Reinforcement Learning-Based Redirection Controller for Efficient Redirected Walking in Virtual Maze Environment*

Wataru Shibayama and Shinichi Shirakawa

Graduate School of Environment and Information Sciences,
Yokohama National University, Yokohama, Japan
xsssx841@gmail.com, shirakawa-shinichi-bg@ynu.ac.jp

Abstract. Redirected walking (RDW) is a locomotion technique used in virtual reality (VR) that enables users to explore large virtual environments in a limited physical space. Existing RDW techniques mainly work on the obstacle-free physical spaces larger than a square of four-meter sides. To improve usability, RDW techniques that work on comparatively smaller physical spaces with obstacles need to be developed. In RDW, users are restricted to the physical space by redirection techniques (RETs) that control the view of the head-mounted display. Reinforcement learning, a branch of machine learning techniques, is advantageous in designing efficient redirection controllers compared to manual design. In this paper, we propose a reinforcement learning-based redirection controller (RLRC) that aims to realize an efficient RDW in small physical spaces. The controller is trained using the simulator and is expected to select an appropriate redirection technique from the current state and route information of the virtual environment. We evaluate the RLRC with simulator and user tests in a virtual maze in several physical spaces, including a square physical space of four-meter sides with an obstacle, and a square physical space of two-meter sides. The simulator test shows that the proposed RLRC can reduce the number of undesirable redirection techniques performed compared with existing methods. The proposed RLRC is found to be effective in the square physical space of two-meter sides in the user test.

Keywords: Redirected Walking · Reinforcement Learning · Virtual Environment · Virtual Reality

1 Introduction

Virtual reality (VR) has become popular with the development of reasonable head-mounted displays (HMDs). Locomotion techniques that enable users to experience a large virtual environment in limited physical spaces have recently

* This is an author version of the paper accepted to the Computer Graphics International (CGI) 2020. The final authenticated version is available online at https://doi.org/10.1007/978-3-030-61864-3_4.

attracted attention. A simple solution is to use a keyboard or a gamepad for long-distance moves, or to allow teleportation in the virtual environment. However, these techniques reduce the immersiveness of the VR experience. Specialized devices such as omnidirectional treadmills have been developed, which enable completely free walking in limited physical spaces, but these devices are expensive and have high installation costs.

Redirected walking (RDW) [12] is a promising technology to realize free walking at low costs without compromising immersion. In RDW, users are guided not to get out of the tracked area and avoid collisions with physical walls and obstacles by controlling the views in the HMDs. The methods to redirect users are called redirection techniques (RETs); various algorithms for controlling RETs have been proposed. A simple algorithm is steer-to-center (S2C), which applies RETs such that users always head to the center of the physical space. Sophisticated algorithms [10,18] can control RETs more effectively than S2C by predicting the user's future path from their current state. However, the computational costs of these algorithms are too expensive for real-time calculations. A few recent works have attempted to construct a control rule for RETs using reinforcement learning algorithms to realize efficient RDWs with reasonable computational costs. The reinforcement learning-based approach trains a redirection controller in the training phase; the trained controller can then be applied to control the RETs at a low computational cost. The steer-to-optimal-target (S2OT) algorithm proposed in [9] is a redirection method using reinforcement learning and shows promising performance. The controller in S2OT selects the target area in the physical space, and the user is guided to move to the selected area using RETs. Obstacles in the physical space are not considered in S2OT. The redirection controllers proposed in [1,15] directly control the parameters of RETs, and the controller is trained by a reinforcement learning algorithm. However, these methods are applied in relatively large physical spaces ($> 4 \times 4$ m^2) in relatively simple virtual environments without complicated routes such as mazes; moreover, they are only evaluated using simulators. It is not clear whether the existing reinforcement learning-based approaches for RDWs can handle small physical spaces or physical space with obstacles. Developing advanced reinforcement learning-based redirection controllers and evaluating them in various situations are important to realize efficient RDWs.

In this paper, we propose a reinforcement learning-based redirection controller (RLRC) for RDWs. The policy for controlling the RETs is directly trained in the simulator with a virtual maze environment; this leverages the route information to select the RETs. The efficiency of the proposed RLRC is verified with simulator and user tests comprising several types of physical spaces, including square physical spaces of 4×4 m^2 with an obstacle, and 2×2 m^2 that have not been considered in previous studies. The experimental results of the simulator test shows that the RLRC can reduce the number of undesirable redirection techniques compared with S2C and S2OT. Moreover, the RLRC was observed to be effective in the 2×2 m^2 physical space of the user test. Our results im-

ply that the reinforcement learning-based approach is promising for RDWs in various physical spaces.

2 Algorithms for Redirected Walking

Redirected walking (RDW) [12] was proposed for free walking in virtual environments within limited physical spaces. Applying subtle manipulations, called redirection techniques (RETs), to the view on the HMD, the users unconsciously change the movement directions or speeds. RETs are used to keep users away from physical boundaries such as the border of the tracked area or walls in the physical space. RETs contain rotation, curvature, translation redirections and reset techniques. Rotation and translation redirections control the user's rotation angle and walk speed, respectively. Curvature redirection is applied to turn the user's walking paths to the right or left. It creates a situation where users go straight in virtual environments but walk on curved paths in the physical space. These RETs have detection thresholds to adjust the redirection strength [14]. In addition, if the user is likely to collide with a physical boundary, a reset technique is activated to reorient or transfer the user toward a safe direction or position.

Various redirection algorithms to control RETs have been proposed. Two simple algorithms are the S2C and steer-to-orbit (S2O). The S2C algorithm controls the RETs so that a user always walks toward the center of the tracked area, and the S2O algorithm controls the RETs so that a user goes around in circles in the tracked area. In addition, both algorithms activate the reset technique if the user moves out of a predetermined safe area. These simple algorithms may apply unnecessary RETs and lead to VR sickness. Therefore, more sophisticated algorithms, such as fully optimized redirected walking for constrained environment (FORCE) [18] and model predictive control redirection (MPCRed) [10], have been developed to control RETs more efficiently. These algorithms predict the walking paths from the user's position, direction, and route on the virtual environment, and decide optimal RETs dynamically. Furthermore, Lee et al. [9] pointed out that the computational cost of these dynamic methods is expensive, and proposed the S2OT algorithm using reinforcement learning. In S2OT, the tracked area is divided into a 6×6 grid; grid points are set as the target candidates to redirect users. The policy for selecting the optimal target area is trained using double deep Q-learning [4]. Experimental results in [9] show that S2OT could reduce the frequency of the reset technique. In contrast to S2OT, the method proposed in [15] trains the policy for controlling the parameters of the RETs using reinforcement learning.

The redirection algorithms discussed so far assume square and obstacle-free areas as physical spaces, but obstacles such as furniture and pillars are present in an actual room. Chen et al. [2] proposed the steer-to-farthest (S2F) algorithm to handle such situations. The S2F algorithm applies RETs to redirect a user toward a safe direction minimized by a predefined cost function. Thomas et al. [16] defined the artificial potential function (APF) [7] that represents the

degree of walkable area in the physical space and redirects a user to walk along the gradient direction of the APF.

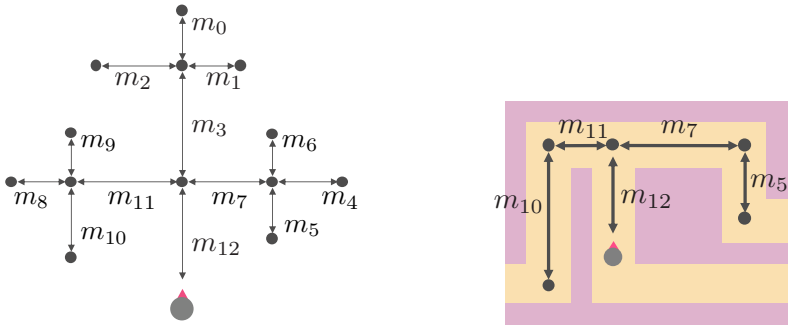
3 Reinforcement Learning-Based Redirection Controller (RLRC)

We train the controller of RETs using reinforcement learning on a simulator. The policy determines the timing and the type of RETs from current observations, e.g., the position and direction of the user in the physical and virtual environments. Reinforcement learning can obtain policies that maximize the expected cumulative reward through trial-and-error by an agent. The state/action spaces and the reward function should be specified to apply a reinforcement learning algorithm. Information on walking paths can help control the RETs efficiently in FORCE [18] and MPCRed [10]. Therefore, we incorporate route information in a virtual environment into the state observation in reinforcement learning.

3.1 Problem Formulation

Here, we describe the design of the state/action spaces and the reward function for reinforcement learning.

State: We use three types of state information: the user's surrounding state in the physical space p, the state in the virtual environment v, and the map state in the virtual environment m. The state variables, p and v, contain distances to the boundaries or obstacles in 36 directions around the user, measured up to 8 meters ahead. The distances are normalized within $[0, 1]$, and the dimension of p and v is 36. We note that normalizing the state values would contribute the stable training. We denote state vectors as $p = (p_0, \ldots, p_{35})^{\mathrm{T}}$ and $v = (v_0, \ldots, v_{35})^{\mathrm{T}}$, where p_i and v_i indicate the distances in the ith directions in the physical and virtual environments, respectively. The map state m represents the shape of the route. The route in the virtual environment can be seen as a graph in which the vertices indicate intersections and corners, and the edges indicate the paths and the distances between them. In this work, we assume that the paths in the virtual environment intersect orthogonally. This type of virtual environment is also used in [9,10]. Under this assumption, we then extract the subgraph consisting of vertices within two hops from the vertex in front of the user's position, where the vertices in the backward direction of the user are ignored. The distances between the vertices are used as the state variables of the map state. In addition, we add the distance to the vertex in front of the user's position. Figure 1 (a) shows an illustration of the subgraph and the corresponding variables. The map state is given by a 13-dimensional vector as $m = (m_0, \ldots, m_{12})^{\mathrm{T}}$. If the path corresponding to m_i does not exist, the value is set to $m_i = 0$. The distance between the vertices is normalized within $[0, 1]$. Figure 1 (b) shows an example of the extracted subgraph. In this case, the



(a) Graphical representation of paths in the virtual environment

(b) Example of a map state

Fig. 1: Map state in the virtual environment

map state vector is given by $m = (0, 0, 0, 0, 0, m_5, 0, m_7, 0, 0, m_{10}, m_{11}, m_{12})^{\mathrm{T}}$. Existing redirection controllers using reinforcement learning [1,9,15] only use the user's spatial information and not use map state information. The overall state vector is given by $s = (p^{\mathrm{T}}, v^{\mathrm{T}}, m^{\mathrm{T}})^{\mathrm{T}} \in \mathbb{R}^{85}$.

Action: We control the RETs directly as the action in reinforcement learning. The following six actions are used in this work: activating the rotational RET to the right (a_1), activating the rotational RET to the left (a_2), deactivating the rotational RET (a_3), activating the reset technique (a_4), deactivating the reset technique (a_5), and do nothing (a_6). In the experiment, we make action decisions every six steps (frames). If both the rotational RET and the reset technique is activated, the reset technique is applied preferentially. The action "do nothing (a_6)" means that no manipulation is applied at that time steps even if the rotational RET or the reset technique is activated.

The rotational RET and the reset technique used in this work are based on the ones used in MPCRed [10]. The rotational RET consists of the rotation and curvature redirections, and modifies the user's rotation by the following additive angle: $\Delta\hat{\phi} = \max(g_C S_{\mathrm{R}}, (1 - g_R) Y_{\mathrm{R}})$, where S_{R} and Y_{R} are the user's real movement angle and yaw rotation per time step, respectively, and g_C and g_R indicate the curvature and rotation gains, respectively. The gain parameters are decided according to [14]. The sign of g_C determines the direction of the redirection, and we set $|g_C| = 0.13$. Regarding the rotational gain, we set $g_R = 0.67$ if the redirection direction and the user's rotation direction are the same; otherwise $g_R = 1.24$. We use 2:1-turn [17] as the reset technique. When the reset technique is applied, the user turns in the physical space, while the view of the virtual environment rotates by twice the physical amount. It means that the rotational gain of the turn is two. For instance, if the user rotates by 180°, the view of the virtual environment is unchanged since the rotation of the viewpoint becomes 360°.

Table 1: Elements of the reward function. The symbol $r_{\min}$ indicates the minimum distance to the boundary or obstacles in the physical space, given by $r_{\min} = \min_i r_i$, and t_{reset} is the number of elapsed steps from the start of applying the reset technique.

Description	Condition	Value	Grant frequency
(1) Position in physical space	—	$6 \times 10^{-2} \times r_{\min}^2$	Every step
(2) Rotational RET usage	If no RET is applied	2×10^{-4}	Every step
(3) Rotational RET switch	If the direction of RET is switched	-1×10^{-3}	Every action
(4) Reset technique usage	If the reset technique is activated	-1.6	Every action
(5) Activating time of reset technique	—	$\frac{-1 \times 10^{-2}}{60} \times t_{\text{reset}}$	Every step
(6) Contacting walls or obstacles	If the user collides with walls or obstacles	-0.9	Every step

Reward function: The reward function employed is the sum of the six elements shown in Table 1. The positive values are added when the user stays away from walls (boundaries) and obstacles in the physical space, or when the rotational RET is not applied. The negative values are added when the direction of the rotational RET is switched, the reset technique is applied, or the user collides with walls and obstacles in the physical space. We strongly penalize the use of the reset technique compared to the rotational RET, because the reset technique is not preferred from the viewpoint of enhancing immersiveness of VR.

3.2 Reinforcement Learning Algorithm

The goal of reinforcement learning is to find a policy that maximizes the expected cumulative reward. We use the proximal policy optimization (PPO) [13] algorithm implemented in the Unity ML-Agents Toolkit (Beta) [5].[1] In the PPO, the policy and value function is represented by a neural network. The neural network used in this work consists of five fully connected hidden layers with 128 units. The activation function of each hidden unit is Swish $f(x) = x \cdot \text{sigmoid}(\beta x)$ [11], which is the default setting of the ML-Agent Toolkit, where $\beta = 1.0$. We use the Adam [8] optimizer to train the neural network. The reinforcement learning algorithm is expected to obtain an optimal policy for RETs that prevents collision with walls or obstacles in the physical space through the training.

[1] `https://github.com/Unity-Technologies/ml-agents`

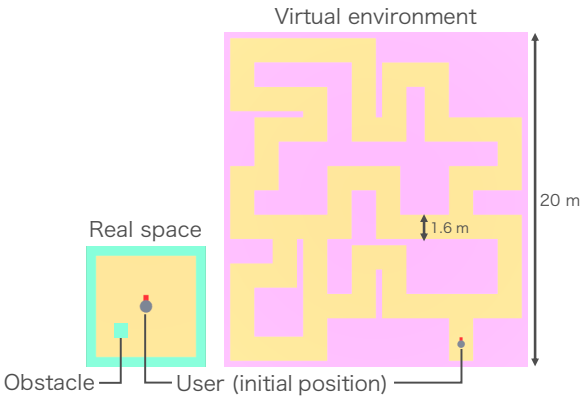


Fig. 2: Physical space and virtual environment on our simulator

3.3 Simulator for Redirected Walking

We implemented a simulator for RDW using Unity.[2] Figure 2 shows the physical space and the virtual environment in the simulator. The virtual environment is the maze used in [10], and has 1.6 meter wide routes in a 20×20 m^2 space. We assume that the shape of the physical space is square. The size of the physical space is $n_\mathrm{p} \times n_\mathrm{p}$ square meter, and the 0.5×0.5 m^2 obstacle can be located at any position. We note that no obstacle is placed if $n_\mathrm{p} \leq 3$ to keep the sufficient walkable area. The redirection controller is trained using this simulator. During the training, the user follows the route as much as possible in the virtual environment. The user will turn along with the route if a corner is reached. At T-junctions, the user turns right or left randomly. The movements of the user are synchronized between the virtual and physical environments. The movement speed of the user is set to 0.34 [m/sec]. The user position is reset to the initial position when 18,000 steps with 60 frames per second (FPS), corresponding to 5 minutes in real time elapse. In addition, the position of an obstacle and the width $n_\mathrm{p} \in \{2, 3, 4, 5, 6\}$ of the physical space are reset randomly.

4 Experimental Evaluation

We evaluate the proposed RLRC with the simulator and user tests. We use the trained redirection controller described in Section 3 for both the simulator and the user tests. We apply the redirection controller to four types of square physical spaces, 10×10 m^2, 4×4 m^2, 4×4 m^2 with an obstacle, and 2×2 m^2, and compare the number of reset techniques per minute and the viewpoint rotation angle per second as the performance measures. The viewpoint rotation angle increases by applying the rotational RET. In both performance measures, smaller values indicate better performance.

[2] `https://unity.com/`

Table 2: Mean and standard deviation of the number of reset techniques per minute [resets/min] in the simulator test. The S2OT results are the values reported in [9].

	10×10 m^2	4×4 m^2	4×4 m^2 + obstacle	2×2 m^2
S2C	0.83 ± 0.14	5.41 ± 0.65	7.61 ± 2.02	14.4 ± 1.60
S2OT [9]	1.15	4.43	—	—
RLRC	**0.56 ± 0.25**	**3.77 ± 0.34**	**4.44 ± 0.63**	**11.9 ± 0.30**

Table 3: Mean and standard deviation of the viewpoint rotation angle per second [degrees/sec] in the simulator test. The S2OT results are the values reported in [9].

	10×10 m^2	4×4 m^2	4×4 m^2 + obstacle	2×2 m^2
S2C	6.98 ± 0.15	**5.47 ± 0.38**	5.98 ± 0.58	**4.65 ± 0.35**
S2OT [9]	9.76	9.52	—	—
RLRC	**5.23 ± 0.36**	6.22 ± 0.36	6.11 ± 0.34	5.78 ± 0.19

4.1 Simulator Test

Experimental Setting We compare the proposed RLRC with S2C and S2OT [9], and follow the experimental setting described in [9] as possible. The physical and virtual environments in the simulator are the same as shown in Figure 2. The user in the simulator acts as explained in Section 3.3. For RLRC and S2C, the user moves for 5 minutes starting from the initial position. When the reset technique is activated, the user rotates by 120 [degree/sec]. We collect the data of 300 trials for each of the four types of physical spaces. The space at least 0.2 meters away from the walls or an obstacle in the physical space is considered as the safe area for S2C. In S2C, when the user enters the unsafe area, the reset technique is applied, and the user rotates 180° in the physical space. For S2OT, we refer to the values reported in [9].

Result and Discussion Tables 2 and 3 summarize the average numbers of reset techniques applied per minute and the average viewpoint rotation angle per second, respectively, for each algorithm in the different physical spaces described. We conducted the Wilcoxon rank sum test between the results of the RLRC and S2C with a significance level $p = 0.01$. If the statistical test shows performance difference, we denote the value of better performance with bold font.

Table 2 shows that the proposed RLRC outperforms S2C and S2OT in all physical space settings. RLRC can reduce the number of reset techniques by 51% and 15% compared with S2OT in the 10×10 m^2 and 4×4 m^2 physical space settings, respectively. This result implies that the design of the state/action spaces, and the reward function of the RLRC are effective; we believe that the map state information we introduced contributes in predicting the middle- or long-term behavior of the user and reducing the number of reset techniques. For instance, consider the situation when the user approaches a corner in the virtual

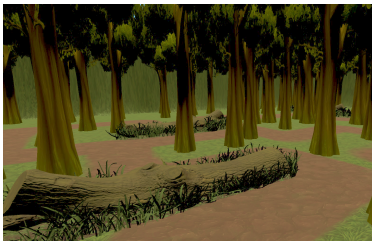


Fig. 3: Scene of the virtual environment redesigned for the user test

environment and the user is located near the wall in the physical space; it is easy to predict the turn direction within a few seconds, and the controller can decide not to apply the reset technique. Comparing RLRC and S2C, RLRC can better reduce the number of reset techniques in the 4×4 m^2 with an obstacle and 2×2 m^2 physical space settings that have not been considered in existing RDW methods using reinforcement learning [1,9,15]. Therefore, we have shown that reinforcement learning-based approaches have the potential to realize efficient RDWs in small physical spaces.

Regarding the result of the viewpoint rotation angle shown in Table 3, RLRC outperforms S2OT in both the 10×10 m^2 and 4×4 m^2 physical space settings, and reduces the angle by approximately 35–45%. However, RLRC performs better only in the 4×4 m^2 physical space setting compared with S2C. We assume that RLRC concentrates in reducing the number of reset techniques rather than the rotation angle. This is because applying the reset technique generates a strong negative reward, and reinforcement learning tends to avoid the reset technique instead of increasing the rotation angle. We note that reducing the number of reset techniques is more desirable than reducing the viewpoint rotation angle to maintain the immersiveness of VR.

4.2 User Test

Experimental Setting We evaluated the trained redirection controller with human users; the users employ RDW with the trained controller and experience the VR. We implement the virtual environment shown in Figure 3, in which the route is the same as the one used in the simulator test. We prepare three conditions of the physical space, 4×4 m^2, 4×4 m^2 with an obstacle, and 2×2 m^2, where the obstacle is located at the relative position $(x, z) = (-0.6, -0.6)$ from the center. We compare RLRC and S2C, resulting in six test patterns (three physical spaces $\times$ two algorithms) being examined. When the reset technique is applied in the user test, a directional arrow is displayed in the HMD, and the user is prompted to rotate in that position. The frame rate is 60 FPS.

We recorded the number of reset techniques and the viewpoint rotation angles for performance comparison. In addition, the participants answered two questionnaires: Kennedy's Simulator Sickness Questionnaire (SSQ) [6] and the NASA Task Load Index (NASA-TLX) [3]. SSQ assesses VR sickness with a

Table 4: Mean and standard deviation of the number of reset techniques per minute [resets/min] in the user test.

	4×4 m^2	4×4 m^2 + obstacle	2×2 m^2
S2C	5.33 ± 1.99	6.58 ± 1.93	12.1 ± 2.32
RLRC	4.30 ± 1.44	8.08 ± 2.34	**8.48 ± 0.84**

Table 5: Mean value standard deviation of the viewpoint rotation angle per second [degrees/sec] in the user test.

	4×4 m^2	4×4 m^2 + obstacle	2×2 m^2
S2C	7.67 ± 1.44	6.68 ± 1.03	6.14 ± 0.71
RLRC	6.44 ± 0.88	6.24 ± 1.01	**4.89 ± 0.37**

4-point Likert scale (0: None–3: Severe), and the total score is calculated by following [6]. NASA-TLX assesses the task load with a 5-point Likert scale (1: None–5: Severe) and evaluates the average scores of six evaluation items: (1) mental demand, (2) physical demand, (3) temporal demand, (4) performance for tasks, (5) effort, and (6) frustration level. The procedure of the user test is as follows:

1. **Consent and pre-questionnaire:** After declaring their assent in a consent form, the participants fill out the pre-SSQ to check their condition before performing tasks.
2. **Perform test:** The participants use an HTC VIVE HMD and then walk freely in the virtual environment shown in Figure 3 for three minutes.
3. **Post-questionnaires and break:** The participants fill out the post-SSQ and NASA-TLX, and then take a break. If a task pattern remains, they continue the test.

Nineteen participants (18 male and 1 female) took part in the user test, with ages between 20 and 25 years. Each participant underwent testing on randomly selected physical space settings. The participants experienced both RLRC and S2C in each selected physical space. The order of the tests was randomized. At least eight data points were collected for each physical space and algorithm. We used a laptop with an Intel Core i7-7820HK CPU, 32 GB RAM, and a GeForce GTX 1070 GPU for the user test, and confirmed that the redirection controllers work on this laptop in real-time.

Result and Discussion Tables 4 and 5 summarize the number of reset techniques per minute and the viewpoint rotation angle per second in the user test. We conducted the Wilcoxon rank sum test ($p = 0.01$) between the results of RLRC and S2C as well as the simulator test.

It can be seen from Table 4 that RLRC outperforms S2C in the 2×2 m^2 physical space setting in terms of the number of reset techniques. For other physical space settings, there is no statistical significance between RLRC and

Table 6: Total SSQ scores calculated by following [6]

	4×4 m^2	4×4 m^2 + obstacle	2×2 m^2
pre-SSQ	8.88 ± 6.30	7.90 ± 8.46	5.40 ± 5.95
S2C	36.2 ± 66.1	47.2 ± 53.6	19.5 ± 13.1
RLRC	25.3 ± 26.7	28.5 ± 19.4.7	32.4 ± 31.7

S2C. The performance improvement of RLRC against S2C is not highlighted as in the simulator test. One reason is the discrepancy of the movements between the simulated user and actual participants. The simulated user always moves mechanically, while human participants naturally make motions like swaying or looking around even when going straight. Since such non-mechanical movements do not appear in the training phase using the simulator, reinforcement learning may not have learned an optimal policy for the user test.

Regarding the viewpoint rotation angle, a similar trend as the number of reset techniques is observed. RLRC outperforms S2C in the 2×2 m^2 physical space setting. We also observed a tendency of the viewpoint rotation angle to decrease when the number of reset techniques increases. The S2C algorithm applies the rotational RET when the user does not face the center of the physical space. Since human users often change direction even when going straight, the total duration of applying the redirection increases. We assume that RLRC is more robust than S2C in such a situation.

From the user test, we can verify that the trained redirection controller in the simulator is valid for real-life situations, and works well in small physical spaces. Finally, we provide the SSQ results in Table 6. We do not observe any significant differences between RLRC and S2C in the total SSQ score and all six items of the NASA-TLX. The NASA-TLX results have been omitted for conciseness.

5 Conclusions

In this paper, we focused on RDW in virtual maze environments, and developed a reinforcement learning-based redirection controller. The proposed RLRC leverages the map state information and trains the controller using PPO. We conducted simulator and user tests, and compared the performance with existing methods. In the simulator test, four types of square physical spaces were examined, including physical spaces of 4×4 m^2 with an obstacle, and of 2×2 m^2. The experimental results of the simulator test suggest that RLRC is superior to S2C, particularly in terms of the number of reset techniques. In addition, we confirmed that the proposed RLRC can realize RDW in the user test, despite using the trained controller on the simulator without additional tuning. In the user test, RLRC can reduce the number of reset techniques and the viewpoint rotation angles in the 2×2 m^2 physical space better compared to S2C. Our experimental evaluation implies that the reinforcement learning-based approach has the potential to realize an efficient RDW in small physical spaces.

To improve the efficiency of RDW in the user test, a possible future work is to make the behavior of the simulated user in the training phase more realistic, e.g., adding noise to the state and action, and using real paths collected from actual user behavior. Furthermore, it might be possible to construct a universal redirection controller by using various virtual environments during the training.

References

1. Chang, Y., Matsumoto, K., Narumi, T., Tanikawa, T., Hirose, M.: Redirection controller using reinforcement learning. arXiv preprint:1909.09505 (2019)
2. Chen, H., Chen, S., Rosenberg, E.S.: Redirected walking in irregularly shaped physical environments with dynamic obstacles. In: 2018 IEEE Conference on Virtual Reality and 3D User Interfaces (VR). pp. 523–524 (2018)
3. Hart, S.G., Staveland, L.E.: Development of NASA-TLX (task load index): Results of empirical and theoretical research. In: Advances in Psychology, vol. 52, pp. 139–183 (1988)
4. Hasselt, H.V., Guez, A., Silver, D.: Deep reinforcement learning with double Q-learning. In: Thirtieth AAAI Conference on Artificial Intelligence (2016)
5. Juliani, A., Berges, V.P., Vckay, E., Gao, Y., Henry, H., Mattar, M., Lange, D.: Unity: A general platform for intelligent agents. arXiv preprint:1809.02627 (2018)
6. Kennedy, R.S., Lane, N.E., Berbaum, K.S., Lilienthal, M.G.: Simulator sickness questionnaire: An enhanced method for quantifying simulator sickness. The International Journal of Aviation Psychology **3**(3), 203–220 (1993)
7. Khatib, O.: Real-time obstacle avoidance for manipulators and mobile robots. In: 1985 IEEE International Conference on Robotics and Automation. vol. 2, pp. 500–505 (1985)
8. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. In: International Conference on Learning Representations (ICLR) (2015)
9. Lee, D., Cho, Y., Lee, I.: Real-time optimal planning for redirected walking using deep Q-learning. In: 2019 IEEE Conference on Virtual Reality and 3D User Interfaces (VR). pp. 63–71 (2019)
10. Nescher, T., Huang, Y.Y., Kunz, A.: Planning redirection techniques for optimal free walking experience using model predictive control. In: 2014 IEEE Symposium on 3D User Interfaces (3DUI). pp. 111–118 (2014)
11. Ramachandran, P., Zoph, B., Le, Q.V.: Swish: a self-gated activation function. arXiv preprint:1710.05941 (2017)
12. Razzaque, S., Kohn, Z., Whitton, M.C.: Redirected walking. In: Eurographics 2001 - Short Presentations. Eurographics Association (2001)
13. Schulman, J., Wolski, F., Dhariwal, P., Radford, A., Klimov, O.: Proximal policy optimization algorithms. arXiv preprint:1707.06347 (2017)
14. Steinicke, F., Bruder, G., Jerald, J., Frenz, H., Lappe, M.: Estimation of detection thresholds for redirected walking techniques. IEEE Transactions on Visualization and Computer Graphics **16**(1), 17–27 (2010)
15. Strauss, R.R., Ramanujan, R., Becker, A., Peck, T.C.: A steering algorithm for redirected walking using reinforcement learning. IEEE Transactions on Visualization and Computer Graphics **26**(5), 1955–1963 (2020)
16. Thomas, J., Rosenberg, E.S.: 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)

17. Williams, B., Narasimham, G., Rump, B., McNamara, T.P., Carr, T.H., Rieser, J., Bodenheimer, B.: Exploring large virtual environments with an hmd when physical space is limited. In: 4th Symposium on Applied Perception in Graphics and Visualization (APGV '07). pp. 41–48 (2007)
18. Zmuda, M.A., Wonser, J.L., Bachmann, E.R., Hodgsons, E.: Optimizing constrained-environment redirected walking instructions using search techniques. IEEE Transactions on Visualization and Computer Graphics **19**(11), 1872–1884 (2013)