Transactions on Machine Learning and Artificial Intelligence - Vol. 10, No. 1
Publication Date: February, 25, 2022
DOI:10.14738/tmlai.101.11616.
Niijima, S., Sasaki, Y., & Mizoguchi, H. (2022). Real-Robot Friendly Passing Motion Planner for Autonomous Navigation in Crowds. Transactions on Machine Learning and Artificial Intelligence, 10(1). 27-40.

Real-Robot Friendly Passing Motion Planner for Autonomous Navigation in Crowds

Shun Niijima
Department of Mechanical Engineering
Tokyo University of Science
2641 Yamazaki, Noda-Shi, Chiba, 278-8510, Japan

Yoko Sasaki
National Institute of Advanced Industrial Science and Technology,
2-3-26 Aomi, Koto-ku, Tokyo 135-0064, Japan

Hroshi Mizoguchi
Department of Mechanical Engineering
Tokyo University of Science
2641 Yamazaki, Noda-Shi, Chiba, 278-8510, Japan

ABSTRACT

This study proposes a real-robot friendly passing motion planner to be used in crowds. The proposed method learns to pass pedestrians with smooth acceleration and deceleration by using passing motion learning. A key feature of the proposed method that it is trained on a simple crowd simulation with both of dynamic and stationary pedestrians. The learned passing behavior can be used directly in an autonomous navigation. Evaluations using the crowd simulations indicate that the proposed method outperforms the existing ones in terms of success rate, arrival time, and keeping a certain distance from the pedestrians. The proposed navigation framework is implemented on a mobile robot and demonstrated its successful navigation between pedestrians in a science museum.

Keywords: passing motion plan, dynamic environment, mobile robot, deep reinforcement learning

INTRODUCTION

Mobile robots are required to operate in close proximity to people, such as in shopping malls or exhibition halls. To move efficiently, many mobile robots use pre-generated map information of the environment to calculate a path, avoid obstacles, and move to its destination.



Figure 1. Mobile robot in crowds.

As shown in Figure 1, there are many pedestrians in a real environment, and a mobile robot needs to detect and avoid them with its on-board sensors to run safely. Motion planners usually make use of established parameterized approaches, such as the dynamic window approach (DWA)

or elastic bands [1], [2], which assume that the current position of the obstacle is static. However, in reality the obstacles can be dynamic as well. When using these methods, the robot assumes that the pedestrians are stationary, hence, it may get too close to them or even collide with them.

Velocity obstacle (VO) [3] was proposed to consider dynamic obstacles. VO is a planner that generates avoidance maneuvers by selecting the robot velocities outside the collision cone, which consists of velocities that would result in collision with obstacles moving at given velocities sometime in the future. The authors of [4] proposed reciprocal velocity obstacle (RVO), which is an extension of VO in a multi-agent environment. In RVO, each agent moves by considering the behavior of other agents to achieve a mutual collision avoidance. Moreover, the authors of [5] used a variant of RVO, namely, the optimal reciprocal collision avoidance (ORCA) to pro-actively avoid pedestrians. These behavioral policies enable the robot to move efficiently even among dynamic pedestrians. However, while these policies are sufficient in an artificial environment where all agents perform uniform movements, they are less optimal in the real world because pedestrians do not always move as expected. In addition, the motion model of the robot is not considered, thus, the robot may accelerate and turn rapidly and is forced to run dangerously.

There has been numerous research on methods for motion planning in dynamic environments. Several methods have improved performance by incorporating pedestrian motion models into navigation [6-8]. In [9], a socially aware recurrent neural network model is used to predict trajectories and set feasible sample waypoints as the optimal path. In [10], a Gaussian process (GP) is used to learn the motion patterns of pedestrians, and a path planning is performed by incorporating probabilities into the cost function of random rapid tree search. Although these methods are effective to ensure that the robot keeps an appropriate distance from the pedestrians, the freezing robot problem (FRP) [11] may occur when faced with dynamic pedestrians. For example, because the pedestrian prediction model contains uncertainty, the robot may not be able to find a safe path through the crowd.

This study proposes a learning method for the passing motion based on deep reinforcement learning (DRL). An approach of end-to-end motion planner based on DRL is very effective in difficult to model environments, such as crowds with many pedestrians. The proposed method trains a passing motion using a simple pedestrian passing simulation. The learned passing motion planner can be directly integrated into the navigation of a mobile robot, and experimental results have been obtained on a real robot.

RELATED WORK

With the rapid developments in the machine learning field and DRL, researchers have recently started using neural networks for robot navigation in dynamic environments [12]. These end-to-end navigation learning methods depend on the environment of the training data, hence, it is difficult to use them outside the trained environment. Therefore, they are combined with path planning methods, such as A*, to learn only the motion generation [13,14].

The method described in [15] includes a distributed multi-agent collision avoidance algorithm based on deep reinforcement learning. This research was later extended by modifying the reward function to ensure that the agents adhere to social norms [16]. Because these models

focus on moving pedestrians, they suffer from stoppages and collisions when faced with unrecognized pedestrians or stationary obstacles. In a real environment, these problems need to be addressed because some people stand still for various reasons, for example, to look at exhibits. Therefore, the aim of this research is to add stationary pedestrians to the learning process to ensure that the system learns to behave effectively in real environments.

In many cases, the network output is a rough discretized value, such as linear velocity or angular direction at a certain time. If the obtained behaviors are directly applied to the velocity commands of the robot, sudden robot velocity and pose changes may occur. Such sudden changes in speed makes pedestrians uncomfortable, therefore are not suitable for robots that operate among them. The proposed passing motion learning system learns to accelerate and decelerate smoothly from the current speed as the robot navigates. In addition, it uses a grid map of the robot's surroundings as its current state and outputs speed control commands directly from its input. This eliminates the need for dedicated nodes to calculate velocity commands and can be easily replaced with existing motion plans.

The main contribution of this study is that it proposes a transit motion learning method that can be directly integrated into the autonomous navigation of real robots. The main features of the proposed method are following: 1) the separation of the passage motion from the path planning makes the learning motion effective in various environments. 2) the method learns not only from moving pedestrians but also from stationary pedestrians. 3) the passage motion planning can learn smooth acceleration and deceleration motions to calculate the velocity command outputs. In the demonstration experiment at the exhibition, the effectiveness of the proposed method is confirmed by implementing it on a real robot.

PASSING MOTION LEARNING FOR MOBILE ROBOT NAVIGATION

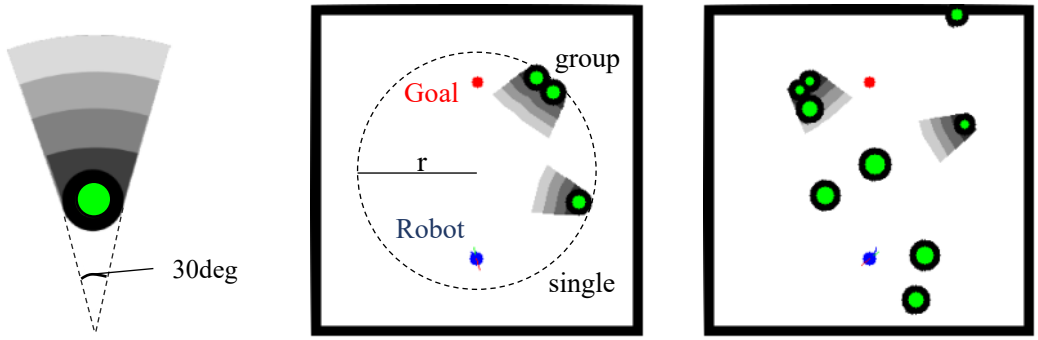
Simulation environment with pedestrians passing by

The learning of passing motion is carried out using a simulation that represents the crowds with pedestrians and robot. To simulate the passing motion of a robot and a pedestrian, it is required that pedestrian and robot move purposefully in a limited space. The simulation model represents pedestrians with a simple circle model. To produce a real-like crowd, the proposed simulation includes stationary pedestrians as well.

Figure 2 (a) shows the representation of a pedestrian model in simulation, where the position of the pedestrian is represented as a green circle with a radius r_p. The pedestrian circle is inflated by the radius of the robot r_r, which denotes the collision area. The predicted range of pedestrian movement is drawn in the direction of the pedestrian velocity as a cone with a central angle of 30 degrees. To learn moving without disturbing pedestrians, the simulation includes the predicted area. The predicted area representation of the 30 degrees cone is constructed by referring to the forward angle of the pedestrian field of view in the pedestrian modeling described in [17]. The size of prediction area is scaled using the pedestrian velocity. The color of the predictive area is represented by a grayscale with a 2 second range at every 0.5 second.

The simulation considers two environments according to the placement of pedestrians. One is an unobstructed space (figure 2 (b)), while the other contains randomly placed stationary pedestrians (figure 2 (c)). Walking pedestrians move toward the opposite side from where they

are placed on a circle. The robot aims to move from the initial position (blue dot) to the goal (red dot). The initial position of the robot and goal position are within the circle. To represent a dynamic environment, pedestrians move in two different walking patterns: single setting, where a pedestrian walks alone, and group setting, where two pedestrians walk side by side. Walking pedestrians are presented in random settings, and at most three people are present in the environment at a time. Pedestrians move according to the ORCA policy. The stationary pedestrians are arranged in random numbers and positions.



(a) Pedestrian model **(b) Unobstructed space** **(c) With stationary pedestrians**

Figure 2. Model and environment in simulation

Passing motion learning in simulation

In reinforcement learning (RL), the environment is formulated as a Markov decision process with state space **S**, action space **A**, state transition probability matrix **P**, reward function R, and discount factor $\gamma \in [0,1)$. The reward function R evaluates state **S** and returns a scalar reward when moving from the state $\mathbf{s}^{\mathbf{t}}$ to the next state $\mathbf{s}^{t+1}$. The goal is to find a policy π that maximizes the expected return for every state **S**.

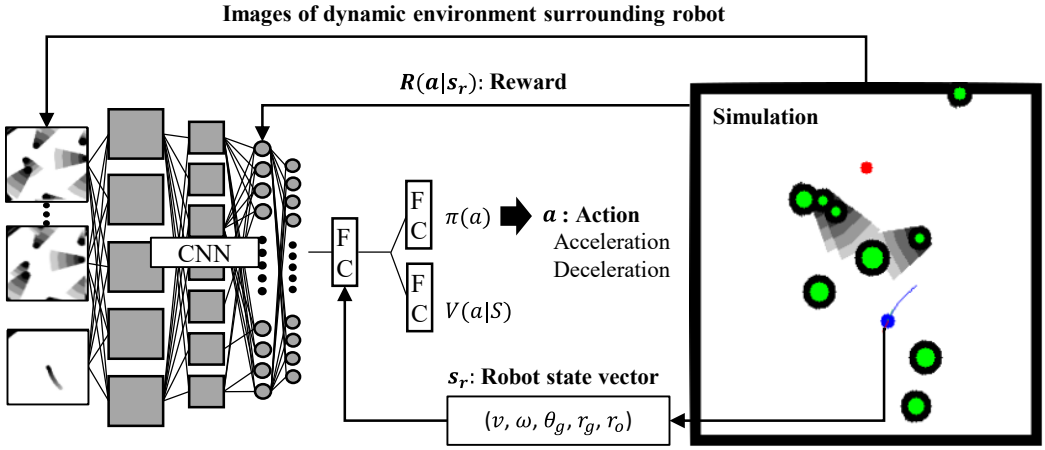


Figure 3. Passing motion learning using simulation

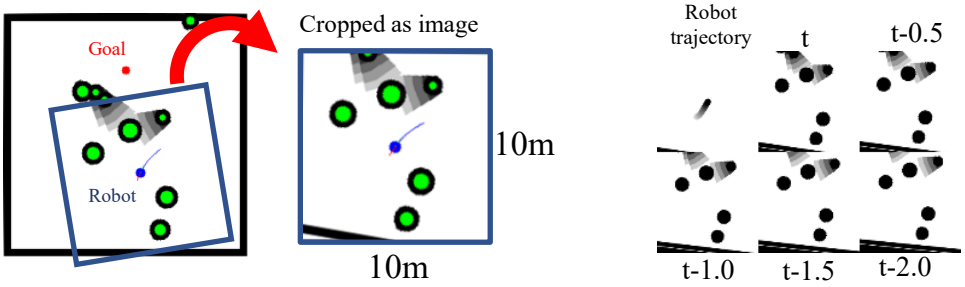
Figure 3 outlines of the learning process for passing motion planning. In the proposed network model, the state space S is obtained from simulation. One state is the robot state vector $\boldsymbol{s}_r = (v, \omega, \theta_r, r_g, r_o)$, where v and ω are the linear and angular velocity of the robot, respectively, θ_r is the robot pose, r_g is the distance to the goal, and r_o is the distance to closest obstacle. The

other state contains the information about dynamic environment surrounding robot. The simulation represents dynamic environment information surrounding robot as a time-series of images. The representation method is described in detail in section 3.3. The action space $\boldsymbol{A}$ consists of control commands for acceleration and deceleration obtained from the current speed, described as a tuple of velocity and rotation. The $\boldsymbol{\pi}(\boldsymbol{a})$ returns the likelihood of being the optimal action for each possible action $\boldsymbol{a}$ given a current state $\boldsymbol{S}$. The policy $\boldsymbol{\pi}(\boldsymbol{a})$ is found by taking actions and letting the agent explore its learning simulation with an objective of maximizing the cumulative rewards $R(\boldsymbol{a}|\boldsymbol{s}_r)$. The reward function $R(\boldsymbol{a}|\boldsymbol{s}_r)$ evaluates the robot state vector $\boldsymbol{s}_r$ to ensure that the robot can reach its destination without colliding with obstacles. The proposed network applies A3C [18] reinforcement learning framework to enable distributed learning.

A representation of dynamic environment

The simulation represents the dynamic environment as a time-series of images around the robot to compute a passing motion. This representation method refers to GOSELO [19] to represent the environment as an image. The image in current time t is generated, which is a square with 10 m long sides centered on the robot's position. Figure 4 (a) shows a cropped section of the image surrounding the robot. The image is rotated to represent the goal at the top. This image does not depend on the robot position or pedestrian orientation in the simulation; it learns only the relative positional relationship between the self-position and goal position. This is very effective in terms of reducing the learning time.

The final input images are shown in Figure 4 (b). Because the input image considers your past movements and those of the pedestrian, the input images are composed of six channel images: time-series of image at times $t, t-0.5, t-1.0, t-1.5,$ and $t-2.0$ seconds, and the trajectory image of the mobile robot over the last 2 seconds. The robot trajectory is an image that represents its self-positioning trajectory with respect to date. The input images are binarized to lower the number of dimensions and learning costs. The actionable area has a pixel value of 255 (white) and the collision area has a pixel value of 0 (black). The predicted pedestrian areas have gray pixel values of [50, 100, 150, 200] depending on the proximity to pedestrians.



(a) Cropped image of the robot surroundings (b) Information about the dynamic environment

Figure 4. Generating input images as a map state

Action and Reward design

The velocity command for the robot $u^t = (v^t, \omega^t)$ consists of linear speed v^t and angular velocity ω^t. The action space $\boldsymbol{A}$ is a combination of acceleration and deceleration based on the current speed $\boldsymbol{u}^t$. The neural network determines the next action $a = (\Delta v, \Delta\omega)$ using action

space A . The matrix of the action space A is determined by the parameter $[\mathrm{va}_{\max}, \omega \mathrm{a}_{\max}, \mathrm{R}_{\mathrm{v}}, \mathrm{R}_{\omega}]$, where $v\mathrm{a}_{\max}, \omega \mathrm{a}_{\max}$ represent the maximum acceleration and deceleration of the velocity command, and R_v and R_ω represent the resolution of the action space. The velocity command u^{t} is calculated from the obtained action **a** by:

$$v^t = \begin{cases} v_{max}(v^t > v_{max}) \\ v_{min}(v^t < v_{min}) \\ v^{t-1} + \Delta v\ (else) \end{cases}, \omega^t = \begin{cases} \omega_{max}\ (\omega^t > \omega_{max}) \\ \omega_{min}\ (\omega^t < \omega_{min}) \\ \omega^{t-1} + \Delta\omega\ (else) \end{cases} \quad (1)$$

where $\$ \mathrm{v}_{\max}, \mathrm{v}_{\min}, \omega_{\max}$, and $\omega_{\min}$ $\$$ represent the maximum and minimum speeds. This calculation model is based on the DWA [1], which is designed to prevent sudden speed and pose changes.

The objective of the reward function is to enable the network to learn the actions that are preferable within a given state $\mathbf{s}_r$. For this purpose, the sparse reward function is set as follows:

$$R(\mathbf{a}|\mathbf{s}_r) = \begin{cases} R_g = 15 & (if\ reached\ goal) \\ R_c = -15 & (if\ collision) \\ R_d + R_o + R_p & (else). \end{cases} \quad (2)$$

The reward function awards the agent with R_g when it reaches the goal and penalizes it with R_c when it collides with pedestrians or obstacles. The motion is evaluated by three functions: $R_d, R_o, and\ R_p$. R_d evaluates whether the robot approaches the goal at each step. It is calculated from the distance to the goal d^t as $R_d = 2.5 \cdot (d^t - d^{t-1})$. R_o evaluates the distance from the closest obstacles and prevents pedestrians and obstacles from invading the agent comfort zone. If the distance d_o between the agent and obstacle is smaller than the threshold d_{th}, it assigns the penalty $R_o = -0.02 \cdot d_o\ (if\ d_o < d_{th} = 2.0)$. R_p evaluates the predicted position of the robot. It is calculated using the predicted position of robot $\mathbf{p}_r$ and pixel value of the local map $M(\mathbf{p})$, as $R_p = -0.001 \cdot (255 - M(\mathbf{p}_r))$. The predicted position of the robot $\mathbf{p}_r$ is defined as the position estimate when the robot moves from its current position at current velocity for 2.0 s.

Integration with mobile robot navigation

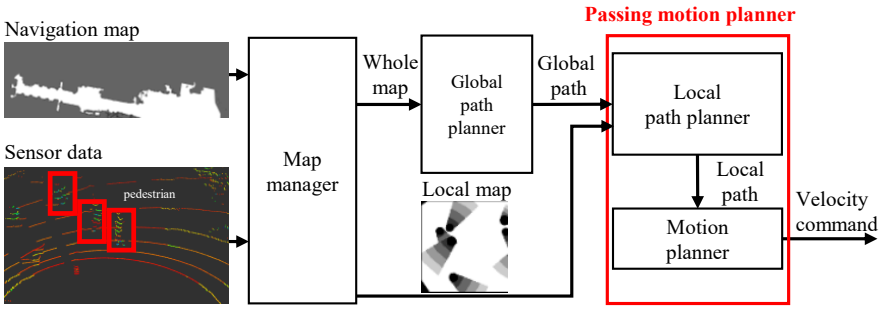


Figure 5. Mobile robot navigation with the proposed passing motion planner

Figure 5 showcases the navigation framework containing the proposed passing motion planner. The squares represent each module and arrows represent the inputs. The map manager updates the navigation map with the current obstacles based on the acquired sensor data. In a conventional navigation framework, this module provides the navigation map to two path planner modules. One is for global path planning, which calculates the rough path from the

initial position to the destination based on the entire navigation map, and the other is for local path planning, which calculates the path from the robot position to the set local goal on the global path. To avoid dynamic obstacles, it is executed at a high frequency. Finally, the motion planning module calculates the velocity command to follow the local path. In the proposed navigation framework, a passage motion planner is applied instead of the two conventional modules (red boxes in Fig. 5).

Evaluation

This section describes how the passing motion learning and proposed navigation framework were evaluated. First, the passing motion learning was evaluated using a simple simulator with pedestrians. Then, a simulation of mobile robot navigation was performed to evaluate the proposed navigation framework. Finally, the proposed navigation framework was incorporated into a mobile robot and demonstrated at a science museum.

Training settings

The proposed network model uses two CNN architectures. The first is AlexNet [20], which is known for its high performance in image recognition tasks. It consists of five convolutional layers and three fully connected layers. To reduce the training time, the AlexNet architecture used in this study can format the image size from 224 pixels to 112 pixels. The second is a small CNN architecture consisting of three convolutional layers and four fully connected layers. The architecture described in [21] was used as a reference to build the CNN framework used in this study. Figure 6 demonstrates the tiny CNN architecture.

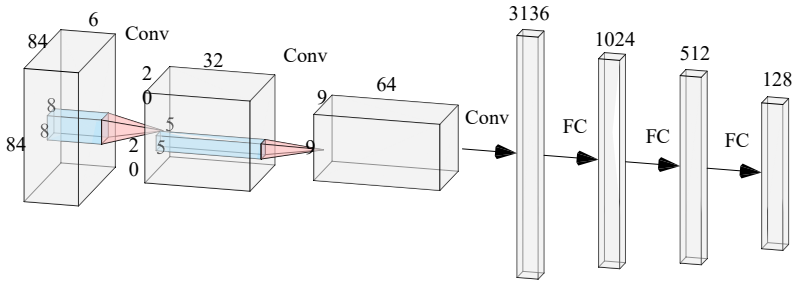


Figure 6. The tiny CNN architecture

The training was carried out in a total of 80 million steps. 30 million steps tok place in the unobstructed space environment, followed by 50 million steps in the environment including stationary pedestrians. Table 1 reports the simulation settings in detail. The number of stationary pedestrians is randomly set in the range 5 - 15.

Table 1. The simulation setting details

Environment		**Robot**	
$\boldsymbol{L}$: Size of simulator	15 m	$\boldsymbol{r_r}$: Radius	0.3 m
$\boldsymbol{R}$: Resolution of image	0.05 m	No. of action spaces	25 ($\boldsymbol{R_v = 5, R_\omega = 5}$)
$\boldsymbol{dt}$: Time interval	100 ms/step	$\boldsymbol{v}$: Linear velocity	[0.0, 1.2] m/s
Pedestrian		$\boldsymbol{\omega}$: Angular velocity	[-200, 200] deg/s
$\boldsymbol{r_p}$: Radius	[0.2, 0.4] m	$\boldsymbol{va_{max}}$: Max. linear acc.	1.0 m/$\mathbf{s^2}$
Motion policy	ORCA	$\boldsymbol{\omega a_{max}}$: Max. angular acc.	100 deg/$\mathbf{s^2}$
Pref. velocity	1.2 m/s		

Passing motion in a learning simulation

The passing motion planner is evaluated with the environment including stationary pedestrians as shown in Figure 2. The initial position and pattern of the pedestrians were random, as well as the initial pose and position of the robot. As comparative baselines, (A*+DWA), ORCA, and CADRL methods are used. A*+DWA is a conventional navigation method. ORCA is a mutual collision avoidance method, where several independent mobile robots or agents avoid colliding with each other without any communication between them. The CADRL is decentralized multiagent collision avoidance algorithm based on deep reinforcement learning. This method trains the motion model considering only moving pedestrians. Tests were conducted considering the same 500 episodes of randomized environments for all baseline models.

This evaluation compared the following metrics: success rate, arrival time, minimum distance to pedestrians, and maximum change in velocity and pose within a time step. The average values in successful episodes were recorded. Table 2 reports the summarized results**.** Figures 6 and 7 show two comparisons of the movement trajectories of pedestrians and robot in one episode. The solid blue circles represent the robot, and the other circles represent pedestrians. The red star represents the destination. The black numbers on the trajectory represent simulation time.

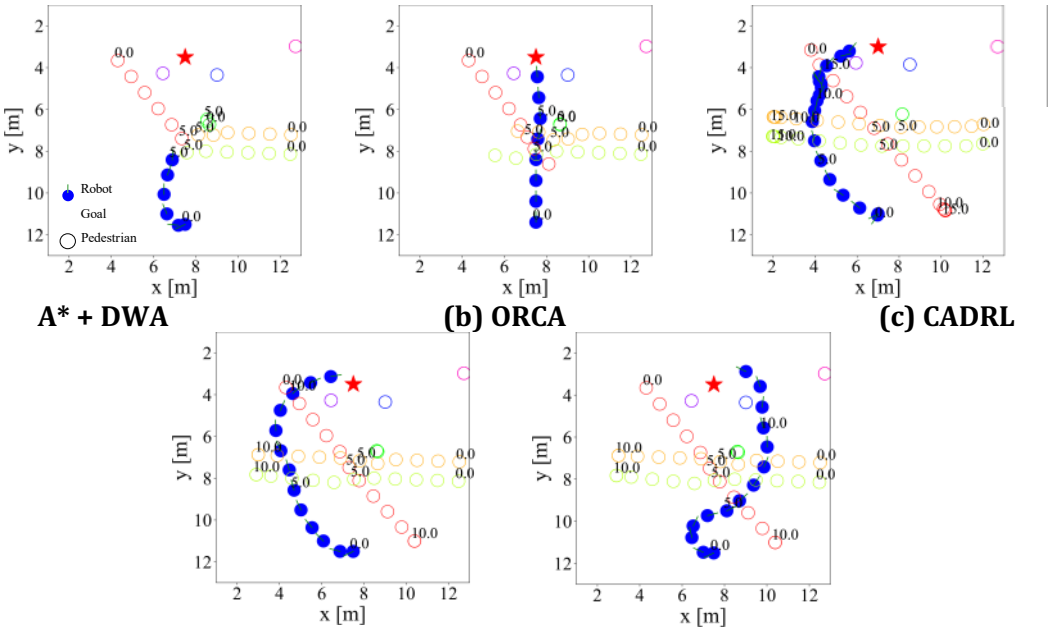
With conventional navigation, the robot based on A*+DWA goes straight to the destination according to the path plan, which leads to the robot trying to cross in front of the pedestrians as shown in Figure 7 (a). A* cannot change the robot path when coming from behind, thus, it collides with the pedestrian as shown in Figure 8 (a). ORCA prioritizes the speed in the direction of the destination if there is no collision, hence, excessive approach and sudden change of speed occurs near the center where the pedestrian and robot meet. CADRL can perform evasive maneuvers when encountering a moving pedestrian as shown in Figure 7 (c). However, it gets stuck in a scene with a group of stationary pedestrians in front as shown in Figure 8 (c) because it does not consider stationary obstacles. In figure 7, the proposed robot can be seen moving efficiently without disturbing the pedestrian's movement. Moreover, in figure 8 it can be seen avoiding the pedestrian approaching from behind. In figure 8, the robot appears to wait for pedestrians to pass before proceeding without disturbing them.

The proposed method achieved a higher success rate and lower arrival time compared all baselines except ORCA. ORCA achieved better results in these metrics because there were no collisions due to its design policy. However, a large change in posture occurred with ORCA.

CADRL caused the robot to make a large turn at a sufficient distance to not hit a pedestrian, which means it took more time to reach its destination. The success rate is low owing to be stuck stationary pedestrians. These results demonstrate that the proposed method made sure that the robot stayed a sufficient distance away from pedestrians and moved efficiently without a sudden change of pose.

Table 2. Results of simple passing motion simulation with different methods

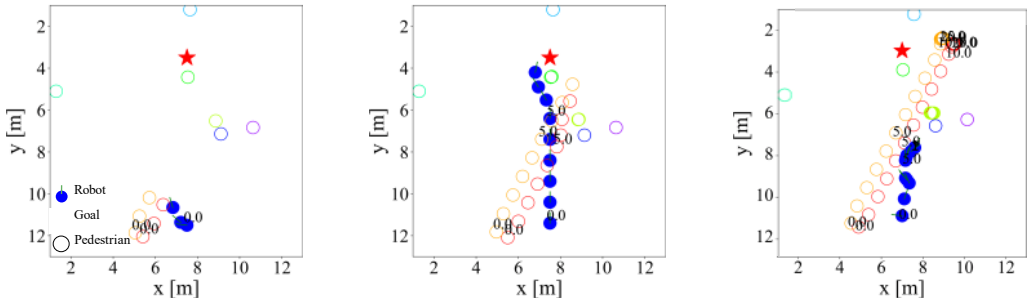
Method	Success rate [--]	Arrival time [s]	Min. dis. of ped. [m]	Max. Δv [m/s step]	Max. $\Delta\omega$ [rad/step]
A*+DWA	0.30	10.9 $\pm$ 2.77	0.30 $\pm$ 0.16	0.08 $\pm$ 0.03	0.11 $\pm$ 0.05
ORCA	1.00	7.45 $\pm$ 1.00	0.46 $\pm$ 0.12	1.20 $\pm$ 0.00	0.46 $\pm$ 0.12
CADRL	0.42	11.3 $\pm$ 2.98	1.06 $\pm$ 0.99	0.79 $\pm$ 0.33	0.60 $\pm$ 0.49
Proposed (AlexNet)	0.95	10.3 $\pm$1.19	1.00 $\pm$ 0.42	0.09 $\pm$ 0.03	0.04 $\pm$ 0.03
Proposed (Tiny CNN)	0.96	10.4 $\pm$1.54	0.93 $\pm$ 0.38	0.09 $\pm$ 0.05	0.03 $\pm$ 0.02



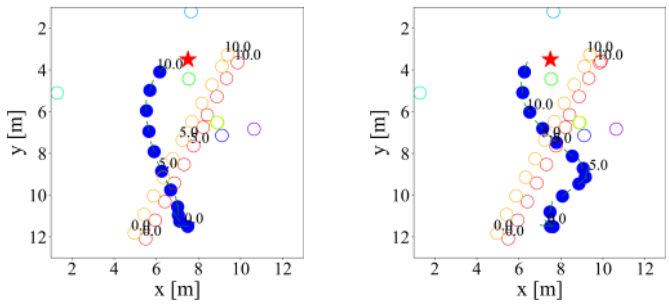
A* + DWA **(b) ORCA** **(c) CADRL**

(d) Proposed (AlexNet) **(e) Proposed (Tiny CNN)**

Figure 7. Comparison of trajectory with each method



A* + DWA **(b) ORCA** **(c) CADRL**



(d) Proposed (AlexNet) **(e) Proposed (Tiny CNN)**

Figure 8. Comparison of trajectory with each method

Mobile robot navigation with passing motion

This section shares the evaluation of the proposed navigation framework in the simulation environment presented in figure 9. This environment is a square with 50 m long sides. Ten moving pedestrians (represented by green cylinders) move through the square, and forty pedestrians remain stationary (represented by red cylinders). The pedestrians randomly choose a goal among nine destinations (in yellow) and walk towards that destination according to the ORCA policy.

The simulation used the Pioneer P3-DX model (https://github.com/mario-serna/pioneer_p3dx_model) as the mobile robot model. The robot was equipped with 3D LIDAR (https://github.com/lmark1/velodyne_simulator) to recognize the surrounding environment. In the experiment, the robot randomly selects nine destinations and performs autonomous navigation. Once it decides its destination, local goals are placed at 7-m intervals on the path drawn by A*. The robot moves to the next local goal when it is within 2 m of the local goal. This process is repeated until the robot reaches its destination or collides with obstacles. Gazebo [22] was used to implement the simulation.

The navigation test ran 100 episodes for evaluation. The proposed navigation framework was evaluated compared to the existing navigation method of A*+DWA. The following metrics were investigated: success rate, average speed, distance to pedestrians, and velocity and posture changes in a step. The average values in successful episodes were recorded.

Table 3 lists the results. The proposed navigation framework achieved a higher success rate than that of the conventional one owing to the proposed passing motion, which enabled pedestrians approaching from the side to be avoided. The arrival time and average speed remained approximately the same when A* was used, indicating that the proposed method achieved an efficient passing motion. The proposed methods have no sudden posture change. The main cause of collision with pedestrians was when the pedestrian arrived at a destination and suddenly changed their behavior to reach the next destination.

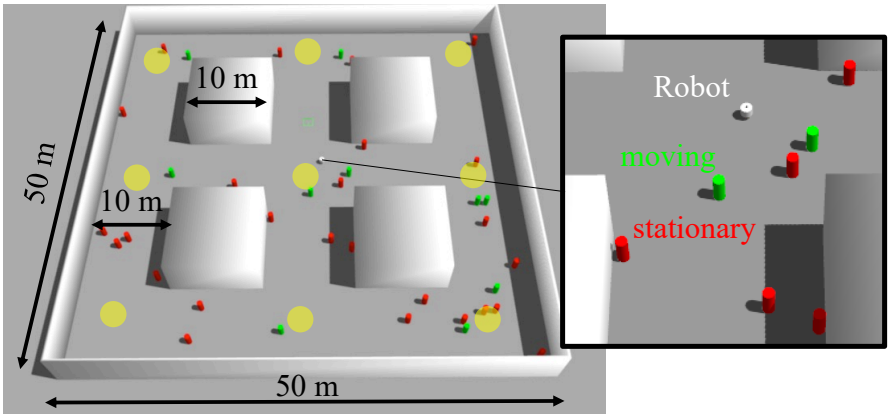


Figure 9. Simulation environment for evaluating mobile robot navigation.

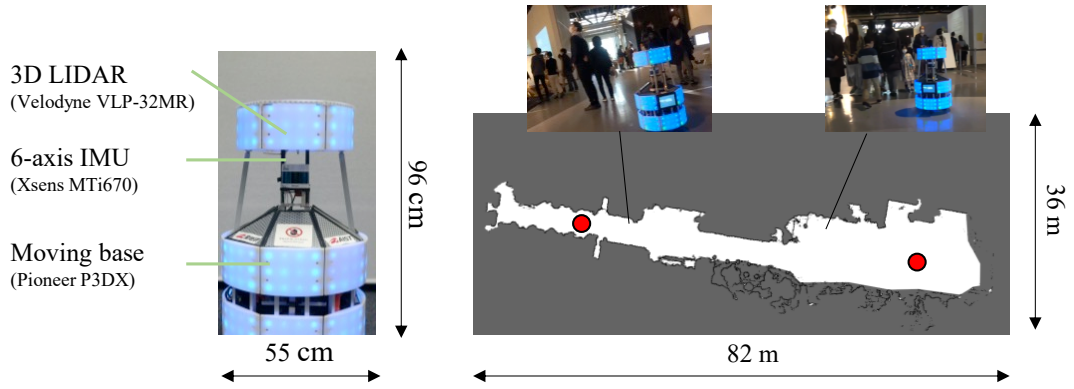
Table 3. RESULTS OF MOBILE ROBOT NAVIGATION IN SIMURATION.

Method	Success rate [--]	Velocity [m/s]	Min. dis. of ped. [m]	Max. Δv [m/s step]	Max. $\Delta\omega$ [rad/step]
A*+DWA	0.53	1.10	1.23 ± 0.52	0.11 ± 0.01	0.13 ± 0.11
Proposed (AlexNet)	0.87	1.17	0.93 ± 0.44	0.10 ± 0.02	0.11 ± 0.05
Proposed (Tiny CNN)	0.89	1.16	0.91 ± 0.42	0.09 ± 0.01	0.13 ± 0.05

Real-world demonstration

The proposed navigation framework was implemented on a mobile robot to test its real-world performance. Figure 10 (a) shows the mobile robot, which includes a PC (Intel, NUC10FNK) with an Intel Core i7-10710U CPU consisting of six 2-threaded cores. The robot is equipped with a 3D LIDAR (Velodyne VLP-32MR) and 6-axis IMU (Xsens Mti670) sensors. The 3D LIDAR detects the surrounding obstacles and represents them on the local map as point cloud data. The MCL-based 6-dof localization method [23] using a 3D LIDAR and multiple IMUs is applied in the demonstration. For detecting moving pedestrians, a moving object tracker (mo tracker) [24] utilizing sample-based joint probabilistic data association filters (SJPDAFs) is applied. The localization, path planning, and motion planning operate at 10 Hz.

The demonstration experiment was carried out in a science museum as shown in figure 10 (b). Similar to the experiment described section in 4.3, three baseline methods were compared: A*+DWA, proposed-Alex, and proposed-tiny CNN. To evaluate the efficiency of each method, autonomous navigation was performed for 30 minutes using each method; the average lap times were compared. The comparison results according to the lap time of each method are presented in figure 11. The proposed method runs at a shorter lap time than those of the conventional methods, which indicates that the proposed method is more efficient.



Mobile robot with sensors. **(b) Navigation map of the science museum.**

Figure 10. Experimental settings in the science museum

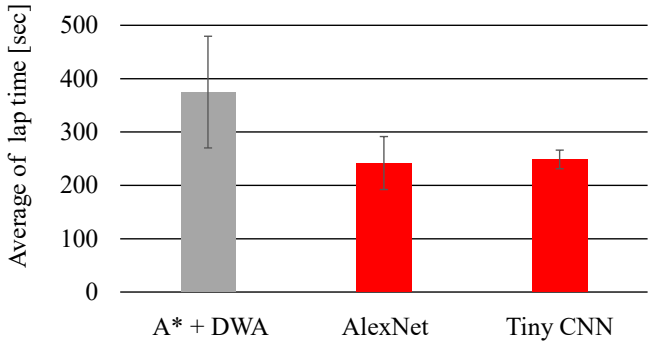


Figure 11. Average of lap time obtained using each method

CONCLUSION

This study presents a method for learning passing movements for autonomous robots using a simulator in a dynamic environment. The simulation includes static pedestrians to simulate a crowded environment more accurately. The proposed framework is designed to be easily integrated into real robots. An important feature of this framework is its ability to generate smooth motion without interfering with the pedestrian. In addition, owing to separating the learning of the passage motion from the path planning, the learned motion is effective in various environments. Experiments show that the learned passing motion achieves a higher success rate than those of conventional methods and an effective passing motion with smooth acceleration and deceleration can be obtained. The proposed framework was implemented on a mobile robot in a science museum, which successfully navigated between pedestrians.

References

D. Fox, W. Burgard, and S. Thrun, “The dynamic window approach to collision avoidance,” IEEE Robotics & Automation Magazine, vol. 4, no. 1, pp. 23–33, 1997

S. Quinlan and O. Khatib, “Elastic bands: connecting path planning and control,” in [1993] Proceedings IEEE International Conference on Robotics and Automation, 1993, pp. 802–807 vol.2

P. Fiorini and Z. Shiller, "Motion planning in dynamic environments using velocity obstacles," The International Journal of Robotics Research, vol. 17, no. 7, pp. 760-772, 1998.J. Van den Berg, M. Lin, and D. Manocha, “Reciprocal velocity obstacles for real-time multi-agent navigation,” in 2008 IEEE International Conference on Robotics and Automation. IEEE, 2008, pp. 1928–1935

J. Van Den Berg, S. J. Guy, M. Lin, and D. Manocha, “Reciprocal n-body collision avoidance,” in Robotics research. Springer, 2011, pp. 3–19.

D. Helbing and P. Molnar, “Social force model for pedestrian dynamics,” Physical review E, vol. 51, no. 5, p. 4282, 1995.

D. Helbing and P. Molnar, “Social force model for pedestrian dynamics,” Physical review E, vol. 51, no. 5, p. 4282, 1995.

S. Pellegrini, A. Ess, K. Schindler, and L. van Gool, “You’ll never walk alone: Modeling social behavior for multi-target tracking,” in 2009 IEEE 12th International Conference on Com- puter Vision, 2009, pp. 261–268.

A. Vemula, K. Muelling, and J. Oh, “Modeling cooperative navigation in dense human crowds,” in 2017 IEEE International Conference on Robotics and Automation (ICRA). IEEE, 2017, pp. 1685–1692.

Z. Chen, C. Song, Y. Yang, B. Zhao, Y. Hu, S. Liu, and J. Zhang, “Robot navigation based on human trajectory prediction and multiple travel modes,” Applied Sciences, vol. 8, no. 11, p. 2205, 2018.

B. D. Luders, G. S. Aoude, J. M. Joseph, N. Roy, and J. P. How, “Probabilistically safe avoidance of dynamic obstacles with uncertain motion patterns,” Tech. Rep., 2011

P. Trautman and A. Krause, “Unfreezing the robot: Navigation in dense, interacting crowds,” in 2010 IEEE/RSJ International

H.-T. L. Chiang, A. Faust, M. Fiser, and A. Francis, “Learning navigation behaviors end-to-end with autorl,” IEEE Robotics and Automation Letters, vol. 4, no. 2, pp. 2007–2014, 2019

T. Fan, X. Cheng, J. Pan, D. Manocha, and R. Yang, “Crowd- move: Autonomous mapless navigation in crowded scenarios,” arXiv preprint arXiv:1807.07870, 2018.

M. Pfeiffer, S. Shukla, M. Turchetta, C. Cadena, A. Krause, R. Siegwart, and J. Nieto, “Reinforced imitation: Sample efficient deep reinforcement learning for mapless navigation by leveraging prior demonstrations,” IEEE Robotics and Automation Letters, vol. 3, no. 4, pp. 4423–4430, 2018.

C. Chen, Y. Liu, S. Kreiss, and A. Alahi, “Crowd-robot interac- tion: Crowd-aware robot navigation with attention-based deep reinforcement learning,” in 2019 International Conference on Robotics and Automation (ICRA), 2019, pp. 6015–6022.

M. Everett, Y. F. Chen, and J. P. How, “Motion planning among dynamic, decision-making agents with deep reinforcement learn- ing,” in 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2018, pp. 3052–3059.

Min Zhou, Hairong Dong, Fei-Yue Wang, Qianling Wang, Xiaoxia Yang, Modeling and simulation of pedestrian dynamical behavior based on a fuzzy logic approach, Information Sciences, Vol. 360, 2016, pp. 112-130.

V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. Harley, T. P. Lillicrap, D. Silver, and K. Kavukcuoglu, “Asynchronous meth- ods for deep reinforcement learning,” in Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48, ser. ICML’16. JMLR.org, 2016, p. 1928–1937.

A. Kanezaki, J. Nitta, and Y. Sasaki, “Goselo: Goal-directed obstacle and self-location map for robot navigation using reactive neural networks,” IEEE Robotics and Automation Letters, vol. 3, no. 2, pp. 696–703, 2018

A. Krizhevsky, I. Sutskever, and G. Hinton, “Imagenet classifi- cation with deep convolutional neural networks,” Neural Infor- mation Processing Systems, vol. 25, 01 2012.

Y. Fujita, P. Nagarajan, T. Kataoka, and T. Ishikawa, “Chainerrl: A deep reinforcement learning library,” Journal of Machine Learning Research, vol. 22, no. 77, pp. 1–14, 2021.

N. Koenig and A. Howard, “Design and use paradigms for gazebo, an open-source multi-robot simulator,” in IEEE/RSJ International Conference on Intelligent Robots and Systems, Sendai, Japan, Sep 2004, pp. 2149–2154.

A. Ohsato, Y. Sasaki, and H. Mizoguchi, “Real-time 6Dof localization for a mobile robot using pre-computed 3D laser likelihood field,” in 2015 IEEE International Conference on Robotics and Biomimetics (ROBIO 2015), China, 2015, pp. 2359–2364.

N. Hatao and S. Kagami, “Cluster-based sjpdafs for classification and tracking of multiple moving objects,” in Field and Service Robotics. Springer, 2015, pp. 303–317.