Optimal Planning for Redirected Walking Based on Reinforcement Learning in Multi-user Environment with Irregularly Shaped Physical Space

Dong-Yong Lee*
Dept. of Computer Science
Yonsei University

Yong-Hun Cho†
Dept. of Computer Science
Yonsei University

Dae-Hong Min‡
Dept. of Computer Science
Yonsei University

In-Kwon Lee§
Dept. of Computer Science
Yonsei University

ABSTRACT

Redirected Walking (RDW) enables users to walk in both virtual and physical tracking spaces simultaneously, which is an effective method to increase presence in Virtual Reality (VR). Recently, RDW technologies have been developed in a multi-user environment where multiple users share the same physical tracking space and simultaneously explore the same virtual space. Meanwhile, in the Steer-To-Optimal-Target (S2OT) method, user actions are planned in RDW by introducing machine learning models such as reinforcement learning. In this paper, we propose a new predictive RDW algorithm "Multiuser-Steer-to-Optimal-Target (MS2OT)" that extends the S2OT method into an environment with multiple users and various types of tracking space. In addition to the steering actions used in S2OT, MS2OT considers pre-reset actions and uses more steering targets and an improved reward function. The locations of multiple users and tracking space information are treated as visual information to be the state of the reinforcement learning model in MS2OT. Hence, the artificial neural network of a multilayer three-dimensional convolutional neural network with a dueling double deep network architecture is learned through Q-Learning. MS2OT significantly reduces the total number of resets compared to the conventional RDW algorithms such as S2C and APF-RDW in a multi-user environment and improves the total distance and average distance between resets during the same period. Experimental results show that MS2OT can process up to 32 users in real-time.

Index Terms: Redirected walking, resetting, virtual environments, multi-user, collision avoidance, reinforcement learning

1 INTRODUCTION

In immersive Virtual Reality (VR), the virtual space that users experience is infinite, while the physical tracking space is limited. This spatial difference may increase the risk of physical collisions in the user experience of VR. Therefore, virtual space manipulation methods such as Redirected Walking (RDW) [26,27] and teleportation [7] are used in VR applications. Among them, RDW is a walking manipulation method that solves the spatial constraints of physical spaces by mapping virtual and real spaces non-identically [26, 27]. RDW uses the fact that a user wearing a Head-Mounted Display (HMD) does not recognize that their walking in the virtual world is different from that in the physical world, thereby preventing the user from a collision [26]. Typically, the RDW manipulates the user's path by continuously applying a level of rotation not perceived by the user at the user's viewpoint. The real walking-based manipulation provided by RDW provides a higher presence in the VR [15, 17].

*e-mail: pfldy2850@gmail.com
†e-mail: maxburst@gmail.com
‡e-mail: bingo904@yonsei.ac.kr
§e-mail: iklee@yonsei.ac.kr

Many studies have proposed various effective RDW algorithms. Most of them have been conducted in a single user environment with rectangular physical tracking space [5, 6, 24, 31]. RDW algorithms are classified into two types, reactive and predictive [25]. Reactive algorithms adjust the redirection techniques (RETs) immediately in the user's current state, including the famous Steer-To-Center (S2C) [26] and APF-RDW [3] algorithms. The predictive algorithms such as Model Predictive Control Redirection (MPCRed) [37] and Steer-to-optimal-target (S2OT) [19] select the best RET by considering the user's future path. In a single-user environment, predictive algorithms reported better collision avoidance performance than conventional reactive algorithms, but require a higher computation time [24, 37].

Recently, few studies on reactive RDW algorithms that work in multi-user environments [2–4, 21] and various physical tracking spaces [8, 21] have been conducted. Although the predictive RDW algorithms in single-user environments demonstrate excellent collision avoidance performance, they are challenging to scale to multi-user environments with various physical spaces of different shape types. This is because the time complexity for selecting the best RET for each user by combining the users' future path, state of the physical space, and RETs increases exponentially with the number of users and the type of physical space.

In this paper, we present a new predictive algorithm, Multi-user-Steer-to-Optimal-Target (MS2OT), that uses a machine-learning model trained using deep reinforcement learning [23] in a multi-user environment with various types of physical spaces. MS2OT is an extension of S2OT that considers a multi-user environment in a physical space of various shapes. In addition to the steering action used by the existing S2OT, MS2OT considers the pre-reset action and achieves an improved performance by using a large number of more general steering targets and an improved reward function. MS2OT comprises 2048 actions, including 1024 steering actions and 1024 pre-reset actions. The location of users and physical space information are treated as visual information to be the state of the reinforcement learning model, and the best action to be performed in this state is selected. Hence, the artificial neural network of a multilayer 3D Convolutional Neural Network (CNN) [13] with the Dueling Double Deep Network (D3QN) [34] architecture is learned through Q-Learning [35]. MS2OT is a fully functional multi-user RDW algorithm and is the first predictive RDW algorithm that takes into account multi-users.

For performance comparisons, we conducted two live-user experiments and two simulation experiments. MS2OT significantly reduces the total number of resets compared to S2C, S2OT and APF-RDW in a multi-user environment. In particular, MS2OT exploits a pre-reset action that resets users in advance to avoid resetting both users when they collide with each other, which reduces the number of resets between users significantly. Experimental results show that MS2OT works well in untrained physical spaces and can process up to 32 users in real-time.

2642-5254/20/$31.00 ©2020 IEEE
DOI 10.1109/VR46266.2020.00034

2 Background

RDW was first proposed by Razzaque et al. [27], which adds imperceptible rotation to a virtual environment, allowing a user to follow a curved path or perform a magnified or reduced rotation. In this way, the user can remain in the physical tracking space even if the virtual space is much larger than the physical tracking space. This approach has since been extended variously by other studies. Williams et al. [36] proposed reset techniques such as freeze-backup, freeze-turn, and 2:1 turn. The reset technique reorients the user's direction when the RDW technique fails to prevent user's collision. Typically, the 2:1 turn method physically rotates a user by 180 degrees but 360 degrees in a virtual space. By doing so, the user's orientation in the virtual space is maintained, and the user's orientation in the physical space is reversed.

Steinicke et al. [29] generalized the subtle changes used in RDW techniques into translation, rotation, and curvature, and identified the user's detection threshold for the gain of each action. Langbehn et al. [18] proposed a bending gain that, when it is moving along a curve in virtual space, it could be bent 4.4 times more than in a physical space. Subtle continuous RDW algorithms generally specify the amount of gain to be applied for translation, rotation, and curvature. These RDW algorithms are classified into reactive and predictive algorithms according to their characteristics.

Reactive algorithms are a way of redirecting users according to generalized rules or patterns, specifically adjusting the redirection technique (RET) immediately in the current state [25]. Typical examples of the reactive algorithms include Steer-to-Center (S2C) [26] that always leads the user to the center of the physical tracking space, Steer-to-Orbit (S2O) [26] that guides the user to orbit along a circle centered on the center of the physical tracking space, and APF-RDW algorithm [3,21,33] that applies Artificial Potential Field (APF) [16] to RDW. The APF-RDW algorithm treats other users and the boundaries of the physical tracking space as obstacles and induces users by using the repulsive force fields generated by the obstacles. The APF-RDW algorithm has significantly reduced the number of resets compared to S2C in multi-user environments and has shown a significant reduction in the number of resets in various shapes and sizes of physical space.

The predictive algorithm is a method proposed by Zmuda et al. [37]. Typical predictive algorithms include Fully Optimized Redirected Walking for Constrained Environment (FORCE) [37], Model Predictive Control Redirection (MPCRed) [24], and Steer-to-Optimal-Target (S2OT) [19]. FORCE evaluates the cost of possible RETs based on a search technique such as a search tree in a constrained space and selects the optimal RET. MPCRed sets 14 RETs as actions. One action is no redirection, two are general redirections for left and right, and the other 11 actions are the resets to different angles. Each action has a different cost for each type, and the actions are evaluated through the cost-to-go function, which is a probabilistic cost expectation. In particular, by separating the collision and reset situation, the infinite cost for the collision and the cost of 500 for the spontaneous reset are priortized for the spontaneous reset over the collision. We call these spontaneous resetting actions "pre-reset" to differentiate them from resets due to collisions. However, because of the high computation time complexity of MPCRed, a limitation exists for applying it in a real-time environment [24]. To solve this problem, S2OT [19] introduced deep Q-Learning that is a type of reinforcement learning to the RDW planning problem to enable real-time planning, resulting in improved performance in a single-user environment.

3 Value Based Reinforcement Learning for RDW

The predictive RDW system must select and apply one of several actions that the user can take at any moment. For example, the system may allow a user to move forward with some degree of curvature gain, or pre-reset to change a user's forward direction

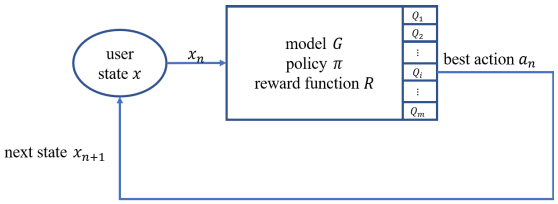


Figure 1: RDW system utilizing reinforcement learning model

to prevent future collisions with obstacles. When the user reaches the wall, the system may have to select an explicit reset, such as a 2:1 turn. Suppose there is a model G in our RDW system that can determine which action the user takes at any moment. The final goal of model G is to allow the user to explore the virtual space for a finite amount of time, minimizing the number of collisions and subsequent resets in the physical tracking area. How can we build this powerful model G? The effective way to construct model G in this type of problem is 'reinforcement learning.' [32]. Reinforcement learning builds up gradually the model G, which can be used to make decisions to reach the final goal every moment.

Lee et al. [19] proposed a Steer-To-Optimal-Target (S2OT) model, formulating RDW as a Markov Decision Process (MDP) and implementing deep Q-Learning which is a type of value-based reinforcement learning. For a RDW system using reinforcement learning, a variable $x_n(\in X)$ represents the particular state of the user ("agent" in general reinforcement learning system) at any time n. The system decides to take a specific action $a_n(\in A)$ at time n, which is one of several actions the user can choose. The selected action a_n causes the user to have a new state x_{n+1} at the next time step $n+1$. The mechanism used to determine the user's actions from the user's current state and environment is called policy $\pi(x_n) = a_n$. Policy π uses the reward function $R : X \times A \Rightarrow \mathbb{R}$ to select an action. The reward function computes the score that a user with current state x_n receives when the user takes action a_n.

The policy π should be able to predict all the cumulative rewards that will eventually be earned in the future when the user performs any action at any moment. This allows the system to choose the action that is most advantageous in its current state by considering the future. The following formula represents a cumulative reward:

$$\begin{aligned} V^{\pi}(x_n) &= \sum_{t=0}^{\infty} (\gamma^t R(x_{n+t}, \pi(x_{n+t})) \\ &= R(x_n, \pi(x_n)) + \gamma V^{\pi}(x_{n+1}), \end{aligned} \tag{1}$$

where $\gamma(\in [0:1])$ is the discount factor, which allows the system to consider the rewards of the currently selected action the most, and to gradually reduce the proportion of rewards earned over time when calculating the cumulative reward.

Q-function represents the expectation of cumulative discounted rewards when selecting action a_n in state x_n to configure the policy π, which is defined as $Q^{\pi}(x_n, a_n) = R(x_n, a_b) + \gamma \max_{a_{n+1}} V^{\pi}(x_{n+1})$. The optimal Q-value function Q^* recursively satisfies the following Bellman equation:

$$Q^*(x_n, a_n) = R(x_n, a_n) + \gamma \max_{a_{n+1}} Q^*(x_{n+1}, a_{n+1}). \tag{2}$$

Thus the optimal policy π^* can be derived as $\pi^* : x \mapsto \text{argmax}_a \, Q^*(x,a)$. In other words, the optimal policy π^* for state x selects the action a from the optimal Q-value function Q^* that derives the maximum expected Q-value. Therefore, in Q-Learning, it is crucial to derive the optimal Q-value function Q^*. Q-Network can be used to approximate the optimal Q-function to neural network [11]. Q-Network is defined as $Q(x,a;\theta) \approx Q^*(x,a)$, where θ represents the weights in the network.

Table 1: Comparison between S2OT and MS2OT

	S2OT	MS2OT
# of Users	single user	multi-users
Input for Model	numerical data	images
Shape of Phy. Space	rectangular	any types
Actions	steering	steering + pre-reset
Steering Targets	inside tracking area	inside + outside tracking area
# of Steering Targets	25	1024
# of Pre-reset Targets	–	1024
Q-Learning model	Double Deep Q-Learning	D3QN architecture

4 Methodology

In this paper, we propose MS2OT, which improves S2OT in several aspects. First, unlike S2OT, which considers only one user, MS2OT can handle the case where two or more users existing in the same physical space. To introduce an unfixed number of multiple users into the reinforcement learning model, MS2OT represent input information such as the user's positions and the boundary of the tracking space as an integrated input image, which allows us to build the model based on CNN, a powerful deep learning component. MS2OT also considers the cases where tracking spaces have various shapes and sizes. While S2OT considers only the steering action by the user, MS2OT allows the user to receive more rewards by selecting pre-reset actions as well as the steering actions, which helps to improve various performance indicators. MS2OT defines physical space by dividing it into an area that can be tracked and an area that is not tracked. While S2OT considers the steering actions into the targets only in the tracking area, the MS2OT allows for more efficient redirection by further considering the steering actions to guide the user to the non-tracking area. Finally, while S2OT uses the double deep Q-Learning model, MS2OT is implemented using the more advanced D3QN model [34] to compensate for the shortcomings of the existing method. Table 1 summarizes the differences between S2OT and MS2OT.

4.1 States

The state $X(=\{X_v, X_s\})$ used in the system contains visual information X_v about the users and the physical space as well as the state X_s used by the S2OT [19]. This information consists of a total of 64 consecutive frames and the interval between frames is 0.02 seconds. The total amount of physical space used by the MS2OT is 100×100 m^2. The total physical space P consists of the tracking area P_a and non-tracking area P_b ($P = P_a \cup P_b$ and $P_a \cap P_b = \emptyset$).

X_s includes the sequences of the user's location and direction in virtual space and the user's location and direction in physical space. For the visual information X_v, locations of users and physical space information are all represented as binary images of 64×64 in the preprocessing step (see Figure 2). User's position in a map is indicated by the pixel zero, as shown in Figure 2(a). The locations of other users are also represented by zeros, as in Figure 2(b). In the physical space information, as shown in Figure 2(c), P_b is indicated by pixels with zero values. Each piece of information is represented by one single channel, resulting in image data with three channels. Figure 2(d) shows the integrated image data.

4.2 Actions

The actions $A(=A_s \cup A_r)$ used in MS2OT are classified into two types: steering actions A_s and pre-reset actions A_r. There are a total of 2048 possible actions, and each action computes the reward for the action through different reward functions.

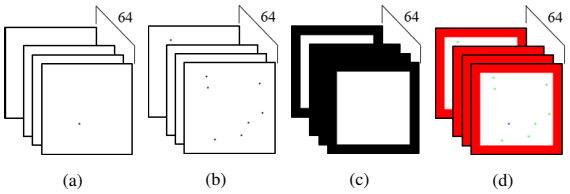


Figure 2: Representation of visual information X_v: (a) user's position, (b) other users' positions, (c) boundary (wall) of the tracking space, and (d) integrated image by combining three channels (a), (b), and (c).

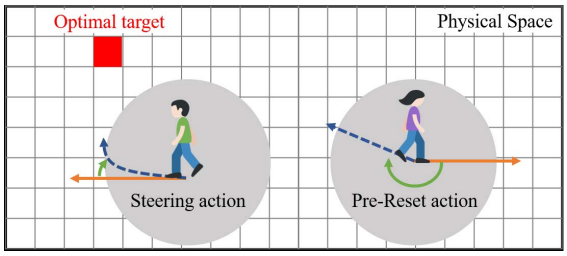


Figure 3: Steering and pre-reset action: a point in the physical space becomes the optimal target (red point), the steering action leads the user to the optimal target through redirection, and the pre-reset action 'resets' the user to the optimal target.

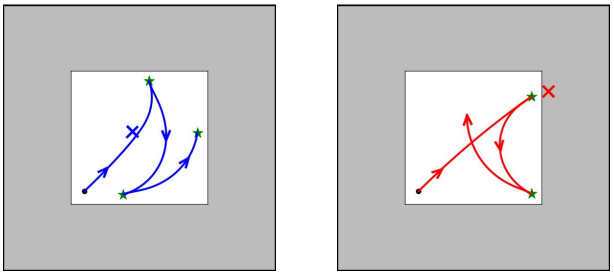
4.2.1 Steering Actions

Steering action A_s consists of 1024 actions, which are actions that induce the user to one of the different steering targets (see steering action in Figure 3). In S2OT [19], the future paths are predicted based on the user's route information, and one of 25 different candidates is selected as the optimal steering target. This method differs from the approach of traditional RDW algorithms, such as S2C and S2O [26], which always hold the fixed steering target. In MS2OT, the steer-to algorithm and dampening method proposed by Hudgson et al. [12] are used to guide the user to the steering target.

In MS2OT, the total physical space P is expressed as a grid of uniform cells of 32×32, and the center coordinates of each cell are set as candidate targets of the steering action. Since the total physical space P is divided into the tracking area P_a and the non-tracking area P_b, the steering action target may be within P_a, or within P_b (i.e, outside P_a) (note that in the case of S2OT [19], the steering target is always inside the tracking area). If we select a steering target inside P_b, the user is not able to proceed and reach that target point. This is because the user reaches the wall, the boundary of P_a, in the middle of the user's progress to the target point. However, in certain situations, selecting a steering target in a non-tracking area can reduce the number of times a user hits a wall in the long run. Figure 4(a) shows the moving path of a user that is redirected to a steering target within P_a. In Figure 4(a), the user collides with the wall three times if the user does not change the steering target while moving. On the other hand, in Figure 4(b), when the steering target is in P_b, the user collides with the wall only twice in the same time interval as in (a).

4.2.2 Pre-reset Actions

Pre-reset action A_r was first introduced in MPCRed algorithm [37]. In predictive algorithms, even if the user is not faced with a collision right now, sometimes there is a lot more benefit to doing a pre-



(a) (b)

Figure 4: User's path of travel based on the steering target's location: Black dots indicate the user's initial physical location. (a) shows the user's movement path (blue line) steering to the steering target (blue 'x' mark) in the tracking area, and (b) indicates the user's movement route (red line) steering to the steering target (red 'x' mark) in the non-tracking area. In both cases, the green asterisks indicate the collisions between the user and the walls. At the point of collision, a reset using a normal 2:1 turn occurs.

reset now when the algorithm can predict that the pre-reset prevents multiple collisions that may occur later. The pre-reset action of the MS2OT is a reset action performed toward one of 1024 different pre-reset targets. These pre-reset targets are the center coordinates of each cell dividing the physical space P by 32×32, as in the steering targets. The user is reorientated to the pre-reset target through the pre-reset action. That is, in the virtual space, the user rotates 360° to maintain the direction of progress, and at the same time, in the physical space, the user rotates toward the corresponding pre-reset target (see pre-reset action in Figure 3). In MPCRed, a pre-reset action can occur at 11 fixed angles, but MS2OT can generally take a pre-reset action in much more directions.

4.3 Reward Function

The reward function R used in the system is composed of two reward functions, R_s for steering action and $R_p = 0$ for pre-reset action, depending on the type of action applied:

$$R(a_{t-1}, x_t, a_t, x_{t+1}) = \begin{cases} R_s, & \text{for } a_t \in A_s \\ R_p = 0, & \text{for } a_t \in A_r. \end{cases} \tag{3}$$

The reward function R_s for steering action is reclassified according to whether the current action is causing the user to collide with something. If the current steering action does not cause a collision with the wall or other users, R_s is computed as the sum of the reward R_w for the distance between the user and the wall in the physical space, R_u for the density of the total users, R_r for the amount of change in the RET of the selected action, and the constant $R_0(=5)$. Conversely, if the current steering action causes a collision with the wall or other users, a constant penalty of $R_c = -10$ is given:

$$R_s(a_{t-1}, x_t, a_t, x_{t+1}) = \begin{cases} R_w + R_u + R_r + R_0, & \text{for } p_{t+1} \in P_a \\ R_c = -10 & \text{for } p_{t+1} \in P_b. \end{cases} \tag{4}$$

The three constant values of $R_0(=5)$, $R_p(=0)$ (in Equation (3)), and $R_c(=-10)$ ensure the relationship among the three kinds of rewards: steering action without collision > pre-reset > reset with collision.

Let us consider the more details about each reward terms. First, in RDW, the farther the user is from the wall of the tracking area, the safer the user is. In consideration of the safety of the user, R_w derives a higher reward value as the distance between the user and the wall increases. Let i be one of the line segments that make up the wall of the tracking area, and l_i is a point on i that is closest to the user's physical location p. Then, R_w is defined by:

$$R_w(a_{t-1}, x_t, a_t, x_{t+1}) = \frac{\sum_i \|p - l_i\|}{M_w}, \tag{5}$$

where, M_w represents the largest sum of the distances from the walls among all possible positions p in the physical space. Due to the denominator, the reward R_w can always be normalized to the range $[0:1]$. Note that M_w can be precomputed before starting the simulation.

Similar to R_w, the further away the user is from other users, the less likely they are to collide, and the more secure the user is. Therefore, we assign the reward term R_u to reduce the possibility of collision between a user and other users. R_u yields a higher value as the distance between the user and other users increases. When the user's physical location is p and other users' physical locations are o_j, R_u is defined as follows:

$$R_u(a_{t-1}, x_t, a_t, x_{t+1}) = \frac{\sum_j \|p - o_j\|}{M_u}, \tag{6}$$

where M_u is a value introduced for normalization that represents the maximum of all possible sums of distances among users in the physical space. M_u can also be computed before the simulation.

S2OT reported that when the steering target changes drastically, the user was able to recognize that he/she was being redirected [19]. This limitation can become more severe when there are many dynamic moving targets such as multiple users. To remedy this problem, our system assigns a reward term R_r to minimize the difference between the previous and subsequent steering targets. R_r derives the value closer to zero as the position difference between the previous steering target and the current steering target becomes larger, and is defined as follows:

$$R_r(a_{t-1}, x_t, a_t, x_{t+1}) = 1 - \frac{\|S_{a_t} - S_{a_{t-1}}\|}{32\sqrt{2}}, \tag{7}$$

where $\|\cdot\|$ denotes Euclidean distance, S_a is a target position of action a, and $32\sqrt{2}$ in the denominator is the maximum of the distances between the two possible steering targets in the 32×32 grid, which is used for normalization.

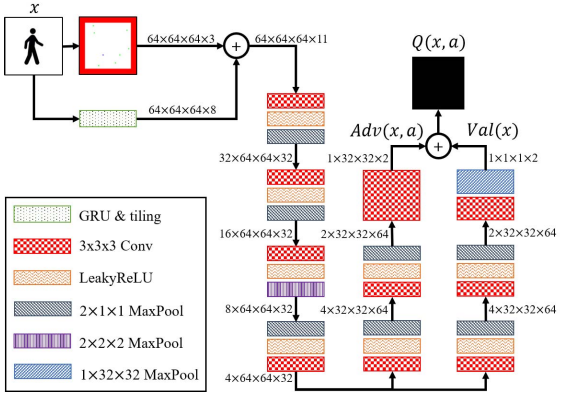


Figure 5: The Q-Network structure used in MS2OT model.

4.4 Network and Training

In the Q-Network used in the system (see Figure 5), X_s through the GRU layer is tiled and embedded in the rest of the preprocessed information X_v and input into a multi-layer 3D CNN having a D3QN

architecture. The 3D CNN extracts features between contiguous visual information in three dimensions through convolution, which leads to good performance in processing video that is a contiguous image frame [20]. MS2OT extracts state data into $(4 \times 32 \times 32 \times 64)$ features through four 3D CNNs. The extracted features are used to derive $Adv(x,a)$ in $(1 \times 32 \times 32 \times 2)$ and $Val(x)$ in $(1 \times 1 \times 1 \times 2)$ dimension of D3QN through two 3D CNNs, respectively. $Adv(x,a)$ computes the evaluation scores of each of the steering actions and the pre-reset actions, and $Val(x)$ derives an evaluation score on how good the state x is for applying steering and pre-reset actions. Finally, Q-value of each steering and pre-reset actions is derived through $Q(x,a) = Adv(x,a) + Val(x)$.

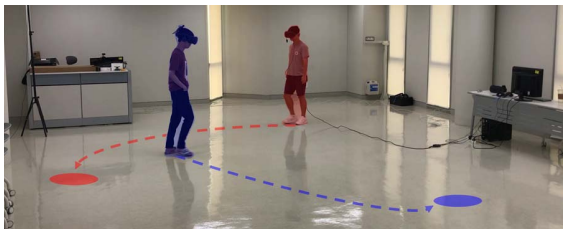
We designed a simulation environment in which reinforcement learning can occur. In each episode, simulated users walk toward 10 randomly occurring targets in a $100 \times 100\text{m}^2$ virtual space. The simulated users' initial virtual location, virtual direction, physical location and physical direction are randomly determined. Furthermore, the number of users ($1 \sim 10$), shape of the physical space (rectangle, trapezoid, cross, L-shaped, and C-shaped), and size of the physical space (400m^2, 900m^2, 1600m^2, and 2500m^2) are randomly determined. Each simulated user undergoes the MS2OT learning process, and walking data to reach the goal are generated using the walking model of Fink et al [10]. When they collide with the walls of the physical space as they progress, they proceed with a 2:1-Turn, and when the distance between users reaches 1.5m, they both proceed with a 2:1-Turn. To learn the Q-function, we used L1-norm loss function with learning rate 10^{-5}. A total of 160,000 episodes were performed in the learning. We used the discount factor $\gamma = 0.999$ in all learning process, which showed the best performance experimentally.

5 Experimental Evaluation

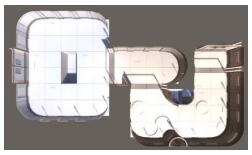
We conducted live-user experiments and simulation experiments to evaluate the performance of the MS2OT. Live-user experiments were conducted to investigate if the MS2OT could be applied to the actual environment and simulation experiments were conducted to analyze the collision avoidance performance of the MS2OT. For comparison, four RDW algorithms were considered: S2C, APF-RDW, S2OT, and MS2OT. Since there are few RDW algorithms for multi-users, we extended the single-user RDW methods S2C and S2OT to handle multi-users for comparison. The S2C algorithm is based on the Redirected Walking Toolkit [1], which implements Hodgson et al. [12]'s enhanced implementations of the widely used S2C algorithm. In a multi-user environment using S2C, users applied S2C independently using a sharing strategy with a common center [2]. For APF-RDW, Messinger et al. [21]'s APF-RDW with proximity scaling, which can be used in physical spaces of various sizes and shapes, was used. We implemented S2OT using the model proposed by Lee et al. [19]. Users executed S2OT independently in multi-user environment. The curvature gain threshold used in all algorithms is 7.5m and the rotation gain threshold is $[-0.2, 0.49]$ [30]. During the experiment, when the distance between the user and the wall reached 1.5m, the users performed a 2:1-Turn [36].

5.1 Live-User Experiments

We investigated whether MS2OT can be applied well in a live user experiment (see Figure 6(a)). We performed two types of real user experiments: one with a single user and another with two users. Both experiments were conducted in a square-shaped tracking area of $6 \times 6\ \text{m}^2$. The participants wore the HTC Vive HMD and were tracked by lighthouse base stations. Figure 6(b) shows a virtual space of about $1400\ \text{m}^2$ designed for the experiment. The virtual space comprised some passages (see Figure 6(b)), and the participants were instructed to move around the virtual space freely for one minute in both experiments. Pre-reset action in MS2OT showed the guide screen of Figure 6(c) to a user with the direction to rotate.



(a)



(b)

(c)

Figure 6: (a) Two users are being redirected to explore the virtual environment. (b) Virtual environment used in live-user experiments. (c) The guide screen is shown to the user during the pre-reset action of the MS2OT algorithm, and the user is instructed to rotate in the direction of the arrow.

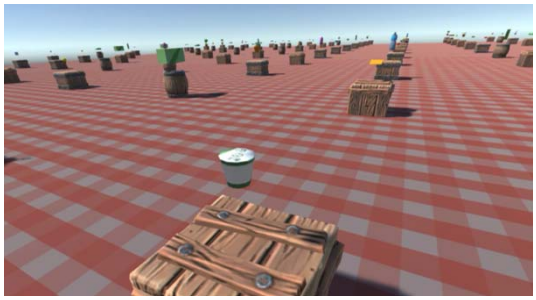


Figure 7: Virtual environment for collecting navigational data from simulated users. The user performs a task to get all of the indicated objects.

For each trial, data was logged with the number of total resets, wall resets, user resets, pre-resets and computation time. In particular, we asked users to answer the Simulator Sickness Questionnaire (SSQ) [14] before and after the experiment. In the multiple user experiment, we experimented with two users, which is known as the maximum number of users that can be placed in a square tracking area $6 \times 6\text{m}^2$ [21]. We established two main hypotheses based on the criteria we want to evaluate. The results of both live-user experiments were used to test these hypotheses:

- **H1**: The increase of sickness produced by MS2OT will be the same as the other three algorithms. Since all algorithms use the same limit of redirection gains, the sickness provided by the four algorithms will be the same.
- **H2**: For a single user, all four algorithms will perform similarly. But for two users, the MS2OT will outperform the S2C and S2OT.

5.2 Simulation Experiments

We performed simulation experiments to compare the performances of various algorithms under various conditions. The data used in the simulation experiment were the live data collected from actual user walking paths. The virtual environment experienced by users was

a $80 \times 80\text{m}^2$ space, in which boxes with miscellaneous goods were placed every 2m (See Figure 7). In this space, users performed tasks that brought all of the indicated objects. Tasks were conducted in a $6 \times 6\text{m}^2$ physical space, and the user overcame the space limitation using a 2:1-Turn. A total of 44 real user walking paths were collected, where one path comprises data sampled for 1 minute at 0.02 second intervals for the user's location and viewing direction except motion data during 2:1-Turn resets. To compare the performance of the MS2OT, we conducted simulation experiments with the following two conditions:

- **Experiments with Varied Room Sizes** measured the performance of the algorithms in varied room sizes. The room sizes used in the experiments were $20 \times 20\text{m}^2$, $30 \times 30\text{m}^2$, $40 \times 40\text{m}^2$ and $50 \times 50\text{m}^2$ and the room shape was fixed as square. Experiments were conducted for 1 to 10 users.
- **Experiments with Varied Room Shapes** measured the performance of the algorithms in varied room shapes. The room shapes used in the experiment were square, trapezoid, Cross, L-shaped and circular, and the room size was fixed at about 400m^2.

To create simulation data for a multi-user environment, we combined the 44 live user walk paths without redundancy. The initial positions and directions of the users were set to randomly selected positions where no initial collision occurred. Of all these combinations, we selected 50 random combinations and used them as navigational data of simulated users. Therefore, the virtual space used in the simulation experiment is the same open and unrestricted space as Figure 7, where the actual user walks were collected. For each trial, data was logged with the number of total resets, wall resets, user resets, pre-resets and computation time. We established two main hypotheses based on the criteria we want to evaluate. The results of both simulation experiments were used to test these hypotheses:

- **H3**: In the case of a single simulated user, MS2OT will perform similarly to S2OT in a square physical space of the same size. S2C will have lower collision performance than the other three algorithms.
- **H4**: In the case of multiple simulated users, MS2OT will yield better performance than S2OT because MS2OT considers other users but S2OT does not consider other users. In addition, MS2OT will perform better than RDW-APF and S2C because collisions can be reduced through prediction-based actions such as the pre-reset.
- **H5**: S2OT will perform similarly to MS2OT in the convex space but will not perform well in the concave space. MS2OT will perform well in the space of the shape used for learning, but will perform poorly in spaces that were not.

6 Results

6.1 Results of Live-User Experiments

A total of 24 users participated in the live user experiment. The participants were 23 to 29 years old, 22 of whom were male, and 2 female. All participants had normal or corrected vision. Most participants had little experience in VR. Each user performed one trial for each algorithm of S2C, APF-RDW, and MS2OT in a single user experiment. Therefore, a total of 72 trials were conducted in the single user experiment. In the multiple user experiments, a total of 12 pairs performed one trial for each of S2C, APF-RDW, and MS2OT respectively. As a result, a total of 72 user trials were conducted in the multi-user environment.

First, to verify the hypothesis **H1**, we conducted a one-way ANOVA test on experimental results for SSQ differences of the four algorithms. The mean of the pre-SSQ score was 1.00 out of 48, and the post SSQ score was S2C($M = 2.125, SD = 0.971$), APF($M = 2.917, SD = 1.077$), S2OT($M = 2.833, SD = 1.247$), and MS2OT($M = 2.792, SD = 0.957$). No statistically significant difference was found in the change of SSQ scores among different algorithms ($F(3,92) = 1.582, p = 0.199$). Therefore, the hypothesis **H1** was correct.

To verify **H2**, we performed a 4 (algorithm) $\times$ 2(number of users) two-way ANOVA test for the total number of resets. The algorithm ($F(3,184) = 3.634, p < 0.05$) and the number of users ($F(1,184) = 19.201, p < 0.05$) had a large main effect on the total number of resets, and the interaction effect was also significant ($F(3,184) = 8.904, p < 0.05$). As a result of multiple post hoc comparison analyses through Tukey's HSD test, for the single user, MS2OT($M = 5.518$) = S2OT($M = 5.742$) = APF($M = 5.873$) < S2C($M = 6.711$) ($p < 0.05$). For two users, the total resets increased in the order of MS2OT($M = 5.981$) = APF($M = 6.409$) < S2OT($M = 6.889$) = S2C($M = 7.585$) ($p < 0.05$). As a result, no significant difference in performance was found for single users, and for two users, MS2OT performed better than S2C and S2OT. Therefore, hypothesis **H2** was satisfied.

6.2 Results of Simulation Experiments

6.2.1 Experiments with Varied Room Sizes

In this experiments, we obtained 50 simulation results each according to the algorithm, room size and number of users. The samples from each group passed Shapiro's normality test, which satisfied the assumptions for the ANOVA test. First, to verify hypothesis **H3**, we conducted a 4 (algorithm) $\times$ 4 (room size) two-way ANOVA test for a single simulated user. The algorithm ($F(3,784) = 299.372, p < 0.001, \eta_p^2 = .534$) and room size ($F(3,784) = 599.009, p < 0.001, \eta_p^2 = .696$) had a large main effect. Furthermore, there was also a significant interaction effect on the algorithm and the room size ($F(9,784) = 76.210, p < 0.001, \eta_p^2 = .467$). As a result of the multiple post hoc comparison analyses through Tukey's HSD test, the total reset count increased in the order of MS2OT = S2OT < APF < S2C. MS2OT and S2OT were the same homogeneous subset. In particular, the average number of total resets in S2C was as high as 0.567, 0.538, and 0.147, compared to MS2OT, S2OT, and APF, respectively. The larger the space, the smaller the total number of resets, and the four spaces were all significantly different from each other($p < 0.05$). These results satisfied the hypothesis **H3**. In the case of single user environment, S2OT and MS2OT had similar states and reward functions in the square physical space, and as a result, the difference in the total reset count was not significant($p = 0.592$).

Next, to verify **H4**, we performed 4 (algorithm) $\times$ 4 (room size) $\times$ 9 (number of users) ANOVA tests for multiple users. The algorithms ($F(3,7056) = 46553.926, p < 0.001, \eta_p^2 = .952$) and room sizes ($F(3,7056) = 159404.229, p < 0.001, \eta_p^2 = .985$) and the number of users ($F(8,7056) = 31932.459, p < 0.001, \eta_p^2 = .973$) had large main effects. Additionally, all two-way and three-way interactions were also significant ($p < 0.001$). As a result of the multiple post hoc comparison analysis through Tukey's HSD test, there was a clear order of the total reset count in the order of MS2OT < APF < S2OT < S2C ($p < 0.05$). Multiple linear regression analysis was performed on the total number of resets for each algorithm according to the size of the room and the number of users. Significant regressions for the total number of resets of each algorithm were obtained ($p < 0.001$):

- **S2C**: $\text{T} = -0.003\text{S} + 0.748\text{U} + 3.723$, ($R^2 = .871$)
- **S2OT**: $\text{T} = -0.003\text{S} + 0.670\text{U} + 3.383$, ($R^2 = .841$)
- **APF**: $\text{T} = -0.002\text{S} + 0.441\text{U} + 2.367$, ($R^2 = .655$)

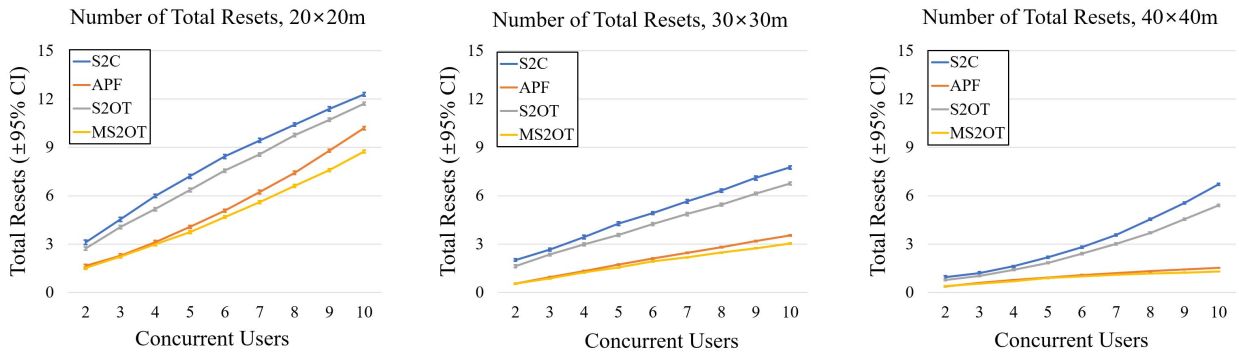


Figure 8: Mean of total resets per minute with each algorithm for different numbers of users and varied room sizes.

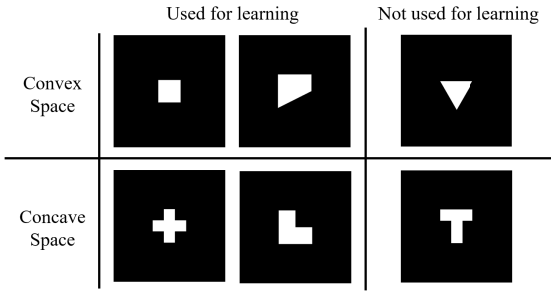


Figure 9: Different types of space shapes were used in the experiments with varied room shapes. The convex space included rectangles, trapezoids, and triangles, whereas the concave space has crosses, L-shapes, T-shapes. The shapes that were not used for learning were triangle and T-shapes.

- **MS2OT**: $\mathrm{T} = -0.002\mathrm{S} + 0.374\mathrm{U} + 2.267$, ($R^2 = .652$)

, where T represents the number of total resets, S and U represent the room sizes and the number of users respectively. R^2 is a measure of how well the estimated linear model fits the given data. Chin et al. [9] recommended R^2 values for endogenous latent variables based on: 0.67 (substantial), 0.33 (moderate), 0.19 (weak). Therefore, the above multiple regressions above were substantial. In all four algorithms, the total number of resets decreased as the size of the space increased, and the total number of resets increased as the number of users increased. Especially for MS2OT, the beta coefficient for the number of users was 0.374, which was significantly different from that for the number of users of S2C, S2OT, and APF ($p < 0.001$). That is, compared with the other three algorithms, the increase in the total number of resets of the MS2OT was significantly smaller as the number of users increased (See Figure 10). In case of S2C and S2OT not considering other users, user resets occurs more frequently than wall resets. In case of APF and MS2OT, user resets occurs more rarely than wall resets. Consequently, MS2OT reduces the total number of resets compared to the other three algorithms; therefore, the hypothesis **H4** was corrected.

6.2.2 Experiments with Varied Room Shapes

In this experiments, we obtained 50 simulation results for each algorithm and the shape of the room. The six shapes of the room used in the experiment were rectangle, trapezoid, triangle, cross, L-shape and T-shape. Among them, rectangle, trapezoid and triangle were used for the convex space, and cross, L-shapes and T-shapes for the concave space. Furthermore, the shapes used for reinforcement learning were rectangle, trapezoid, cross, and L-shape. The triangle and T-shape were space shapes not used for learning (see Figure 9). All spaces were scaled to approximately 400m^2.

To verify **H5**, we performed a 4(algorithms) $\times$ 6(shapes of room) two-way ANOVA test for the results of the total resets. The algorithms ($F(3,1176) = 430.475, p < 0.001, \eta_p^2 = .523$) and the shapes of the room ($F(5,1176) = 1639.048, p < 0.001, \eta_p^2 = .875$) had significant main effects on the total number of resets. The interaction effect between the algorithms and the shapes of the room was also significant ($F(15,1176) = 63.948, p < 0.001, \eta_p^2 = .449$). As a result of the multiple post hoc comparison analysis through Tukey's HSD test indicated that; the total reset count was increased in the order of rectangle $<$ trapezoid $=$ T-shape $<$ triangle $<$ L-shape $<$ cross. The total number of resets in trapezoid and T-shape was the same homogeneous subset, while the total number of resets in different shapes were significantly different in the homogeneous subsets. In the concave space, the total number of resets increased compared with that in the convex space except for triangle. The algorithm increased the total number of resets in the order of MS2OT $<$ APF $<$ S2OT $<$ S2C ($p < 0.05$).

For the space that was not used for learning, the largest total reset of convex space occurred in triangle space, but the smallest total reset of concave space occurred in T-shape space. In the triangle space, the total number of resets of the four algorithms differed significantly ($p < 0.05$), and the total number of resets increased in the order of MS2OT $<$ APF $<$ S2OT $<$ S2C. In the T-shaped space, the total number of resets increased in the order of MS2OT $=$ APF $<$ S2OT $=$ S2C. In conclusion, S2OT performed similarly to MS2OT in the rectangular shapes of convex spaces, but generated higher total reset counts than MS2OT in the trapezoid and triangle shapes. In addition, MS2OT not only generated the lowest total reset count for the space used for learning, but also generated the lowest total reset count for the space not used for learning. The similarity between the performances of MS2OT and S2OT in convex space in Hypothesis **H5** was partially correct, and the prediction that MS2OT will not perform well for shapes that were not used for learning was incorrect.

7 Discussions

Based on the experimental results for live user experiments and simulation experiments, the hypothesis **H1**, **H2**, **H3**, **H4** were satisfied and **H5** was partially satisfied. In live user experiments, the motion sickness produced by MS2OT was not significantly different from the other three algorithms (**H1**). For the single user in the 6×6m^2 physical space, all four algorithms showed similar collision avoidance performance, but for two users, MS2OT and APF were better than S2C and S2OT (**H2**). Simulation experiments showed the collision avoidance performance of the algorithms according to

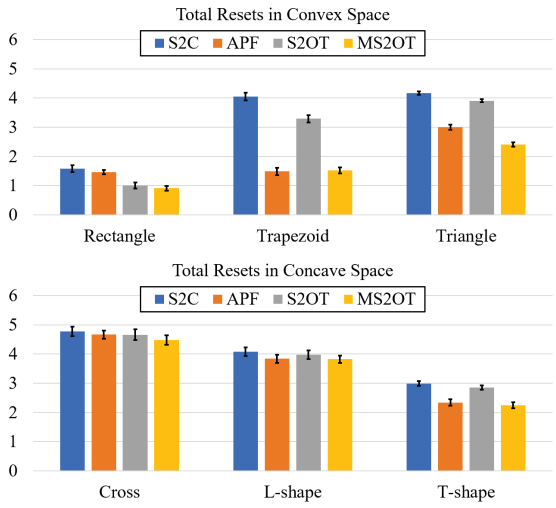


Figure 10: Mean of total resets per minute with each algorithm for different room shapes with the 95% confidence interval.

room size, room shape, and number of users. In the case of single user, MS2OT exhibited a collision avoidance performance similar to that of S2OT in a square physical space of the same size, and S2C had lower collision performance than the other three algorithms (**H3**). In the multi-user environment, the MS2OT performed significantly better in collision avoidance than the other three algorithms regardless of the size of the space (**H4**). Hypothesis **H5** was partially correct, because MS2OT provided the lowest overall reset counts for the four types of physical space shapes used for learning and the two types of physical space shapes that were not used.

The total reset count of MS2OT was significantly affected by the number of pre-reset actions. Hence, compared with existing algorithms, the number of resets due to collisions with other users was reduced significantly than that with walls. During learning process, the collision with the wall increases the reset frequency of one user by one, but the collision with other users increases the reset frequency of at least two users one by one. Therefore, the model was trained to minimize the number of resets due to user collisions in general. In a single-user environment, MS2OT has better collision avoidance performance than S2OT because there are more actions that MS2OT considers than actions of S2OT. Alternative reinforcement learning methods such as PPO [28] and A3C [22] can handle continuous action spaces. Using these methods, we could achieve more precise planning by handling the redirecting gain directly into the action space rather than the action that specifies the steering target used by the MS2OT. As the number of users increases, the MS2OT reduces the frequency of reset more significantly than S2C, S2OT and APF. This is because the performance of all algorithms decreases as the number of users increases, and the performance of MS2OT, which suppresses reset between users through pre-reset actions, is relatively higher than that of other algorithms. In live-user experiments, MS2OT averaged 5.981 resets per minute for two users in rectangular space of 6×6 m^2. In other words, considering about 10 minutes of VR experience, we expect about 60 resets per user. In the experiment, the reset time was averaged about 1 second, so in the 10-minute VR experience, the time spent on reset was about 10%, which is about 1 minute.

The computational time of the MS2OT was 9.262 msec per frame on average for all scenarios using 2 to 10 users, which was much longer compared to 0.118 msec in S2C and 1.434 msec in APF. However, 9.262 msec per frame can accommodate approximately 107 frames per second, which is sufficient to accommodate the 50 fps used in simulation experiments. As a result of analyzing the computation time for the number of users and the size of the tracking space through ordinary least square (OLS) linear regression, both the constant term and the number of users had a significant effect on the regression equation with $p < 0.001$. However, the size of the tracking space was -2.483×10^{-6}, which had no significant effect on the regression equation ($p = 0.595$). As a result, we get the equation of linear regression: $T = 0.3992U + 6.8668$, ($R^2 = 0.880$), where T is computation time, U is the number of users, and $R^2 \in [0:1]$ is the proportion of the variance in the dependent variable that is predictable from the independent variables. In general, 0.3992 msec of computation time is added for each additional user. According to the regression equation, the MS2OT can accommodate 50 fps for approximately 32 users, regardless of the size of the space.

8 Conclusion

In this study, we extended the RDW predictive algorithm to work in multi-user environments with tracking spaces of various shapes and sizes using reinforcement learning. Our MS2OT algorithm uses 1024 steering targets and 1024 pre-reset targets to select the optimal steering or pre-reset action to avoid the collision as much as possible. The selection policy was approximated by learning a multi-layered 3D CNN using Q-Learning with the D3QN structure. We conducted simulation and live user experiments to compare the performance of the MS2OT against those of S2C, S2OT and APF-RDW. Compared to S2C, S2OT and APF-RDW, the MS2OT significantly reduces the total number of resets in most situations. MS2OT could voluntarily select a pre-reset action, avoiding many potential collisions in the future. The total number of resets in MS2OT was affected by the number of pre-reset actions. In addition, the steering action of MS2OT was similar to that of APF-RDW.

In our work, due to the lack of experimental equipment and space, live-user experiments could only be performed under limited conditions (room-scale square-shaped physical space, up to two users). More live-user experiments with more than two users and different types of physical spaces are required. In some cases, inappropriate behaviors were observed, such as three consecutive pre-reset operations during the operation of MS2OT. By adding some penalties to the successive pre-reset operation, MS2OT could be improved to perform more naturally. In addition, using the 3D CNN layers to process a series of images required considerable computation time. Using two GeForce 1080 GPUs, we obtain 9.262 msec per frame on average. MS2OT could accommodate approximately 50 fps in a multi-user environment of 32 or fewer number of users, but it is hard to work in real-time in environments with more users than 32 or when GPU was not available. It is necessary to accelerate the computation by improving the MS2OT model using more advanced network structures. MS2OT did not consider the internal structure in the virtual space and potential points of interest of the user. Taking these factors into account can improve the performance of predicting future user behavior, which allows us to design more efficient RDW algorithms.

Acknowledgments

This research was supported by the MSIT (Ministry of Science and ICT), Korea, under the ITRC (Information Technology Research Center) support program (IITP-2018-2018-0-01419) supervised by the IITP (Institute for Information & communications Technology Promotion)

References

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

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

[3] 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.

[4] E. R. Bachmann, J. Holm, M. A. Zmuda, and E. Hodgson. Collision prediction and prevention in a simultaneous two-user immersive virtual environment. In *2013 IEEE Virtual Reality (VR)*, pp. 89–90. IEEE, 2013.

[5] M. Bhatt. *Redirected Walking in Virtual Reality during eye blinking*. PhD thesis, University of Bremen, 2016.

[6] B. Bolte and M. Lappe. Subliminal reorientation and repositioning in immersive virtual environments using saccadic suppression. *IEEE Transactions on Visualization and Computer Graphics*, 21(4):545–552, 2015.

[7] E. Bozgeyikli, A. Raij, S. Katkoori, and R. Dubey. Point & teleport locomotion technique for virtual reality. In *Proceedings of the 2016 Annual Symposium on Computer-Human Interaction in Play*, pp. 205–216. ACM, 2016.

[8] H. Chen, S. Chen, and E. S. Rosenberg. Redirected walking strategies in irregularly shaped and dynamic physical environments. In *25th IEEE Conference on Virtual Reality and 3D User Interfaces (VR 2018). Workshop on Everyday Virtual Reality*, 2018.

[9] W. W. Chin et al. The partial least squares approach to structural equation modeling. *Modern methods for business research*, 295(2):295–336, 1998.

[10] P. W. Fink, P. S. Foo, and W. H. Warren. Obstacle Avoidance During Walking in Real and Virtual Environments. *ACM Transactions on Applied Perception (TAP)*, 4(1):2, 2007.

[11] M. Hausknecht and P. Stone. Deep recurrent q-learning for partially observable mdps. In *2015 AAAI Fall Symposium Series*, 2015.

[12] 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.

[13] S. Ji, W. Xu, M. Yang, and K. Yu. 3d convolutional neural networks for human action recognition. *IEEE Transactions on pattern analysis and machine intelligence*, 35(1):221–231, 2012.

[14] R. S. Kennedy and J. E. Fowlkes. Simulator Sickness Is Polygenic and polysymptomatic: Implications for Research. *The International Journal of Aviation Psychology*, 2(1):23–38, 1992.

[15] B. Keshavarz, B. E. Riecke, L. J. Hettinger, and J. L. Campos. Vection and visually induced motion sickness: how are they related? *Frontiers in psychology*, 6:472, 2015.

[16] B. Krogh. A generalized potential field approach to obstacle avoidance control. In *Proc. SME Conf. on Robotics Research: The Next Five Years and Beyond, Bethlehem, PA, 1984*, pp. 11–22, 1984.

[17] E. Langbehn, P. Lubos, G. Bruder, and F. Steinicke. Application of redirected walking in room-scale vr. In *2017 IEEE Virtual Reality (VR)*, pp. 449–450. IEEE, 2017.

[18] 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.

[19] 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)*. IEEE, 2019.

[20] D. Maturana and S. Scherer. Voxnet: A 3d convolutional neural network for real-time object recognition. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 922–928. IEEE, 2015.

[21] 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)*. IEEE, 2019.

[22] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *International conference on machine learning*, pp. 1928–1937, 2016.

[23] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529, 2015.

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

[25] 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.

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

[27] S. Razzaque, Z. Kohn, and M. C. Whitton. Redirected Walking. In *Eurographics 2001 - Short Presentations*. Eurographics Association, 2001. doi: 10.2312/egs.20011036

[28] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[29] F. Steinicke, G. Bruder, K. Hinrichs, J. Jerald, H. Frenz, and M. Lappe. Real Walking through Virtual Environments by Redirection Techniques. *JVRB-Journal of Virtual Reality and Broadcasting*, 6(2), 2009.

[30] E. A. Suma, G. Bruder, F. Steinicke, D. M. Krum, and M. Bolas. A taxonomy for deploying redirection techniques in immersive virtual environments. In *2012 IEEE Virtual Reality Workshops (VRW)*, pp. 43–46. IEEE, 2012.

[31] 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):67, 2018.

[32] R. S. Sutton, A. G. Barto, et al. *Introduction to reinforcement learning*, vol. 2. MIT press Cambridge, 1998.

[33] Thomas, Jerald and Rosenberg, Evan Suma. 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. IEEE, 2019.

[34] Z. Wang, T. Schaul, M. Hessel, H. Van Hasselt, M. Lanctot, and N. De Freitas. Dueling network architectures for deep reinforcement learning. *arXiv preprint arXiv:1511.06581*, 2015.

[35] C. J. Watkins and P. Dayan. Q-learning. *Machine learning*, 8(3-4):279–292, 1992.

[36] 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.

[37] 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.