdon.11@ucl.ac.uk) and Anthony Steed (a.steed@ucl.ac.uk) are with University College London.

Typically this involves wearing a head-mounted display (HMD). The tracking system provides information on HMD position and therefore user head direction. This works well for looking around a virtual environment. However, locomotion in VR is inherently difficult. Ideally, we would like virtual movements to bear a close resemblance to a user's tracked physical movements. This reduces the potential for nausea and helps create the illusion that the user is truly present within a virtual environment. However, in the case of many VR systems, only a limited space can be tracked. For other systems, users will instead be limited by the physical space around them and cannot wander freely without meeting obstacles.

This creates the motivation for artificial methods of locomotion; approaches which match a user's movement on some intuitive level without requiring free movement in space. Navigation schemes have been proposed and are in use today which are primarily driven by head direction, or by an external device such as a joystick. These

schemes have proven to be less than ideal, increasing simulator sickness and reducing presence, subjective measures of nausea and immersion respectively. "Real walking", i.e., physical locomotion within the track space, more closely matches the virtual counterpart, and has been shown to be greatly preferable [24].

Redirected walking (RDW) is a method for real walking in virtual reality which can allow users to walk further than their physical environment allows [18]. This technique imperceptibly transforms the virtual environment around the user, disrupting the mapping between track space and virtual environment. Selecting the correct transformations leads to a favourable mapping, permitting exploration of virtual environments larger than the track space.

Early RDW implementations were effective but inflexible. Users were required to follow waypoints in the virtual environment, carefully spaced to allow sufficient time for redirection [18]. Recent research has focused on generalized RDW, extending the principles to allow free exploration of arbitrary environments. However, without a guaranteed user path, selecting good transformations becomes a difficult problem. Simple heuristic techniques consider only the user's current position and typically attempt to steer towards a target (e.g., Steer-to-Center, Steer-to-Orbit, Steer-to-Multiple [5, 9, 17]). These approaches require very large track spaces and make no use of virtual environment layout. In simple use cases (such as walking in a straight line) the limiting factor on RDW performance is human perceptual thresholds so heuristic techniques are effective. However, obstacle-rich virtual environments encourage frequent changes of direction. Improved RDW performance should be possible in these environments.

Optimization-based techniques can take virtual environment layout into account by evaluating possible future paths and aiming to maximize some metric over a window (e.g., FORCE [28], MPCRed [13]). Meanwhile, deep-learning based techniques use a pre-trained network to estimate good future redirections (e.g., Steer-to-Optimal-Target [11], Steering via Reinforcement Learning [21]). While developments in optimisation and deep-learning based techniques are encouraging, the authors of these techniques currently report only a small improvement (at most around 15%) over Steer-to-Center (S2C) [11, 13, 21, 28].

This paper contains the design and evaluation of the **Monte-Carlo Redirected Walking** (MCRDW) algorithm, a novel approach to gain selection. The aim of this technique is to improve on existing redirection selection algorithms while remaining applicable to any virtual environment, simple to implement and computationally lightweight. The algorithm uses simulated walks to anticipate future user trajectories. In our simulation-based experiment, MCRDW was significantly more effective at keeping users within physical boundaries when compared with existing RDW algorithms, particularly in VEs which are obstacle-rich.

2 Related Work

In its original form as proposed by Razzaque et al., RDW guides the user through a series of waypoints in both the virtual environment and the tracked space [18]. Only rotation gain is used. The magnitude is adjusted dynamically based on user movement. The final rotation applied is the maximum of three factors: scaled linear velocity, scaled rotational velocity and a baseline rotation per second. The direction of this rotation is calculated to send the user towards the next waypoint in the track space based on the user direction in the virtual environment, assumed to be the direction towards the next virtual environment waypoint.

In the general case, future positions and orientations are uncertain as the user is free to move as they wish. This is known as generalised [9], generic [20] or reactive [15] RDW. Early gain selection algorithms make heavy use of heuristics. *Steer-to-Center* (S2C) redirects the user to the center of the track space, and *Steer-to-Orbit* (S2O) redirects the user to a circle around the edge of the track space [5, 17]. S2C has proven to be preferable when users change direction frequently, while S2O is more effective when users walk long distances in a straight line [7].

Extended heuristic approaches exist in the literature, such as using translation gain together with S2C to extend walkable distance when heading directly away from the center point [1]. APF-RDW uses artificial potential fields (APFs) to generate a steering target rather than using the center in order to create a heuristic approach suitable for more complicated physical environments [12]. Push/pull reactive (P2R) redirection combines both APFs and translation gain [23].

Heuristic techniques are simple to compute, but frequently choose sub-optimal redirections. S2O assumes that the user will never change direction, while S2C assumes that the user may change direction at any time with equal likelihood for each direction. This ignores valuable sources of data. For example, walls and obstacles in the virtual environment provide bounds on walkable space. In constrained environments S2C and S2O waste a great deal of the track space on these unwalkable areas. Despite greater available information on user path, they perform no better in constrained environments than open ones [8].

Gain selection can be viewed as an optimization problem over the space of all possible combinations of redirections and user paths, as with FORCE [28] and MPCRed [13]. Complexity is an issue with this approach, as there are many combinations of virtual path and possible redirection strategies. Both algorithms use a fixed maximum depth to prevent exponential complexity growth.

More recently, machine learning techniques have been applied to the gain selection problem. *Steer-to-optimal-target* (S2OT) uses reinforcement learning to train a model using the Deep Q-Learning approach [11]. The system divides the tracked or physical space evenly into squares and places a target at each intersection point. For each intersection, the predictor is run to determine the outcome were the user to be redirected with the point as the target. The goal function balances the likelihood of a physical collision against the amount of rotational redirection required to redirect towards the target. Strauss et al. propose *Steering via Reinforcement Learning* (SRL) [21]. Unlike S2OT, SRL treats the physical space as continuous. Both techniques use a Deep Q-Learning approach, though SRL uses a variant known as Proximal Policy Optimisation (PPO).

Comparing redirection techniques directly is difficult because a strategy that is effective in one environment may not be the most effective for another. With this caveat, the above machine learning approaches can demonstrate an improvement over heuristic techniques: both S2OT and SRL are shown to reduce collisions when compared with S2C. The effect size is, however, modest for both techniques. SRL was able to increase mean distance covered by around 4% on simulated paths when compared with S2C. No significant difference was observed on real paths. S2OT demonstrated improvements in both simulation and user study, reducing collisions by approximately 15% [11]. This performance improvement comes at the cost of higher rotation gain. The goal function weighting in S2OT favours reducing physical collisions over keeping redirection levels low. Overall, S2OT applies significantly more rotational gain than other approaches, with 30% or more over S2C.

Gain selection for generalised redirected walking remains an open problem. To build on existing work, the ideal gain selection algorithm would be widely applicable to many virtual environments, and when compared with older heuristic techniques would reduce the level of gain applied to users and significantly reduce the incidence of boundary collisions.

3 Method

In this section we describe gain selection through simulated walks at a theoretical level. See Section 4 for a sample implementation of these ideas and Section 5 for an evaluation of that implementation using simulations.

To start we consider the redirection function f. As the user walks in the physical space, we apply f to their movements (translation and orientation) to calculate the resulting virtual world position. Without redirection, f is the identity function, and the user's movements therefore mapped 1 to 1. With redirection (or 'gain'), the virtual rotation and translation may be smaller or greater. Note that f only applies a redirection strategy, rather than generating one:

$$v_t = f(x_t, v_{t-1}, w_t, w_{t-1}) \tag{1}$$

Where: x_t is a 2-tuple containing rotation and translation gain at time t; v_t is the user's virtual position and orientation at time t; w_t is the user's track (physical) position and orientation at time t.

The aim of redirection selection algorithms is to generate the redirection strategy x. Any value of x is possible, but the ideal value is likely to change frequently as the user moves around the environment, or as the environment changes. At an abstract level the optimal redirection selection algorithm has two goals when selecting x: 1) To maximize **boundary avoidance** by selecting redirections that avoid physical boundaries, and 2) To maximize **subtlety** by selecting redirections that minimise disruption to the user.

The simulated-walk approach to redirected walking is to virtually conduct many possible walks from the user's position under different redirection strategies. The walks are then scored in favour of those that best satisfy the *boundary avoidance* and *subtlety* goals. Various metrics can be applied during scoring. For example, walks could be weighted in favour of those that apply the lowest gain levels, or maximise the time to physical collision. The strategy with the highest score at a given moment is provided to the redirection function in the form of x. The proposed approach is iterative and can combine the results of previous runs. As each simulation is computationally simple and standalone, this algorithm suits real-time applications as it can be run as long as allowed by the frame timing. An example diagram can be found in Figure 1. The remainder of this section will consider theoretical approaches to simulation (see Sections 3.1 and 3.2) and scoring (Sections 3.3 and 3.4).

3.1 Generating Potential Virtual Paths

The first step of any simulated walk is to calculate $\bar{v}$, a path through the virtual environment. The path must start at v_0, the user's current position and orientation in the virtual space, and can end at any point.

The simulated-walk approach makes the assumption that a sample of possible walks under different redirection strategies is representative of the set of all possible walks under all possible redirection strategies. To improve the quality of our sampling and bring us closer to a representative sample, virtual paths can be selected based on a probability distribution which favours more likely paths. The task of generating good virtual paths is therefore analogous to the path prediction task found elsewhere in redirected walking.

In theory, any long-term path predictor can be used, and a variety exist in the literature [14, 16, 22, 26, 27]. Many path predictors only provide a single estimate of the user's path. More useful for our purposes are those that provide a range of possible paths with accompanying probabilities; we will call these stochastic path predictors. Accurate probabilities are to be preferred as less computation time will be wasted on unlikely paths. Additionally, when scoring walks, it becomes possible to weight redirections in favour of those that perform best on more likely paths.

Historical user virtual path is a previously used indicator of future virtual path [16], optionally taking into account models of human locomotion [27]. The layout of the virtual environment is also an indicator which has been used in the form of a graph [26] or a navigation mesh [2]. Previous work has also shown paths are predictable when movement is goal-orientated [6]. Finally, as paths must be generated at runtime, a very quick path predictor is also preferred. This provides as much time as possible for simulations.

As the MCRDW approach is compatible with any stochastic path predictor we provide no recommendations in this section, other than to note that the ideal path predictor is computationally lightweight and stochastic. The concrete implementation provided in Section 4 uses an approach loosely based on that used by Peck [16], extended to provide probabilities.

3.2 Simulating Physical Walks with RDW Strategies

Once the virtual path $\bar{v}$ has been generated, the next task is to work back to the physical path that would have led the user round this virtual path. The physical path is dependent on the redirection strategy in use. Related to f (see Equation 1), we now need a new function which calculates where a user would be in the track space if they were to walk a virtual path with a certain set of redirections applied. As the user subconsciously counters the redirections in the virtual environment, we should then be able to calculate where those movements will place them in the physical space. We will call this function g:

$$w_t = g(x_t, w_{t-1}, v_t, v_{t-1}) \tag{2}$$

For our simulations complete virtual paths are generated. All values of v (and consequently w) are therefore known. Different approaches to f are possible; for the sake of calculating physical paths, easily invertible approaches are preferred. Example equations for f and g are provided in Section 4.

3.3 Scoring with Boundary-Avoidance Metrics

The fundamental outcome that *boundary avoidance* metrics are trying to minimise is boundary collisions. A good function will provide an accurate estimate of the user's future likelihood of a boundary collision. It is also of benefit if they are simply expressed and computationally efficient.

Hodgson et al. suggest a number of metrics for comparing the performance of redirected walking techniques [7] that we may be able to adapt into boundary avoidance metrics. Over the course of the entire path, we could measure the mean and max distance from the track space center (MDC and MaxDC), or the mean distance from the nearest track boundary (MDNB).

MDC and MDNB are based on the concept that greater distance from the track space center is an indication of poor redirection performance. This is only sometimes the case. The heuristic technique *Steer-To-Orbit*, which leads the user around the optimal path for straight-line walking [5, 17]. The path used is circular with a high radius. For the same path, Steer-To-Center would lead the user in a figure-of-eight pattern with a much higher maximum radius, increasing the likelihood of boundary collision. However, both techniques would score similarly under MDC.

With MaxDC, Steer-To-Orbit will score better than Steer-To-Center in straight-line walking. However, MaxDC has the same problem as MDC for our purposes: distance from track space center is not a problem if it does not lead to boundary collisons. A very large maximum is equivalent to a small maximum if neither of these results is larger than the track space radius.

The example implementation described in Section 5 uses the simplest possible approach, time to first boundary collision (TTBC). A possible hazard with this technique is its inability to capture near misses. Approaching the boundary and only just avoiding collisions will be scored highly by TTBC. However, as simulations will never be completely precise, in practice the strategy may lead to collision. However, TTBC is very simple to implement, and as the implementation limits simulations to a fixed length in time, results from TTBC are easily normalised. This simplifies the process of combining *boundary avoidance* and *subtlety* metrics into a single measure. As TTBC is also very simple to calculate, more time is available for simulations.

3.4 Scoring with Subtlety Metrics

Re-orientation after a boundary collision is overt and constitutes a significant disruption to user experience. It may be that high gain levels are preferable to boundary collisions. The simulated walk approach can allow for a balance to be found between *boundary avoidance* and *subtlety*, as larger redirections are less subtle but more effective. Redirection strategies can therefore include large redirections with a score penalty, permitting somewhat perceptible redirection if it would prevent a user from encountering a boundary.

Possible metrics could be Hodgson's mean and max rate of unsigned redirection (MRUR and MaxRUR) [7]. Short periods of high gain can be obscured by MRUR but can overly skew MaxRUR. A metric could combine both, to counteract these issues.

The example implementation described in Section 5 stays below the prior thresholds found in [3] and [19]. This allows for straightforward comparison with earlier techniques. So long as gains are imperceptible, it is perhaps not helpful to reduce them further. However, we still apply a very small score penalty based on the rate of gain (MaxRUR) for the initial strategy. This encourages MCRDW to apply the minimum level

of gain required, partly for consistency; settling on a strategy when differences in performance are small.

4 Implementation

This section includes our concrete implementation of the ideas in Section 3: generate a virtual path; for all n strategies, apply the strategy in inverse to the virtual path, yielding n physical paths; score all n physical paths, and update the score table for their corresponding strategy. Repeat until computation time elapses. Finally, return the current best scoring strategy. Our implementation used 9 strategies: A 3×3 combination of different directions of rotation (left, none, right) and translation (reduce, none, magnify) gain.

4.1 Implementing Virtual Path Generation

For each simulation we generate a walk using a path predictor following a similar process to that described in [16]. For the purposes of path prediction, the virtual environment is divided into evenly spaced nodes. Edges are generated between nodes which can be directly walked between. For a given path prediction query, the user is considered to be at the closest current node. The predictor weighs the likelihood of visiting any neighbouring nodes, then a node is picked randomly from the (weighted) possibilities, and finally the predictor is updated with the new information. This process repeats until the max path length is exceeded.

The predictor used has two components. Each component outputs a list of probabilities, one for each neighbouring node. The first component, 'history', records visited nodes and decreases the likelihood of visiting those which have been visited very recently. The second component, 'direction', increases the likelihood of visiting nodes which the user is heading towards. Direction is calculated by sampling the user's movements to generate a smoothed direction vector.

To weigh the two components against each other, the consistency of the direction vector and the speed of the user are combined to create a value, 'confidence'. When confidence is low we rely on history. When confidence is high, we rely on direction.

4.2 Implementing Physical Walk Simulation

When we have the virtual path generated, we simulate the physical path that the user would have to walk to follow that virtual path. As described above, 9 strategies were a combination of rotation (left, none, right) and translation (reduce, none, magnify) gain. More sophisticated strategies are possible; these simply applied a fixed level of gain. The base levels of gain applied followed the subtlety thresholds described in [19].

We simulate one physical walk per strategy. First we divide the walk into sections ('legs'), each treated as a straight line. We use the waypoints of the path as the start and end points of legs. For each leg, we select a redirection strategy.

Early in the walkthrough, all legs use the same strategy: this is the strategy that will have its score updated. Strategies for later legs are selected randomly with no weighting. This is because the algorithm is free to combine strategies. The best approach is almost certain to be one strategy now and different strategies later. Randomly selecting strategies for later legs helps to represent these combinations.

To simulate a leg, we calculate the angle the user must turn to face the end point of the leg, and the distance the user must walk between the start and end points. We call these the virtual turn delta and virtual position delta. The user may turn more or move further should they move on a curved path or back and forth, but we use the deltas for our simulations as they can be considered the worst case; it gives the algorithm the least opportunity to apply gain.

Typically in RDW physical movement is fixed; we apply redirection to this physical movement to generate redirected virtual movement. In these walkthroughs, the virtual movement is fixed. We are instead interested in calculating what physical movement would have generated this virtual movement given a particular redirection strategy. With a sufficiently simple redirection formula, we can work back from virtual turn and position deltas to calculate physical turn and position deltas. We use the following physical-to-virtual equations for delta rotations and positions:

$$\Delta v_{rot} = \Delta w_{rot} \cdot \begin{cases} coro & \text{if } \operatorname{sgn} x_{rot} = \operatorname{sgn} \Delta w_{rot} \\ anti & \text{otherwise} \end{cases} \tag{3}$$

$$\left\|\Delta v_{pos}\right\| = \left\|\Delta w_{pos}\right\| \cdot \begin{cases} magn & \text{if } x_{pos} > 0 \\ redu & \text{otherwise} \end{cases} \tag{4}$$

Where: x is a 2-tuple containing (rot and pos) redirection instruction; v is a 2-tuple containing user's virt rotation and position; w is a 2-tuple containing user's phys rotation and position; *coro* is the gain when turning with redirection instruction $\in [1,\infty)$; *anti* is the gain when turning against redirection instruction $\in (0,1]$; *magn* is the gain when magnifying physical movement $\in [1,\infty)$; *redu* is the gain when reducing physical movement $\in (0,1]$.

We use *coro*, *anti*, *magn* and *redu* because earlier work has found users have varying tolerances depending on the direction of redirection [3, 19]. Note that for positions, only the magnitude is modified as position deltas are applied relative to the current facing vector in the appropriate space (virtual or physical). Additionally, in practice, redirection equations will also apply smoothing. We omit this when performing walkthroughs to significantly simplify the process of simulating legs. Finally, note the method is never required to apply these forward transformations; we only use them to generate the following corresponding virtual-to-physical equations:

$$\Delta w_{rot} = \Delta v_{rot} \cdot \begin{cases} \frac{1}{1+r\cdot(coro-1)} & \text{if } \operatorname{sgn} x_{rot} = \operatorname{sgn} \Delta v_{rot} \\ \frac{1}{1+r\cdot(anti-1)} & \text{otherwise} \end{cases} \tag{5}$$

$$\left\|\Delta w_{pos}\right\| = \left\|\Delta v_{pos}\right\| \cdot \begin{cases} \frac{1}{1+r\cdot(magn-1)} & \text{if } x_{pos} > 0 \\ \frac{1}{1+r\cdot(redu-1)} & \text{otherwise} \end{cases} \tag{6}$$

Where r is a random number $\in [0,1]$. The random term is included to represent the possibility that the algorithm may change redirection strategy partway through a leg. This is a real possibility as the MCRDW is run continually, and the best strategy updated continually, so the strategy in use could change at any time. During early development of the technique the addition of this term led to a small but consistent performance improvement.

In this implementation strategies are limited to fixed gain levels. We can therefore assume sufficient subtlety, and do not calculate subtlety metrics (see Section 3.4).

The walkthrough continues until encountering the end of the path or a boundary in the (simulated) physical environment. Finally, the score is recorded using the TTBC metric (see Section 3.3). The score is the total time walked before encountering a boundary, divided by the max path length for normalisation.

4.3 Finalising Scores

Finally, scores are merged with the scores from previous frames with the following process. 1) For each strategy, divide all scores by the number of walks to normalise; 2) apply a weighting to slightly disincentivize those strategies which apply greater levels of redirection; 3) push the new scores onto a queue along with a scalar value for the current time. For each score in the queue, the score is given a weight based on its age, with more recent scores weighted higher. To calculate the final score for each strategy, we multiply each score in the queue by its weight, then sum all the weighted scores, and finally we divide the result by the sum of all weights. The highest scoring method using this approach is the current redirection strategy. The weight values we used were: 0.995 for (Left, Magnify), (Left, Reduce), (Right, Magnify) and (Right, Reduce). 0.9975 for (Left, None), (Right, None), (None, Magnify) and (None, Reduce), and 1.0 for (None, None).

5 Evaluation

This section contains information on the simulation-based experiment conducted to validate the technique. This includes a concrete implementation of MCRDW and details on simulation methodology and experimental setup. We conclude with the results of the experiment and a discussion of these results.

5.1 Simulator

The simulator used in this section has uses a *runner*, a configurable procedure for conducting simulations. The input to the runner has two components: a layout, which is a virtual environment and path around that environment; and a method, which is one of the redirection techniques under evaluation. The *runner* applies the *method* to the movements of a simulated user as they follow the path described in the *layout*. The output from the runner is a set of performance statistics gathered during the run.

We use a large number of procedurally generated layouts to represent the population of possible virtual environments. For the sake of comparison we run through each layout once with each method. Optionally, a *scheduler* can be used to queue up runs with the correct layout and method, and a *visualizer* can provide a graphical output to help monitor the simulations while in-progress.

5.2 Layouts

Layouts contain two elements: the virtual environment itself and a path through the environment. The virtual environment is represented by a set of walls which cannot be traversed, and the path by a series of a waypoints.

We use a connected graph as an intermediate step to help generate the layout. Note however that the layout itself is not graph based. The inputs to our generation follow: 1) the size ($n \times m$); the number of nodes in each dimension, 2) the node spacing; the distance between each node (evenly spaced), 3) the path length, 4) the edge factor (EF) $\in [0,1]$; overall connectedness of the layout, 5), a max straight length; constraint on length of straight sections, and 6) a max straight path length; constraint on length of straight sections of path.

To generate the environment, we use the following four step process: start with an $n \times m$ grid of evenly spaced nodes; randomly generate a minimally connected graph of these nodes; randomly add edges as required by the edge factor; wherever two nodes are not connected by an edge, generate a wall.

For the path, we follow a three step process: randomly select a start and destination node; generate the shortest possible path between start and destination; if total path length would exceed desired length, cut it short and return, otherwise, pick another destination node and go to step 1.

After generation the path and layout are checked for compliance with the max straight length and max straight path length constraints. Should any constraint fail the check, the path and layout are discarded. These constraints are included to make sure that boundary collisions are theoretically avoidable by good gain selection; see Section 5.5 for more on this topic.

The edge factor has the most significant effect on the overall layout. With an edge factor of 0, the layout is minimally connected. With an edge factor of 1, every node is connected to every neighboring node. We can vary the edge factor to smoothly generate any point in between these two extremes. For sample layouts at different levels of edge factor, see Figure 2.

5.3 Runner

The runner is responsible for conducting the simulation using the environment and the path defined by a layout, and the redirection technique defined by a method.

The runner loads the environment described in the layout and advances the simulation at a fixed timestep, first updating the simulated user's position and orientation along the path found in the layout. The runner then applies the method. The runner repeats this process until the simulated user reaches the end of the layout's path. Should a user encounter a boundary in their simulated track space, a simulated 'reset' [25] occurs: the user is returned to the center of their track space but their position in the virtual environment remains the same.

A very simple model of locomotion is used when following the path; the simulated user turns to face their current waypoint and moves towards it. When arriving at the waypoint, the process is repeated with the next waypoint.

5.4 Redirection Methods

At each time step, methods are provided a delta time and the user's virtual and track space position and orientation. Methods are then free to manipulate the virtual position and orientation. To do this, all methods use a common 'redirector' to apply gain. The redirector applies simple smoothing as is typical in RDW applications [4, 7, 17]. Methods are also notified of discontinuities (e.g., resets). This gives the method an opportunity to reset smoothing and prediction variables.

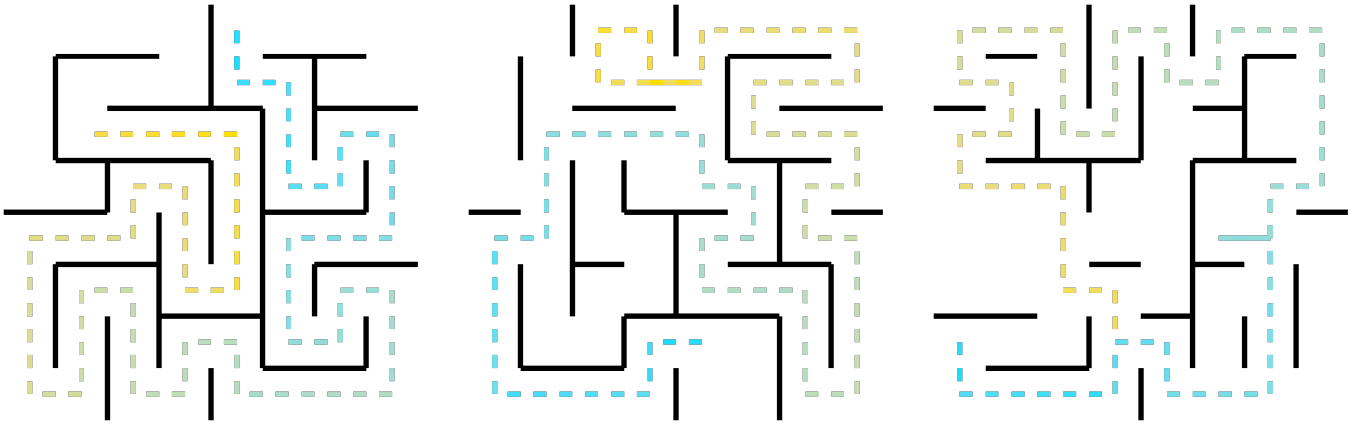
In our simulations, the following methods were evaluated:

- **S2C**: *Steer-to-Center,* which guides the user towards the center of their physical space [5, 17]. Rotation gain only.
- **S2O**: *Steer-to-Orbit,* which guides the user on a circular path around the edge of their physical space [5, 17]. Rotation gain only.
- **S2T**: *Steer-to-temporary,* as S2C but with temporary targets when facing directly away from the center to ensure consistent gain direction [7]. Rotation gain only.
- **S2T-S**: *S2T with static magnification,* as S2T but applying a constant translation gain, effectively increasing the size of the user's physical space. Rotation and translation gain.
- **S2T-D**: *S2T with dynamic magnification,* as S2T but applying a dynamic translation gain; magnifying user movement when moving away from the center, and reducing it when moving towards the center [1]. Rotation and translation gain.
- **NoRDW**: *No redirected walking,* control condition, map user movements 1:1. No rotation or translation gain.
- **MC**: *Monte-Carlo redirected walking* (**MCRDW**, abbreviated for space), as described in Section 4. Rotation and translation gain.
- **MC-Fast**: *Monte-Carlo redirected walking with half compute time,* as MCRDW but reduce available compute time by half, intended to help approximate computation requirements. Rotation and translation gain.
- **MC-Lo**: *Monte-Carlo redirected walking, over-threshold, low,* as MCRDW but when a boundary condition is likely the technique is permitted to exceed perceptual thresholds by 10%. Rotation and translation gain.
- **MC-Med**: *Monte-Carlo redirected walking, over-threshold, medium,* as MC-Lo but instead exceed thresholds by 20%. Rotation and translation gain.
- **MC-Hi**: *Monte-Carlo redirected walking, over-threshold, high,* as MC-Lo but instead exceed thresholds by 30%. Rotation and translation gain.

We aim to compare MCRDW with current top-performing gain selection techniques for small tracking spaces. Direct comparisons are difficult as performance depends on virtual environment and path. We considered S2T-D a reasonable stand-in as it was fast to simulate but still appears to improve on S2C by 20% [1], on a similar level to APFs [12], and the reinforcement learning method, S2OT [11].

5.5 Experimental Setup

Using the procedure described in Section 5.2, we generate 1000 layouts at 3 different levels of edge factors: .00, .15 and .30. This gives us 3000 layouts total. The generator had the following configuration: 8x8 nodes, node spacing 1.3m, total path length 60m, max straight length and max straight path length 4m. As the purpose of these simulations is to distinguish between methods, we use a 5m by 5m physical space to increase



(a) Edge-Factor 0 (b) Edge-Factor 0.15 (c) Edge-Factor 0.30

Fig. 2: Virtual environment layouts generated for evaluation purposes. The process followed for generating these environments is described in Section 5.2. Solid lines are walls. The dotted line is the path followed by the virtual agent. Edge factor helps us evaluate how redirection performance varies in more or less challenging environments. The higher the edge factor, the more branching.

the frequency of boundary collisions. With such a small physical space, no redirection technique can make long-distance straight line walking possible, so we constrain these two aspects to make each boundary collision meaningful. For selected layouts from the experiment generated with this configuration, see Figure 2.

We then simulated the walk generated for each layout with each of the 11 methods described in Section 5.4. The recorded outcome measures, summed across each simulation, were:

- Total boundary collisions
- Total position gain (in metres, absolute)
- Total rotation gain (in degrees, absolute)

For our simulations, the user walking speed is 1 meter per second. The user turns at a rate of 90 degrees per second. As redirection gain is applied multiplicatively, these speeds have minimal impact on the outcome of a simulation. The time step used was 60 updates per second.

For methods with a configurable run-time (MC, MC-Fast, MC-Lo, MC-Med, MC-Hi), the reference calculation time was 10 milliseconds across 6 threads on a Ryzen 7 5800H. However, to help gather results more quickly, simulations were run across machines. To standardise results, a small section of the simulations was performed with the reference setup above and the total number of path sections recorded. The result was a mean of ≈ 22500 and standard deviation of ≈ 2000. This was our calibration value; the methods were therefore limited to calculating no more than 22500 path sections, except for MC-Fast which was instead limited to 11250.

5.6 Results

We consider the three independent variables separately. These variables were total collisions, total position gain and total rotation gain. Each was assessed with a 2-way mixed ANOVA. The within-subjects factor was the condition, as each layout had each condition applied. The between-subjects factor was the level of edge factor, as this generated three different sets of layouts.

Unless otherwise stated, we look for significance at the 1% level ($p < 0.01$), rather than the typical 5%. This to reflect the impact high sample count has on p.

5.6.1 Total Boundary Collisions

The data was normally distributed as assessed by visual inspection of Normal Q-Q Plots. The data included a small number ($n \leq 4$) of outliers (± 4 standard deviations) among all methods. As the sample size is large, we include these values regardless. We omit no values from the dataset.

The data violated Levene's and Box's test for homogeneity of variance and covariance respectively. However, these tests are known to be sensitive with large sample sizes, and with groups of equal size mixed ANOVA is considered robust to heterogenity of variances and covariances [10]. Additionally, visual inspection of scatter plots of residuals showed the expected shape, so we conclude the assumptions of the mixed ANOVA are not violated and perform no transformations on the data.

There was a statistically significant interaction between the method and level of edge factor on boundary collisions, $F(13.189, 19763.341) = 62.045$, $p < .0005$, partial $\eta^2 = .4$. Greenhouse-Geisser correction was used ($\varepsilon = .659$) as Mauchly's test of sphericity indicated that the assumption of sphericity was violated for the two-way interaction, $\chi^2 = 13.34$, $p < .0005$.

Analysis of the **simple main effect of method** is used to determine whether method had an effect on the number of boundary collisions at each level of edge factor. EF .00: $F(5.9, 5905.4) = 5934.8$, $p < .0001, \eta^2 = .856$; EF .15: $F(6.7, 6711.9) = 5560.7$, $p < .0001, \eta^2 = .848$; EF .30: $F(7.1, 7094.0) = 5166.6$, $p < .0001, \eta^2 = .838$.

Pairwise comparisons follow. As is to be expected given the high sample count, almost every pairwise comparison between methods showed significant difference ($p < 0.0001$). So instead we here list the *exceptions*; those comparisons which were *not* significant. MC × MC-Fast: No significant comparison at any level of edge factor, $p > 0.9999$; S2C × S2O: No significant comparison at edge factor level .00, or .15; S2C × S2T: No significant comparison at edge factor level .00, or .30.

Analysis of the **simple main effect of edge factor** is used to determine whether edge factor had an effect on the number of boundary collisions for each method. S2C, S2O, S2T: No significant effect; S2T-S: $F(2, 2997) = 13.9$, $p < .001, \eta^2 = .009$; STS-D: $F(2, 2997) = 13.3$, $p < .001, \eta^2 = .009$; NoRDW: $F(2, 2997) = 10.8$, $p < .001, \eta^2 = .007$; MC: $F(2, 2997) = 551.6$, $p < .0001, \eta^2 = .269$; MC-Fast: $F(2, 2997) = 597.6$, $p < .0001, \eta^2 = .285$; MC-Lo: $F(2, 2997) = 619.1$, $p < .0001, \eta^2 = .292$; MC-Med: $F(2, 2997) = 638.6$, $p < .0001, \eta^2 = .263$; MC-Hi: $F(2, 2997) = 587.3$, $p < .0001, \eta^2 = .282$. Pairwise comparisons summarized in Table 1.

5.6.2 Total Absolute Position Gained

The methods S2C, S2O, S2T and NoRDW do not apply gain to positions so are excluded from this analysis. The remaining data was visually inspected for normality via Normal Q-Q Plot. S2T-D, MC, MC-Half, MC-Lo, MC-Med and MC-Hi appeared normally distributed. However, S2T-S only fit the model very approximately. With large sample sizes, ANOVA is considered robust to violations of normality so we include

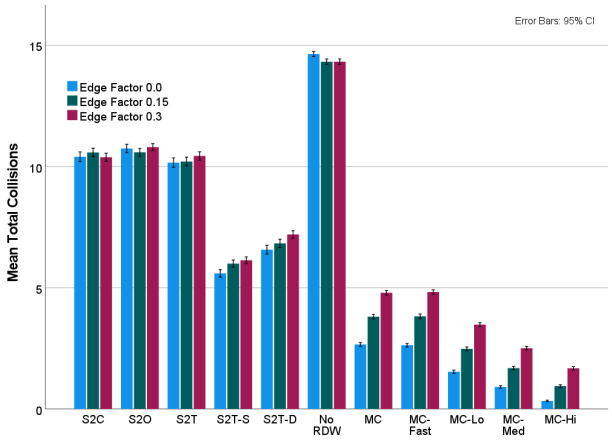


Fig. 3: Mean total **collisions** by method across all simulations, grouped by edge factor.

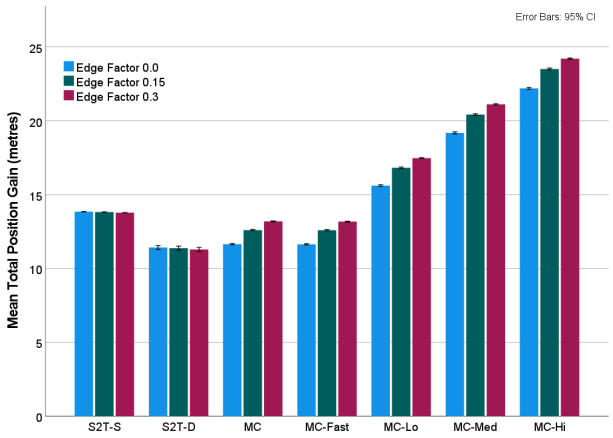


Fig. 4: Mean total absolute **position gained** by method, grouped by edge factor. This is the sum of the absolute differences between virtual and physical movements, averaged across trials.

Method	$.00 \times .15$	$.00 \times .30$	$.15 \times .30$	η^2
S2C				-
S2O				-
S2T				-
S2T-S	*	*		.009
S2T-D		*	*	.009
NoRDW	*	*		.007
MC	*	*	*	.269
MC-Fast	*	*	*	.285
MC-Lo	*	*	*	.292
MC-Med	*	*	*	.263
MC-Hi	*	*	*	.282

Table 1: Pairwise comparisons for **simple main effects of edge factor** on **boundary collisions**, by method. Edge factor had no significant effect on S2C, S2O and S2T.

Method	$.00 \times .15$	$.00 \times .30$	$.15 \times .30$	η^2
S2T-S	*	*	*	.026
S2T-D				-
MC	*	*	*	.494
MC-Fast	*	*	*	.504
MC-Lo	*	*	*	.471
MC-Med	*	*	*	.422
MC-Hi	*	*	*	.418

Table 2: Pairwise comparisons for **simple main effects of edge factor** on **position gain**, by method. S2C, S2O, S2T and NoRDW methods applied no position gain so are not considered. Edge factor had no significant effect on STS-D.

S2T-S [10].

The data included a small number ($n \leq 4$) of outliers (± 4 standard deviations) among all included methods. As the sample size is large, we include these values regardless. We omit no values from the dataset.

The data violated Levene's and Box's test for homogeneity of variance and covariance respectively. However, these tests are known to be sensitive with large sample sizes, and with groups of equal size mixed ANOVA is considered robust to heterogenity of variances and covariances [10]. Additionally, visual inspection of scatter plots of residuals showed the expected shape, so we conclude the assumptions of the mixed ANOVA are not violated and perform no transformations on the data.

There was a statistically significant interaction between the method and the level of edge factor on total position gained, $F(3.7, 5542.7) = 222.7$, $p < .0005$, partial $\eta^2 = .129$. Greenhouse-Geisser correction was used ($\varepsilon = .308$) as Mauchly's test of sphericity indicated that the assumption of sphericity was violated for the two-way interaction, $\chi^2 = 14201.9$, $p < .0005$.

Analysis of the **simple main effect of method** is used to determine whether method had an effect on the total absolute position gained at each level of edge factor. EF .00: $F(2.2, 2243.8) = 15354.1$, $p < .0001, \eta^2 = .939$; EF .15: $F(1.9, 1865.1) = 21492.3$, $p < .0001, \eta^2 = .956$; EF .30: $F(1.4, 1413.9) = 26400.3$, $p < .0001, \eta^2 = .964$.

Pairwise comparisons follow. As is to be expected given the high sample count, almost every pairwise comparison between methods showed significant difference ($p < 0.0001$). So instead we here list the *exceptions*; those comparisons which were *not* significant. MC $\times$ MC-Fast: No significant comparison at any level of edge factor $p > 0.9999$; S2T-D $\times$ MC, and S2T-D $\times$ MC-Fast: No significant comparison at edge factor level .00.

Analysis of the **simple main effect of edge factor** is used to determine whether edge factor had an effect on the total absolute position gained for each method. S2T-S: $F(2, 2997) = 40.5$, $p < .001, \eta^2 = .026$; STS-D: No significant effect; MC: $F(2, 2997) = 1461.4$, $p < .0001, \eta^2 = .494$; MC-Fast: $F(2, 2997) = 1521.8$, $p < .0001, \eta^2 = .504$; MC-Lo: $F(2, 2997) = 1331.9$, $p < .0001, \eta^2 = .471$; MC-Med: $F(2, 2997) = 1094.6$, $p < .0001, \eta^2 = .422$; MC-Hi: $F(2, 2997) = 1074.8$, $p < .0001, \eta^2 = .418$. Pairwise comparisons summarized in Table 2.

5.6.3 Total Absolute Rotation Gained

The NoRDW method does not apply gain to rotations so is excluded from this analysis. The remaining data was visually inspected for normality via Normal Q-Q Plot. The MC techniques MC, MC-Half, MC-Lo, MC-Med and MC-Hi appeared to fit a normal distribution well. The heuristic techniques S2C, S2O, S2T, S2T-S and S2T-D, were consistently slightly left-skewed, but still approximately normal. With large sample sizes, ANOVA is considered robust to violations of normality so we include the heuristic techniques [10].

The data included a very small number ($n \leq 1$) of outliers (± 4 standard deviations) among all included methods. As the sample size is large, we include these values regardless. We omit no values from the dataset.

There was homogeneity of variance for all methods except for MC-Hi ($p < 0.001$). However, the data violated Box's test for homogeneity of covariance. These tests are known to be sensitive with large sample sizes, and with groups of equal size mixed ANOVA is considered robust to heterogenity of variances and covariances [10]. Additionally, visual inspection of scatter plots of residuals showed the expected shape, so

we conclude the assumptions of the mixed ANOVA are not violated and perform no transformations on the data.

There was a statistically significant interaction with a small effect size between the method and the level of edge factor on total rotation gain, $F(12.5, 18706.4) = 46.6$, $p < .001$, partial $\eta^2 = .03$. Greenhouse-Geisser correction was used ($\varepsilon = .694$) as Mauchly's test of sphericity indicated that the assumption of sphericity was violated for the two-way interaction, $\chi^2 = 9838.7$, $p < .0001$.

Analysis of the **simple main effect of method** is used to determine whether method had an effect on the total absolute rotation gained at each level of edge factor. EF .00: $F(5.9, 5878.4) = 129.6$, $p < .001, \eta^2 = .115$; EF .15: $F(6.3, 6253.2) = 347.1$, $p < .0001, \eta^2 = .258$; EF .30: $F(6.5, 6461.7) = 494.4$, $p < .0001, \eta^2 = .331$.

Pairwise comparisons follow. Unlike boundary collisions and position gain, many of these comparisons showed no significant difference. For reasons of space we summarise notable patterns below. MC and MC-Fast: No significant comparison at any level of edge factor $p > 0.9999$; MC-Lo, MC-Med and MC-High: Significant comparisons with all other methods at all levels of edge factor.

Analysis of the **simple main effect of edge factor** is used to determine whether edge factor had an effect on the total absolute rotation gained for each method. S2C: $F(2, 2997) = 7.8$, $p < .001, \eta^2 = .005$; S2O: $F(2, 2997) = 12.7$, $p < .001, \eta^2 = .008$; S2T: No significant effect; S2T-S: $F(2, 2997) = 6.2$, $p \approx .002, \eta^2 = .004$; STS-D: $F(2, 2997) = 5.9$, $p \approx .003, \eta^2 = .004$; MC: $F(2, 2997) = 127.9$, $p < .001, \eta^2 = .079$; MC-Fast: $F(2, 2997) = 126.4$, $p < .001, \eta^2 = .078$; MC-Lo: $F(2, 2997) = 304.0$, $p < .001, \eta^2 = .171$; MC-Med: $F(2, 2997) = 402.6$, $p < .001, \eta^2 = .212$; MC-Hi: $F(2, 2997) = 445.8$, $p < .001, \eta^2 = .229$. Pairwise comparisons summarized in Table 3.

5.7 Discussion

We loosely structure this discussion around the three independent variables. Where helpful we consider the variables together.

5.7.1 Total Boundary Collisions

Overall the family of MCRDW techniques significantly reduced collisions when compared with the heuristic approaches (see Figure 3). At edge factor 0 in our simulations, MCRDW reduced collisions by over 75% when compared with S2C and over 50% when compared with the next best technique, S2T-S. While performance gains were still good at higher levels of edge factor, there was a notable performance reduction. As an environment becomes more open, path prediction becomes more challenging, so we make less accurate guesses about which path a user might take. However, MCRDW methods did continue to outperform heuristic methods even with very open environments at edge factor 0.3.

The over-threshold MCRDW methods MC-Lo, MC-Med and MC-Hi demonstrated a linear performance gain correlated with the level of over-thresholding permitted. The effect was significant. However, as covered later in this Section, these performance improvements came at the cost of noticeably higher total gain overall.

No significant difference was found between MC and MC-Fast ($p > 0.9999$, see Figure 3). This is a good result and likely indicates the method had more than enough time for computation for the environments in our simulations. The available computation time could likely have been reduced further to find the point at which it begins affecting performance. However, the aim of this experiment is to discover whether simulations can be a robust and performant approach to finding good redirection strategies, and this appears to be well demonstrated. Any of the MCRDW techniques can operate as a lightweight background process in the kind of machines capable of driving VR. This constitutes a substantial improvement over a brute force approach which would not be possible in real time.

Based on [7], we expected S2C to outperform S2O as S2O is unable to lead a user around a large circle in a confined space. We also expected S2T to slightly outperform S2C due to more consistent handling of rotations. In practice, these expectations were met and can be observed in the data. However, while the differences were statistically significant,

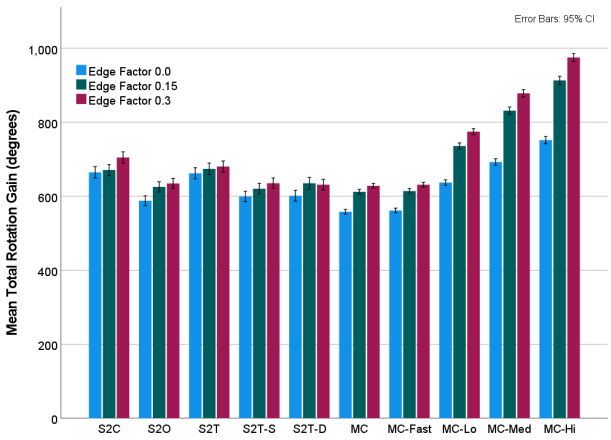


Fig. 5: Mean total absolute **rotation gained** by method, grouped by edge factor. This is the sum of the absolute differences between virtual and physical rotations, averaged across trials.

Method	.00 × .15	.00 × .30	.15 × .30	η^2
S2C		*	*	.005
S2O	*	*		.008
S2T				-
S2T-S			*	.004
S2T-D	*			.004
MC	*	*	*	.079
MC-Fast	*	*	*	.078
MC-Lo	*	*	*	.171
MC-Med	*	*	*	.212
MC-Hi	*	*	*	.229

Table 3: Pairwise comparisons for **simple main effects of edge factor** on **rotation gain**, by method. The NoRDW method applied no rotation gain so is not considered. Edge factor had no significant effect on S2T.

the effect size was very small and so the results largely similar. These methods performed similarly across edge factor levels.

One surprising result was the performance difference observed between S2T-S and S2T-D. Theoretically S2T-D directs the user towards the center of their space more efficiently than S2T-S, but overall S2T-S performed significantly better. One possible explanation would be that during long straight walks S2T-S maximized the overall physical space while S2T-D had a more neutral effect.

Another surprising result is that these methods and NoRDW performed differently across edge factor levels, despite using no form of prediction. This likely indicates that edge factor is not a perfect method for adding branching to environments. We can see why this occurs from the path generation algorithm described in Section 5.2. We only generate paths after already adding the branching edges from edge factor. As a result, edge factor also has an effect on our path generation, and the kind of paths generated. This is a methodological flaw in our approach which could have been avoided by generating paths before adding edge factor. However, it is unlikely to have effected the results materially, as the effect size is notably very small when compared with those techniques which do use path prediction: $\eta^2 < .009$ vs $\eta^2 > .263$.

5.7.2 Total Position Gain

MC reduced total position gain when compared with the best performing heuristic method, S2T-S. However, S2T-S method applies the maximum level of gain for the entire walk and MC only slightly reduces the level of gain applied (see Figure 4). As a result, MC does appear to favour position gain and used it a great deal in our simulations. As with the other metrics, MC and MC-Fast had similar results.

The superior boundary collision performance of the over-threshold methods MC-Lo, MC-Med and MC-Hi came with the cost of higher position gain overall. There was a possibility that these over-threshold methods would not dramatically increase total gain due to their scoring mechanism favouring low gain strategies. However, this is not borne out by the data, especially as in our simulations the MCRDW family of methods appeared to select redirection strategies with position gain frequently. This could perhaps be achieved through better weights, or a more intelligent scoring mechanism. Another possible explanation could be that due to the small tracking space, the user was almost always encountering a boundary so gain was frequently required.

Edge factor had a notable impact on the level of position gain applied (see Table 2 for effect sizes). However, this is likely a result of low standard deviations among position gain results across techniques. The difference in mean total position gain is small in practice. The likely explanation is that those layouts with higher edge factor more frequently placed the user in situations likely to lead to boundary collision. As a result, the small bias in favour of not applying translation gain is more frequently ignored.

5.7.3 Total Rotation Gain

Total rotation gain was similar across all heuristic techniques. MC had the lowest rotation gain overall by a small but significant factor, though this gap closed at higher edge factors. Interestingly at low edge factors, the over-threshold methods MC-Lo, MC-Med and MC-Hi saw slightly increased rotation gains overall, but significant performance gains. This may be a good argument for allowing at least rotation gain to slightly exceed established thresholds in challenging situations, as it is possible that MCRDW is able to only apply it in situations where it is necessary. However, confirming this would require additional outcome measures that were not included in this experiment. Again as with position gain and boundary collisions, no significant difference was found between MC and MC-Fast $p > 0.9999$.

Edge factor had a less notable impact on the level of rotation gain applied than it did with position gain (see Table 3 for effect sizes). However, as with position gain, the effect was particularly notable for the over-threshold methods. Likely this is for the same reason; layouts with higher edge factor more frequently put the simulated user on path towards boundary collision, overwhelming the small bias towards low gain strategies.

6 Conclusion

This paper introduces the MCRDW algorithm, a method for selecting redirection strategies which minimises boundary collisions through simulated walks. We include an evaluation of MCRDW under a variety of different conditions, and a comparison with existing methods. In our evaluation, the MCRDW family of methods significantly reduce boundary collisions when compared with the heuristic methods by over 50% while reducing total rotation or position gain. No significant difference was found between MC and MC-Fast on all metrics. MCRDW can therefore make good decisions without needing as much computation time as allotted within our simulations. The technique can be considered lightweight.

The over-threshold MCRDW methods MC-Lo, MC-Med and MC-Hi significantly outperformed all other techniques. However, these methods resorted to over-threshold gains too often, at the cost of significantly increasing total rotation and position gain. This behaviour could potentially be avoided by tweaking the associated 'score' penalties, or by introducing a more intelligent *subtlety* metric.

While MCRDW methods significantly outperformed heuristic methods at all levels of edge factor, MCRDW performance was more closely linked to the virtual environment than with heuristic methods. This is likely due to simple environments having fewer potential routes. This makes the simple path prediction used in our implementation more reliable. Quality of available path prediction should be considered a significant factor in MCRDW performance. Advanced path prediction techniques would be a valuable avenue of future research.

Acknowledgments

This project has received funding from the European Union's Horizon 2020 Research and Innovation program under grant agreement No 739578 (RISE). This work was also supported in part by the UK's EPSRC under project number EP/G037159/1.

References

[1] M. Azmandian, T. Grechkin, M. Bolas, and E. Suma. Physical space requirements for redirected walking: How size and shape affect performance. The Eurographics Association, 2015. doi: 10.2312/egve.20151315 2, 5

[2] M. Azmandian, T. Grechkin, M. Bolas, and E. Suma. Automated path prediction for redirected walking using navigation meshes. In *2016 IEEE Symposium on 3D User Interfaces, 3DUI 2016 - Proceedings*, pp. 63–66, 2016. doi: 10.1109/3DUI.2016.7460032 3

[3] G. Bruder, V. Interrante, L. Phillips, and F. Steinicke. Redirecting walking and driving for natural navigation in immersive virtual environments. *IEEE Transactions on Visualization and Computer Graphics*, 18(4):538–545, 2012. doi: 10.1109/TVCG.2012.55 3, 4

[4] B. J. Congdon and A. Steed. Sensitivity to rate of change in gains applied by redirected walking. Association for Computing Machinery, 2019. doi: 10.1145/3359996.3364277 5

[5] T. Field, S. Bay, and P. Vamplew. Generalised Algorithms for Redirected Walking in Virtual Environments. *AISAT2004 International Conference on Artificial Intelligence in Science and Technology*, 65(11):1357–1366, 2004. doi: 10.1111/j.1398-9995.2010.02441.x 2, 3, 5

[6] H. Hicheur, Q.-C. Pham, G. Arechavaleta, J.-P. Laumond, and A. Berthoz. The formation of trajectories during goal-oriented locomotion in humans. I. A stereotyped behaviour. *European Journal of Neuroscience*, 26(8):2376–2390, 2007. 3

[7] 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):634–43, 2013. doi: 10.1109/TVCG.2013.28 2, 3, 5, 8

[8] 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. doi: 10.1109/TVCG.2014.34 2

[9] E. Hodgson, E. Bachmann, and D. Waller. Redirected Walking to Explore Virtual Environments: Assessing the Potential for Spatial Interference. *ACM Trans. Appl. Percept.*, 8(4):22:1—-22:22, dec 2008. doi: 10.1145/2043603.2043604 2

[10] G. Keppel, W. H. S. Jr., and H. Tokunaga. *Introduction to design and analysis: A student's handbook, 2nd ed.* W H Freeman/Times Books/Henry Holt & Co, 1992. 6, 7

[11] D. Y. Lee, Y. H. Cho, and I. K. Lee. Real-time optimal planning for redirected walking using deep q-learning. In *26th IEEE Conference on Virtual Reality and 3D User Interfaces, VR 2019 - Proceedings*, 2019. doi: 10.1109/VR.2019.8798121 2, 5

[12] J. Messinger, E. Hodgson, and E. R. Bachmann. Effects of Tracking Area Shape and Size on Artificial Potential Field Redirected Walking. In *2019 IEEE Conference on Virtual Reality and 3D User Interfaces (VR)*, pp. 72–80, 2019. doi: 10.1109/VR.2019.8797818 2, 5

[13] 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 2014, 3DUI 2014 - Proceedings*, pp. 111–118, 2014. doi: 10.1109/3DUI.2014.6798851 2

[14] T. Nescher and A. Kunz. Using head tracking data for robust short term path prediction of human locomotion. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 7848, 2013. doi: 10.1007/978-3-642-38803-3-10 3

[15] 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. doi: 10.1109/MCG.2018.111125628 2

[16] T. C. Peck. *Redirected Free Exploration with Distractors : A Large-Scale, Real-Walking Locomotion Interface*. PhD thesis, The University of North Carolina at Chapel Hill, 2010. 3, 4

[17] S. Razzaque. *Redirected Walking*. PhD thesis, The University of North Carolina at Chapel Hill, 2005. 2, 3, 5

[18] S. Razzaque, Z. Kohn, and M. C. Whitton. Redirected Walking. *Proceedings of EUROGRAPHICS*, pp. 289–294, 2001. 2

[19] 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. doi: 10.1109/TVCG.2009.62 3, 4

[20] F. Steinicke, G. Bruder, T. Ropinski, and K. H. Hinrichs. Moving towards generally applicable redirected walking. In *Proceedings of the Virtual Reality International Conference (VRIC)*, pp. 15–24. IEEE Press, 2008. 2

[21] 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*, 2020. doi: 10.1109/TVCG.2020.2973060 2

[22] J. Su. Motion Compression for Telepresence Locomotion. *Presence: Teleoper. Virtual Environ.*, 16(4):385–398, aug 2007. doi: 10.1162/pres.16.4.385 3

[23] J. Thomas and E. S. Rosenberg. A General Reactive Algorithm for Redirected Walking Using Artificial Potential Functions. In *2019 IEEE Conference on Virtual Reality and 3D User Interfaces (VR)*, pp. 56–62, 2019. doi: 10.1109/VR.2019.8797983 2

[24] M. Usoh, K. Arthur, M. Whitton C., R. Bastos, A. Steed, M. Slater, and F. Brooks P. Walking > Walking-in-Place > Flying, in Virtual Environments. *SIGGRAPH '99 Proceedings of the 26th annual conference on Computer graphics and interactive techniques*, pp. 359–364, 1999. doi: 10.1145/311535.311589 2

[25] 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. *Proceedings of the 4th symposium on Applied perception in graphics and visualization - APGV '07*, 1(212):41, 2007. doi: 10.1145/1272582.1272590 5

[26] M. Zank and A. Kunz. Eye tracking for locomotion prediction in redirected walking. In *2016 IEEE Symposium on 3D User Interfaces, 3DUI 2016 - Proceedings*, pp. 49–58, 2016. doi: 10.1109/3DUI.2016.7460030 3

[27] M. Zank and A. Kunz. Using Locomotion Models for Estimating Walking Targets in Immersive Virtual Environments. In *Proceedings - 2015 International Conference on Cyberworlds, CW 2015*, pp. 229–236, 2016. doi: 10.1109/CW.2015.20 3

[28] 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. doi: 10.1109/TVCG.2013.88 2