Mobile Robot Navigation Using Learning-Based Method Based on Predictive State Representation in a Dynamic Environment

Kohei Matsumoto[1], Akihiro Kawamura[2], Qi An[2], and Ryo Kurazume[2]

***Abstract*— Mobile robot navigation in a dynamic environment with pedestrians is essential for service robots operating in a living environment. Accordingly, the robot needs to understand and predict the behavior of pedestrians. However, predicting pedestrian behavior in advance is difficult because human behavior may be affected by factors that cannot be directly observed or modeled in advance, such as intentions and environmental influences. In addition, pedestrian behavior may be affected by the behavior of the robot.**

In this study, we apply a deep reinforcement learning method based on a novel predictive state representation (PSR) model to mobile robot navigation for realizing a navigation method considering the changes in pedestrian behavior caused by robot actions and other pedestrians. In addition, we propose two methods for integrating the states of the PSRs corresponding to each pedestrian and evaluate these methods in situations where the number of pedestrians differs between learning and testing.

I. Introduction

Mobile robot navigation in dynamic environments is an important element for service robots that provide services in spaces inhabited by humans. Handling this task requires the robots to understand the behavior of pedestrians. However, pedestrian behavior may change depending on their intentions and other factors that depend on particular environments. Thus, it cannot be easily modeled in advance with sufficient accuracy, particularly if the road shape and attractions in the environment (*e.g.,* stores or scenery) are considered. Furthermore, the behavior of the robot may affect the pedestrian behavior.

To address these issues, we propose a learning-based method based on predictive state representation (PSR) that considers the changes in pedestrian behavior caused by robot actions. This method can learn the behavior of pedestrians by observing their interactions with the environment and the changes in their behavior due to the robot's actions. Mobile robot navigation methods based on deep reinforcement learning (RL) have been actively studied in recent years [1]–[5]. However, the methods proposed so far mainly use predesigned models or do not specifically consider the changes in pedestrian behavior caused by robot actions.

In this study, we apply a deep RL method based on the new PSR model to mobile robot navigation for considering the changes in pedestrian behavior caused by robot actions and other pedestrians. In addition, we consider the situation where the number of pedestrians changes between training and testing. The main contributions of this study are as follows:

- the application of a deep RL method using PSR for mobile robot navigation and the verification of the effects of the structure based on PSR for mobile robot navigation tasks;
- proposing a new PSR architecture for mobile robot navigation in a dynamic environment with pedestrians considering the interactions among pedestrians and the verification of the effectiveness of the architecture;
- proposing methods for integrating the states of the PSRs corresponding to each pedestrian for handling the change in the number of pedestrians between training and testing, and comparing these integrating methods.

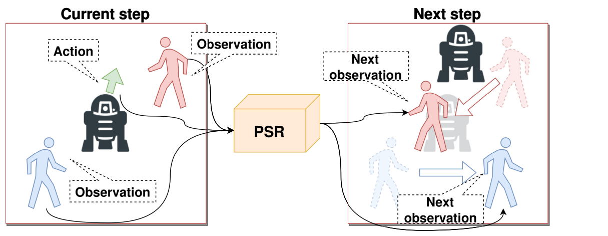


Fig. 1: PSR modeling for mobile robot navigation in a dynamic environment with pedestrians

II. Related works

Various methods based on deep RL techniques have been proposed. Everett *et al.* [1] proposed a method based on policy-based RL and long short-term memory. Chen *et al.* [2] proposed a pooling module for learning the relationships between the attention mechanism, changes in human behavior, and surrounding pedestrians; they also proposed a method to learn robot behavior, including human–robot and human–human interactions. Chen *et al.* [3] proposed an excellent navigation method that uses a graph convolutional network and attention mechanism based on human gaze. Chen *et al.* [4] used model-based deep RL methods and a graph convolutional network to achieve efficient navigation by encoding the interactions among agents. Liu [5] proposed a deep RL method that focuses on robot navigation in crowded real-world environments.

PSR [6] is used for modeling dynamics by considering partial observability. Various methods for utilizing PSR have been proposed. Boots *et al.* [7] proposed an extended PSR model based on the Hilbert space embeddings of distributions to manage infinite sets of continuous observations and

[1]Kohei Matsumoto is with the Graduate School of Information Science and Electrical Engineering, Kyushu University, Fukuoka 819-0395, Japan `matsumoto@irvs.ait.kyushu-u.ac.jp`

[2]Akihiro Kawamura, Qi An, and Ryo Kurazume are with the Faculty of Information Science and Electrical Engineering, Kyushu University, Fukuoka 819-0395, Japan {`kawamura, anqi, kurazume`} `@ait.kyushu-u.ac.jp`

actions. Hefny *et al.* [8] proposed a supervised learning method based on the kernel version of instrumental variable regression for PSR models. They also proposed an efficient PSR model for learning controlled dynamical systems using random Fourier features with a kernel function [9]. In addition, methods utilizing PSR for deep learning have been proposed recently. Venkatraman *et al.* [10] studied the use of PSR for representing the latent states of recurrent neural networks (RNNs). They observed that the proposed method could improve the performance and compensate for the disadvantages of RNN in probabilistic filtering, imitation learning, and RL. Hefny *et al.* [11] proposed a fundamental deep RL method, called the recurrent predictive state policy network, to utilize PSR.

In robotics, PSR has been applied in human–robot communication by utilizing its ability to model the results after performing actions [12]. It has also been applied in in-hand manipulation, which includes interactions with the environment [13]. This method extends PSR to address partial observability using a new kernel-based feature that integrates actions and observations. However, PSR has not yet been applied to mobile robot navigation in dynamic environments.

III. Preliminaries

A. Predictive state representation

In PSR [6], the dynamics are modeled considering the interactions between an agent and the environment. The characteristic of PSR is that the state is expressed in a way that facilitates the prediction of the future. The state is represented by observations and actions. Therefore, values other than the fully observable elements do not need to be defined. In this section, we describe PSR in detail.

1) Original PSR: The basic concept of PSR is that, if the expected results of all possible tests are known, the dynamical system becomes completely understood. By expressing the state using observable information, partially observable dynamic systems can be modeled without prior knowledge. Consider a discrete system with a finite set of observations, $\mathcal{O} = \{o_1, o_2, \ldots, o_k\}$, and actions, $\mathcal{A} = \{a_1, a_2, \ldots, a_k\}$. The state representation of the system at time t is a vector composed of the probability of occurrence of a test based on the latest history. Each test is a sequence of actions and observations starting at time $t+1$; the history at time t is a sequence of actions and observations up to and including time t. The probability of success of a test, τ, of length m for history h, i.e., the probability of obtaining an observation sequence in τ when taking the sequence of actions in τ, is represented by $p(\tau \mid h) = p(h,\tau)/p(h) = \prod_{i=1}^{m} \Pr(o_i \mid h, a_i)$.

Knowing the success probabilities of some tests may help us understand those of other tests. Given a test set $\mathcal{T} = \{\tau_1, \tau_2, \ldots, \tau_k\}$, if there exists a function f_t whose prediction vector $p(\mathcal{T} \mid h) = [p(\tau_1 \mid h)\, p(\tau_2 \mid h) \ldots p(\tau_k \mid h)]$ satisfies $p(\tau_l \mid h) = f_t(p(\mathcal{T} \mid h))$ for any test τ_l, then $\mathcal{T}$ is called the *core test* and the prediction vector, $p(\mathcal{T} \mid h)$, represents the state of PSR.

2) Recurrent PSR: The original PSR model can only be applied to systems consisting of discrete observations and actions. Methods to make PSR compatible with continuous systems have been proposed [7], [9]. Herein, these methods are collectively referred to as recurrent PSR (RPSR). In RPSR, the state q_t is represented as a conditional distribution of future observations, $o_{t:t+k-1}$, conditioned by future actions, $a_{t:t+k-1}$. The state update of RPSR is performed in two steps.

- Extension: The linear map W_{ext} is applied to the state q_t to obtain the extended state, e_t. The extended state, e_t, is a conditional distribution of the extended observations, $o_{t:t+k}$, conditioned by the extended action, $a_{t:t+k}$:
$$e_t = W_{\text{ext}} q_t. \tag{1}$$
- Conditioning: For the action a_t and the observation o_t at time t, the known conditioning function, f_{cnd}, updates the state as follows:
$$q_{t+1} = f_{\text{cnd}}\left(e_t, a_t, o_t\right). \tag{2}$$

In discrete systems, q_t and e_t are represented by conditional establishment tables, and a Bayesian rule is applied by f_{cond}. To apply these to a continuous system, the Hilbert space embeddings of the distributions [7] and the kernel Bayes' rule [14] are used.

IV. Approach

A. Application of PSR to mobile robot navigation in a dynamic environment with pedestrians

In this study, we applied a PSR-based deep RL method to mobile robot navigation in a dynamic environment with multiple pedestrians. The elements of mobile robot navigation and PSR can be associated as follows:

- Observations: The position and velocity data of N pedestrians and the robot are treated as observations. The observation data for each pedestrian and robot include vector $(p_x^i, p_y^i, v_x^i, v_y^i)$, where (p_x^i, p_y^i) represents the position and (v_x^i, v_y^i) represents the velocity of the ith pedestrian or robot.
- Actions: In this study, we assumed a holonomic omnidirectional mobile robot and used a two-dimensional vector (v_x, v_y) consisting of the input velocity v_x in the x-axis direction and the input velocity v_y in the y-axis direction of the robot in a two-dimensional space.

The PSR used in this study becomes a model that can predict the position of the pedestrian and the velocity in the next step by inputting the command velocity of the robot. The conceptual diagram of the PSR model is shown in Fig. 1.

B. Architecture of proposed method

The architecture of the proposed method is shown in Fig. 2. The proposed method consists of a PSR consisting of a feature extractor, state updater, and observation predictor, as well as state integrator and value estimator.

- Feature extractor: It extracts features from each input datum using a nonstationary spectral kernel function [15], [16].

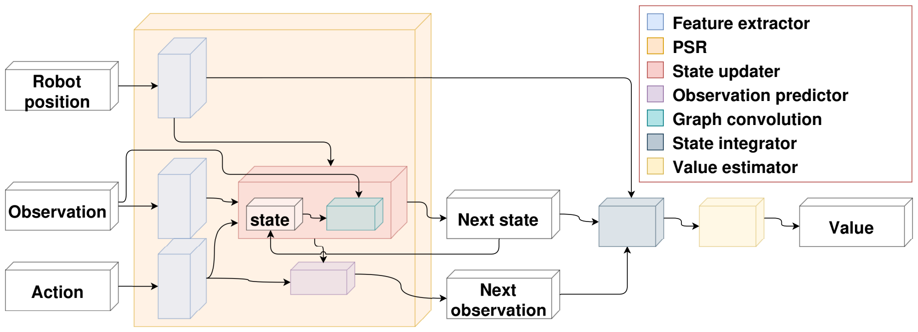


Fig. 2: Architecture of the proposed method

- State updater: It updates states via the state updater using the features extracted from the observation and action.
- Observation predictor: It predicts the next observations using the states features extracted from action.
- State integrator: It integrates the states of the PSR corresponding to each pedestrian and provides the integrated state to the value estimator.
- Value estimator: It estimates the value from the integrated state and robot feature.

C. Graph convolutional RPSR

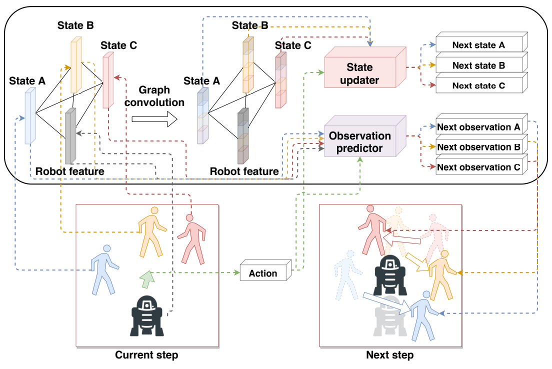


Fig. 3: Procedure of GC-RPSR

The basic RPSR model can consider the effect of the action of the agent on the environment. Through this feature, the effects of robot actions on each pedestrian can be considered. However, the RPSR cannot consider the effects of the interactions among pedestrians. For handling this problem, we propose graph convolutional RPSR (GC-RPSR).

This architecture has a graph convolution part in the state update phase in the RPSR for describing the relation effects among pedestrians. The state update process is shown in Fig. 3. The kernel function for the adjacency matrix A_t is based on the inverse of the L_2 norm of the difference of each position [17]. This function is expressed in Eq. (3).

$$a_t^{ij} = \begin{cases} 1/\left\|p_t^i - p_t^j\right\|_2 & \text{if } \left\|p_t^i - p_t^j\right\|_2 \neq 0 \\ 0 & \text{Otherwise} \end{cases} \tag{3}$$

Finally, we use the normalized adjacency matrix. The normalization process is expressed in Eq. (4).

$$\tilde{A}_t = \Lambda_t^{-\frac{1}{2}} \hat{A}_t \Lambda_t^{-\frac{1}{2}}, \tag{4}$$

where $\hat{A}_t = A_t + I$ and Λ_t is the diagonal node degree matrix of A_t.

By using the graph convol05tion and Eqs. (1) and (2), the state update process of GC-RPSR is expressed as follows:

$$\tilde{q}_t = f_{gc}\left(q_t, \tilde{A}_t\right) \tag{5}$$

$$e_t = W_{\text{ext}} \tilde{q}_t \tag{6}$$

$$q_{t+1} = f_{\text{cnd}}\left(e_t, a_t, o_t\right), \tag{7}$$

where f_{gc} represents the function for graph convolution.

D. State integrator

We need an integration process to construct input data for the value estimator from the state data corresponding to each pedestrian. We propose two methods for the integration. The first method uses graph convolution and the second method is based on an occupancy map.

1) Integration using graph convolution: This method uses graph convolution for integrating the states of the PSRs corresponding to each pedestrian. The value estimator only receives the robot feature to which graph convolution is applied. The kernel function to be used for the adjacency matrix is the same as the state update of the GC-RPSR expressed in Eq. (3). The conceptual diagram of this method is shown in Fig. 4.

2) Integration using occupancy map: This method uses an occupancy map for integrating the states of the PSRs corresponding to each pedestrian. The states of the pedestrians are stored in the corresponding cell depending on the position of each pedestrian. If multiple states are required to be stored in the same cell, an average value is stored. The conceptual diagram of this method is shown in Fig. 5.

E. Action generation

The action is generated using the value estimator f_v, state updater f_p^q, and observation predictor f_p^o from the PSR model f_p by selecting an action obtaining the maximum reward

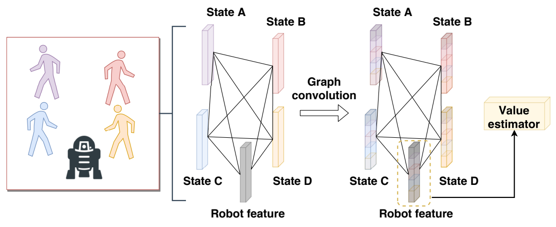
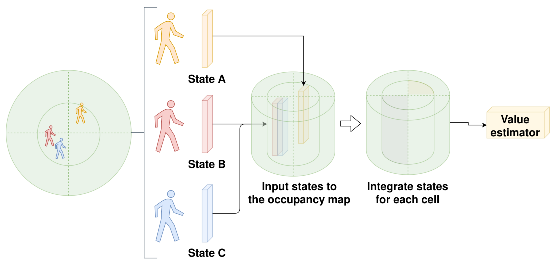


Fig. 4: State integration using graph convolution


State A
State B
State C
Input states to the occupancy map
Integrate states for each cell
Value estimator
Fig. 5: State integration using an occupancy map

from the action space $\mathcal{A}$. The formulation is shown in Eq. (8), where γ is the discount factor, s_t is the integrated state at time t, p_t is the robot position at time t, and $\hat{o}_t = f_p^o(q_t, a_t)$.

$$a_t \leftarrow \mathrm{argmax}_{a_t \in \mathcal{A}} R(\hat{o}_t) + \gamma^{\Delta t} f_v(s_t, p_t) \tag{8}$$

In addition, $R(o_t)$ is a reward function at time t. This function is expressed in Eq. (9), where d_t is the minimum separation distance between the robot and the pedestrians and p_g is the goal position of the navigation task.

$$R(o_t) = \begin{cases} -0.25 & \text{if } d_t < 0 \\ -0.1 + d_t/2 & \text{else if } d_t < 0.2 \\ 1 & \text{else if } p_t = p_g \\ 0 & \text{otherwise} \end{cases} \tag{9}$$

Accordingly, we use d-step planning [18] that considers a d-step future. By rollout using the PSR and value estimator d-step, the action is selected with the maximum return along the d-step prediction using Eq. (10).

$$f_v^d(s_t, p_t) = \begin{cases} f_v(s_t, p_t) & \text{if } d = 1 \\ \frac{1}{d} f_v^1(s_t, p_t) + \frac{d-1}{d} \max_{a_t}(\\ R(\hat{o}_{t+1}) + \gamma f_v^{d-1}(\hat{s}_{t+1}, \hat{p}_{t+1})) & \text{otherwise} \end{cases} \tag{10}$$

where $\hat{o}_{t+1} = f_p^o(q_{t+1}, a_{t+1})$, and $\hat{q}_{t+1} = f_p^q(q_t, a_t, o_t)$.

F. Training process

Training is performed in two steps. The first step is an initialization of the PSR model and the imitation learning of the value estimator. Subsequently, we train the value estimator using RL and the PSR model using supervised learning.

1) Preliminary learning step: In the first step, the initial values of the PSR parameters are determined. This initialization is performed through two-stage regression [8] after collecting data using an exploration policy that follows ORCA [19] for initialization. Subsequently, the value estimator is trained in imitation learning using the collected data.

2) RL and supervised learning step: In the second step, we train the entire model with Algorithm 1, where E is the number of episodes for training and d is the target update frequency. The value estimator is trained with the RL procedure based on the bootstrapped DQN [20].

Algorithm 1: Optimization of the proposed method

```
Initialize the PSR $f_p$ and the value estimator $f_v$ with
  two-stage regression and imitation learning
Initialize the target value estimator $\hat{f}_v$
for $i = 1$ to $E$ do
    Select $a_t$ following the exploration policy and obtain
      the reward $r_t$, observation $o_t$, and robot position $p_t$
    After finishing an episode, store the trajectory of
      $(o_t, a_t, r_t, p_t)$ to buffer $\mathcal{B}$

    Sample from the buffer $\mathcal{B}$ and obtain $M$ set of
      trajectories
    for $j = 1$ to $M$ do
        Obtain the trajectories below,
            observation $\mathbf{o}^j = \{o_1^j, o_2^j, \ldots, o_T^j\}$,
            robot position $\mathbf{p}^j = \{p_1^j, p_2^j, \ldots, p_T^j\}$,
            action $\mathbf{a}^j = \{a_1^j, a_2^j, \ldots, a_T^j\}$
        Calculate the trajectories below,
            state $\mathbf{q}^j = \{q_1^j, q_2^j, \ldots, q_T^j\}$,
            target value $\mathbf{y}^j = \{y_1^j, y_2^j, \ldots, y_T^j\}$,
            integrated state $\mathbf{s}^j = \{s_1^j, s_2^j, \ldots, s_T^j\}$
    end
    Update $f_p$ by minimizing $L_{\text{pred}} = \mathrm{MSE}(f_p^o(\mathbf{q}, \mathbf{a}), \mathbf{o})$
    Update $f_v$ by minimizing $L_{\text{value}} = \mathrm{MSE}(f_v(\mathbf{s}, \mathbf{p}), \mathbf{y})$
    if $i$ mod $d$ then
        Update $\hat{f}_v$ by $\hat{f}_v \leftarrow f_v$
    end
end
```

V. Experiment

A. Implementation details

The structure of the value estimator is a multilayer perceptron with a hidden size of {500, 300}. In the RPSR structure, the feature extractor uses 500 random Fourier features and sets the PCA dimensionality reduction to 40 dimensions. The parameters are trained using AdaBelief [21] in the imitation, supervised, and reinforcement learning phases, and the learning rate is 10^{-4}. We trained our models in 10k scenarios by setting the batch size of the trajectory to 5 and the target update frequency to 1k. The discount factor γ is set to be 0.9. The ε-greedy policy is used for the exploration with decays from 0.5 to 0.1 linearly in the first 4k episodes. The action space $\mathcal{A}$ consists of 80 discrete actions, consisting of 5 velocities exponentially spaced between (0, 1] and 16 headings evenly spaced between [0, 2π). Regarding action generation, we set the depth and width of d-step planning to 2. For the state integration of the second model, we use an occupancy map with a radius of 3 m, a distance resolution of 0.3 m, and an angular resolution of 30°.

B. Experimental environment

We use the circle crossing scenarios in the CrowdNav environment used in some related works [2], [4]. In this environment, pedestrians are controlled by ORCA [19]. The pedestrians are randomly placed on a circle of radius 4m, and their position (x, y) is subjected to a random perturbation. The robot and pedestrians are influenced by each other, and their behavior changes with position and velocity. The evaluation is conducted with 500 random test cases.

In the training phases of all the experiments and the testing phases of the first and second experiments, we set the number of pedestrians N to 5.

C. Comparison of GC-RPSR and RPSR

Here, we confirm the effectiveness of GC-RPSR by comparing it with the basic RPSR. We compare the success rate, collision rate, execution time, and average return. We use a simple concatenate for the state integrator. The result is shown in Table I.

TABLE I: Numerical comparison of RPSR and GC-RPSR

Method	Success [%]	Collision [%]	Exec. time [s]	Avg. return
RPSR	57.8	33.8	8.84	0.258
GC-RPSR	64.0	11.0	8.73	0.356

Thus, GC-RPSR outperforms RPSR in terms of all the evaluation items. Accordingly, the proposed PSR model enhances the performance of the mobile robot navigation task compared with basic RPSR.

D. Comparison models with baseline

Here, we compare the proposed models with a method proposed in a related study [4], which is considered as the baseline. The evaluation items are the same as earlier. Fig. 6 shows samples of the result trajectory of the proposed model with state integration using graph convolution (GGC-RPSR) and Fig. 7 shows samples of the result trajectory of the proposed model with state integration using an occupancy map (OGC-RPSR). In each figure, the black trajectory represents the robot, the colored trajectories represent pedestrians, and the numbers indicate time steps. Both methods can navigate the robot to the goal while avoiding pedestrians. However, GGC-RPSR has a higher efficiency because it can generate shorter paths. The numerical comparison of the two proposed methods and the baseline method is shown in Table II.

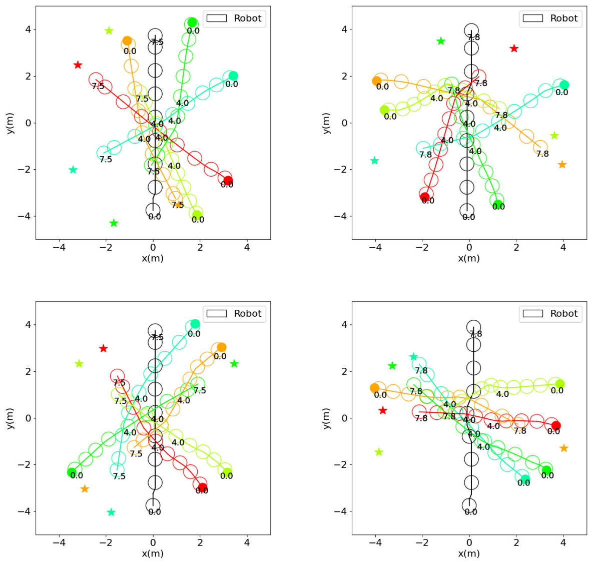


Fig. 6: Sample trajectories of the results of the GGC-RPSR. The black trajectories describe the robot, and the colored trajectories describe pedestrians.

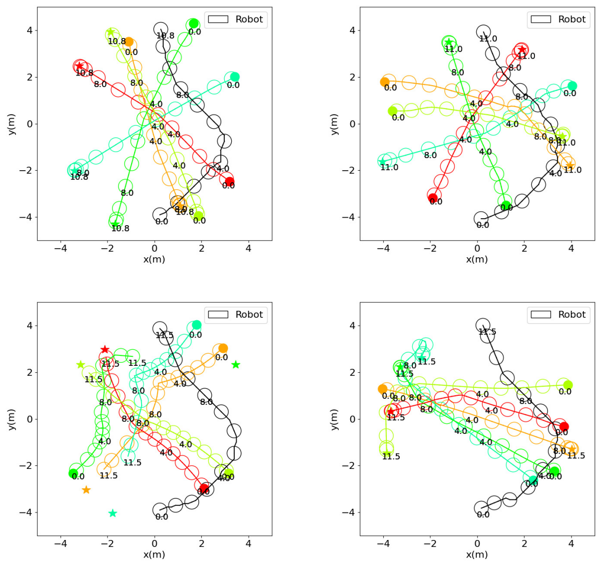


Fig. 7: Sample trajectories of the results of the OGC-RPSR. The black trajectories describe the robot, and the colored trajectories describe pedestrians.

TABLE II: Numerical comparison of the proposed methods and the baseline method

Method	Success [%]	Collision [%]	Exec. time [s]	Avg. return
RGL [4]	92.0	7.0	9.09	0.560
GGC-RPSR	94.6	5.4	7.788	0.587
OGC-RPSR	91.4	6.8	11.536	0.512

The GGC-RPSR outperforms the other methods in terms of all the evaluation items. This result demonstrates the effectiveness of the proposed method using GC-RPSR and state integration using graph convolution. The GGC-RPSR significantly outperforms the OGC-RPSR in terms of the execution time.

E. Comparison of the models in the situation where the number of pedestrians differs between training and testing

Here, we compared the two proposed methods in the situation where the number of pedestrians differs between training and testing. Both the models were trained using an environment in which the number of pedestrians was 5 and were tested in environments in which the number of pedestrians was 1 to 10.

The results of the two models are shown in Table III, where N indicates the number of pedestrians. Thus, the performance difference between the two methods can be observed from the success rate by comparing with the result for $N = 5$. The maximum decline of the success rate of the GGC-RPSR is 43.4%; in contrast, the corresponding result for the OGC-RPSR is 7.4%. This result shows that the OGC-RPSR is more stable when the number of pedestrians is varied between training and testing.

VI. Conclusion and Future Work

This study applied a deep RL method based on PSR to mobile robot navigation tasks in a dynamic environment with pedestrians. We proposed a new PSR architecture for mobile robot navigation in a dynamic environment with pedestrians considering the interactions among pedestrians. In addition,

TABLE III: Results of the GGC-RPSR and OGC-RPSR while changing the number of pedestrians. The values in the brackets in the success rate column show the difference from the result in the situation with $N = 5$.

	GGC-RPSR				OGC-RPSR			
N	Success [%]	Collision [%]	Exec. time [s]	Avg. return	Success [%]	Collision [%]	Exec. time [s]	Avg. return
1	63.6 (− 31.0)	2.4	8.171	0.419	98.2 (+ 6.8)	0.6	10.983	0.587
2	68.2 (− 26.4)	16.4	8.109	0.393	95.0 (+ 3.6)	3.4	11.017	0.560
3	65.8 (− 28.8)	22.0	8.122	0.361	94.4 (+ 3.0)	4.8	11.126	0.548
4	91.0 (− 3.6)	5.8	7.988	0.556	92.0 (+ 0.6)	6.8	11.048	0.527
5	94.6 (± 0.0)	5.4	7.788	0.587	91.4 (± 0.0)	6.8	11.536	0.512
6	76.2 (− 18.4)	20.8	7.985	0.431	88.0 (− 3.4)	10.0	11.251	0.485
7	93.6 (− 1.0)	5.8	7.889	0.578	91.2 (− 0.2)	7.6	11.327	0.502
8	89.0 (− 5.6)	10.2	7.885	0.541	91.2 (− 0.2)	7.6	11.270	0.500
9	51.2 (− 43.4)	39.6	8.135	0.220	84.0 (− 7.4)	13.0	11.505	0.422
10	89.0 (− 5.6)	10.8	7.878	0.540	85.8 (− 5.6)	11.8	11.714	0.447

we proposed methods for integrating the state of the PSRs corresponding to each pedestrian for handling the change in the number of pedestrians between training and testing.

The experiments conducted showed the effectiveness of the proposed model. In addition, we confirmed that the second version of the proposed model, which uses an occupancy-map-based state integration, is more stable in the situation where the number of pedestrians differs between training and testing.

In future work, the two types of state integration methods proposed in this paper have not yet been fully discussed; further experiments are needed to clarify in detail the causes of the differences between the two methods. For the occupancy map-based method, further experiments are needed to clarify the relationship between parameters such as the number of grids and the performance change. In addition, we will consider the more effective model and the learning process for a real-world scenario. The learning costs and exploration processes are critical problems in real-world deep RL tasks. If a robot can learn in the real world directly, the trained system can perform better in actual environments than in simulated environments. However, the robot would be required to repeat its interactions with the environment and learn from numerous success and failure patterns, which is difficult to achieve practically. Thus, a more efficient learning process is required to overcome these problems.

ACKNOWLEDGMENT

This work was partially supported by JSPS KAKENHI Grant Number JP20H00230.

REFERENCES

[1] M. Everett, Y. F. Chen, and J. P. How, "Motion Planning among Dynamic, Decision-Making Agents with Deep Reinforcement Learning," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 3052–3059, 2018.

[2] C. Chen, Y. Liu, S. Kreiss, and A. Alahi, "Crowd-robot interaction: Crowd-aware robot navigation with attention-based deep reinforcement learning," in *Proceedings - IEEE International Conference on Robotics and Automation*, vol. 2019-May, pp. 6015–6022, 2019.

[3] Y. Chen, C. Liu, B. E. Shi, and M. Liu, "Robot Navigation in Crowds by Graph Convolutional Networks with Attention Learned from Human Gaze," *IEEE Robotics and Automation Letters*, vol. 5, pp. 2754–2761, apr 2020.

[4] C. Chen, S. Hu, P. Nikdel, G. Mori, and M. Savva, "Relational graph learning for crowd navigation," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2020.

[5] L. Lucia, D. Daniel, C. Gianluca, S. Roland, and D. Renaud, "Robot Navigation in Crowded Environments Using Deep Reinforcement Learning," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2020.

[6] S. Singh, M. James, and M. Rudary, "Predictive State Representations: A New Theory for Modeling Dynamical Systems," in *20th The Conference on Uncertainty in Artificial Intelligence (UAI)*, 2004.

[7] B. Boots, A. Gretton, and G. J. Gordon, "Hilbert space embeddings of predictive state representations," in *29th Conference on Uncertainty in Artificial Intelligence (UAI)*, pp. 92–101, 2013.

[8] A. Hefny, C. Downey, and G. J. Gordon, "Supervised learning for dynamical system learning," in *Advances in Neural Information Processing Systems*, vol. 2015-Janua, pp. 1963–1971, 2015.

[9] A. Hefny, C. Downey, and G. Gordon, "An efficient, expressive and local minima-free method for learning controlled dynamical systems," in *32nd AAAI Conference on Artificial Intelligence (AAAI)*, pp. 3191–3198, feb 2018.

[10] A. Venkatraman, N. Rhinehart, W. Sun, L. Pinto, M. Hebert, B. Boots, K. M. Kitani, and J. A. Bagnell, "Predictive-state decoders: Encoding the future into recurrent networks," in *Advances in Neural Information Processing Systems*, vol. 2017-Decem, pp. 1173–1184, Neural information processing systems foundation, 2017.

[11] A. Hefny, Z. Marinho, W. Sun, S. S. Srinivasa, and G. Gordon, "Recurrent predictive state policy networks," in *35th International Conference on Machine Learning, ICML 2018*, vol. 5, pp. 3104–3119, 2018.

[12] E. Meisner, S. Das, V. Isler, J. Trinkle, S. Śabanović, and L. R. Caporael, "Predictive state representations for grounding human-robot communication," in *Proceedings - IEEE International Conference on Robotics and Automation*, pp. 178–185, 2010.

[13] J. A. Stork, C. H. Ek, Y. Bekiroglu, and D. Kragic, "Learning Predictive State Representation for in-hand manipulation," in *Proceedings - IEEE International Conference on Robotics and Automation*, vol. 2015-June, pp. 3207–3214, Institute of Electrical and Electronics Engineers Inc., jun 2015.

[14] K. Fukumizu, L. Song, and A. Gretton, "Kernel Bayes' rule: Bayesian inference with positive definite kernels," *Journal of Machine Learning Research*, vol. 14, pp. 3753–3783, 2013.

[15] S. Remes, M. Heinonen, and S. Kaski, "Non-stationary spectral kernels," in *Advances in Neural Information Processing Systems*, vol. 2017-Decem, pp. 4643–4652, 2017.

[16] J. F. Ton, S. Flaxman, D. Sejdinovic, and S. Bhatt, "Spatial mapping with Gaussian processes and nonstationary Fourier features," *Spatial Statistics*, vol. 28, pp. 59–78, dec 2018.

[17] A. Mohamed, K. Qian, M. Elhoseiny, and C. Claudel, "Social-STGCNN: A Social Spatio-Temporal Graph Convolutional Neural Network for Human Trajectory Prediction," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 14412–14420, 2020.

[18] J. Oh, S. Singh, and H. Lee, "Value prediction network," in *Advances in Neural Information Processing Systems*, vol. 2017-Decem, pp. 6119–6129, 2017.

[19] J. Van Den Berg, S. J. Guy, M. Lin, and D. Manocha, "Reciprocal n-body collision avoidance," in *Springer Tracts in Advanced Robotics*, vol. 70, pp. 3–19, 2011.

[20] I. Osband, C. Blundell, A. Pritzel, and B. Van Roy, "Deep exploration via bootstrapped DQN," in *Advances in Neural Information Processing Systems*, pp. 4033–4041, 2016.

[21] J. Zhuang, T. Tang, Y. Ding, S. Tatikonda, N. Dvornek, X. Papademetris, and J. S. Duncan, "AdaBelief optimizer: Adapting stepsizes by the belief in observed gradients," in *Advances in Neural Information Processing Systems*, vol. 2020-Decem, 2020.