2026 IEEE/SICE International Symposium on System Integration (SII)
January 11-14, 2026. Cancún, México

Moving Obstacle Avoidance using MPPI based on Cost Maps Considering Velocity and Predicted Poses

Yuuka Iwamura*

* Graduate School of Science and Technology, Meiji University, Kanagawa, Japan

Yoshitaka Hara†

† Future Robotics Technology Center (fuRo), Chiba Institute of Technology, Chiba, Japan

Yoji Kuroda‡

‡ School of Science and Technology, Meiji University, Kanagawa, Japan

***Abstract*— In this paper, we propose a method of motion planning using MPPI based on obstacle cost maps, to realize stable avoidance behavior for moving obstacles. The proposed method creates cost maps considering moving obstacles, and performs motion planning based on cost function formulation with MPPI. In cost map creation, we propose a cost shape that takes into account velocity and predicted poses of moving obstacles. In motion planning, we adopt MPPI which is one of the mainstream sampling-based methods. To apply cost maps to MPPI, we formulate cost functions of MPPI that appropriately avoid obstacles. In simulation experiments, we performed moving obstacle avoidance in an environment with 50 pedestrians, and our proposed method achieved the highest success rate (94%). Our method realized stable avoidance behavior for moving obstacles even in crowded environments, by creating cost maps considering velocity and predicted poses of moving obstacles, and performing motion planning based on these maps using MPPI.**

I. Introduction

Autonomous mobile robots require navigation while avoiding not only stationary obstacles but also moving obstacles, including pedestrians. Motion planning is necessary that appropriately avoid obstacles near the robot.

Model Predictive Path Integral Control (MPPI) [1] has become one of the mainstream approaches for sampling-based motion planning. Unlike Model Predictive Control (MPC), MPPI has the advantage that it can handle non-linear and non-differentiable cost functions, such as cost maps. Furthermore, since it is a sampling-based method, combining parallelization and GPU processing enables high-speed computation.

Cost functions of MPPI can be designed arbitrarily and are not uniquely defined. Therefore, when using MPPI for moving obstacle avoidance, designing suitable cost functions is required.

Fig. 1 shows the concept of the proposed method. In this paper, we aim to realize stable avoidance behavior for moving obstacles, and propose a motion planning method using MPPI based on obstacle cost maps. Our proposed method creates cost maps considering velocity and predicted poses of moving obstacles and formulates cost functions of MPPI that appropriately avoid obstacles.

The contributions of this paper are as follows:

- We propose a cost shape that considers velocity and predicted poses of moving obstacles, and create obstacle cost maps based on it.
- To apply created cost maps, we formulate MPPI cost functions that avoid obstacles appropriately.
- We conducted pedestrian avoidance experiments in crowded environments using a simulator to demonstrate the usefulness of the proposed method.

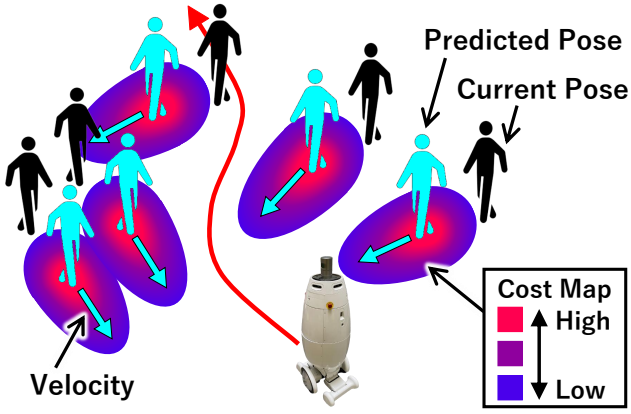


Fig. 1. Concept of cost maps that take into account velocity and predicted poses of moving obstacles.

II. Related Work

A. Motion Planning for Obstacle Avoidance

Representative methods for motion planning to avoid obstacles include Dynamic Window Approach (DWA) [2], Vector Field Histogram (VFH) [3] and Potential Field [4].

DWA [2] can avoid obstacles that consider the robot kinematics and dynamics. Motion planning is performed by sampling arc-shaped paths.

VFH [3] is a motion planning method that represents the shape of obstacles using polar coordinate histograms centered on the robot. The movement direction of the robot is divided into angular segments and sampled as straight-line paths.

Potential Field [4] generates repulsive forces from obstacles and attractive forces from the goal, and performs motion planning toward the direction of lower potential. This enables the robot to navigate to the goal while avoiding obstacles based on the potential.

MPPI [1] used in the proposed method is also one of the mainstream approaches for motion planning. Since sampling is based on probability distributions, solutions can be found from paths of various shapes. Furthermore, arbitrary cost functions such as cost maps can be applied.

Also, there are methods using machine learning for motion planning in environments with moving obstacles such as pedestrians [5] [6] [7] [8].

Maddumage et al. [5] proposed a motion planning method using reinforcement learning that considers relative velocity between pedestrians and the robot. Instead of the circular penalty areas that only consider distance, they use penalty areas extended in the direction of relative velocity to enable consideration of interactions between pedestrians and the robot.

Kanezaki et al. [6] proposed a motion planning method using Goal-Directed Obstacle and Self-Location (GOSELO) maps, which are obstacle maps that are cropped, rotated, and rescaled according to the target pose and the current pose of the robot. This enables obstacle maps to be represented that correlate with the movement of the robot, thereby improving the generalization performance of navigation using a CNN.

Matsumoto et al. [7] proposed a motion planning method that switches between learning-based and rule-based approaches. It combines motion planning based on a learning-based approach in intended environments, and motion planning based on a rule-based approach to reliably avoid collisions in unintended environments.

Nishimura et al. [8] proposed a method that avoids collisions with pedestrians in crowded environments, while also requesting surrounding pedestrians to clear the path by the robot emitting sounds as necessary. Through human-robot interaction, robots achieve both safe and efficient movement.

The proposed method performs motion planning using MPPI without using machine learning. By sampling based on probability distributions, we ensure diversity of paths for obstacle avoidance. Furthermore, we explicitly represent moving obstacles in cost maps and apply costs considering velocity and predicted poses to MPPI.

B. Cost Maps for Obstacle Avoidance

Lu et al. [9] proposed Layered Costmap for creating cost maps by integrating Static Layer, Obstacles Layer, and Inflation Layer. This enables handling of not only static obstacles, but also dynamic (quasi-static) obstacles. Obstacles Layer reflects poses of acquired dynamic obstacles on cost maps, but does not consider their velocity. In contrast, Chen et al. [10] added Velocity Obstacle Layer that represents velocity based on Velocity Obstacle [11].

Furthermore, Jian et al. [12] proposed a method that creates cost maps by predicting obstacle movement and then assigning costs considering velocity around predicted poses. However, the expansion size except for the direction of obstacle movement that considers velocity is determined by fixed parameters. Movement prediction is always subject to error, but this method cannot take into account those errors. In addition, motion planning is performed using DWA [2].

C. Obstacle Avoidance using MPPI

MPPI is one of the mainstream sampling-based motion planning approaches and is also implemented in ROS 2 Navigation2 [13] [14]. Other related work using MPPI include SVG-MPPI [15], which addresses the multimodality of optimal path distributions for obstacle avoidance, and SCP-MPPI [16], which combines path sampling with spline interpolation. In these methods, the Stein Variational Gradient Descent (SVGD) method [17] is applied to the MPPI. Furthermore, MPPI can be applied to mobile robots with various drive systems. For example, 4WIDS-MPPI [18] has been proposed applied to a four-wheel independent drive and steering (4WIDS) vehicle.

Additionally, there are methods that perform moving obstacle avoidance using MPPI [19] [20].

Suresh et al. [19] proposed a method that creates elliptical cost maps with the direction of obstacle movement as the major axis and combines them with MPPI. However, velocity and predicted poses of obstacles are not considered, and the same size cost is set for all obstacles. Although our proposed method and their approach are similar, there is a difference in whether only the direction of obstacle movement is considered or whether both velocity and predicted poses are also considered.

Also, Mohamed et al. [20] performed movement prediction using Linear Kalman Filter (LKF), and combined collision checks with its trajectory using MPPI. However, since only collision checks with the trajectory are performed and the cost map concept is not applied, paths that traverse the immediate vicinity of obstacles may be selected unless a collision occurs.

III. Motion Planning Considering Moving Obstacles

A. Overview

As shown in Fig. 1, the proposed method creates cost maps considering velocity and predicted poses of moving obstacles, and then performs motion planning using MPPI based on these maps. Cost functions of MPPI used in motion planning reflect obstacle costs from created cost maps and are designed to enable proper obstacle avoidance. In this paper, simulation experiments are conducted, while velocity and poses of moving obstacles are obtained from the simulator.

The movement prediction uses constant velocity model [21] that assumes obstacles move at constant velocity. The obstacle cost is centered on the predicted pose and is shaped to consider not only velocity but also prediction error.

The implementation of MPPI in this paper uses MPPI-Generic [22]. MPPI-Generic is a library implemented in C++/CUDA. By the way, there are also other libraries such as mppi_playground [23] and python_simple_mppi [24] implemented in Python.

B. Cost Design of Moving Obstacles

Fig. 2 shows the concept of obstacle cost design considering velocity and predicted pose. The obstacle cost is set centered on the predicted pose, with its length increasing in

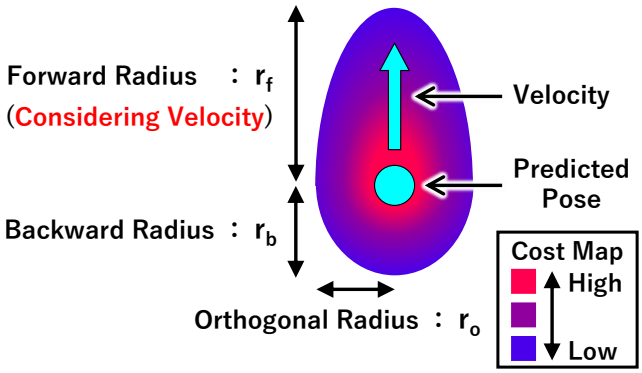


Fig. 2. Obstacle cost is designed considering velocity and predicted pose.

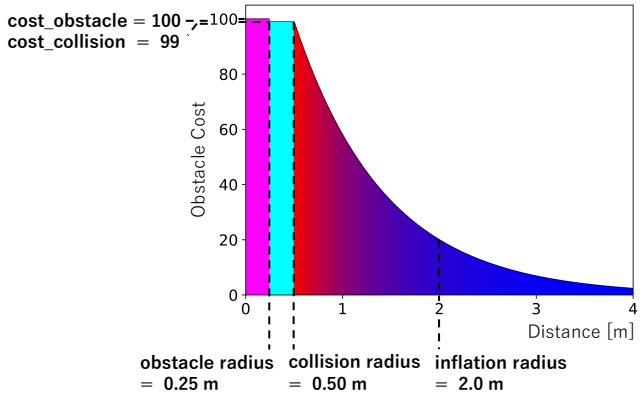


Fig. 3. Example of costs based on distance from obstacles.

the direction of obstacle movement according to velocity, and its size increasing according to the distance from the robot to the obstacle to take into account prediction error.

By making obstacle costs longer according to velocity, the robot is made less likely to select paths that cross the direction of obstacle movement. In addition, designing an obstacle cost larger according to the distance between the robot and the obstacle reflects the increasing prediction error for obstacles with longer prediction times (i.e., obstacles farther from the robot) in the obstacle cost.

Fig. 3 shows an example of obstacle costs based on distance from obstacles. The cost of obstacle radius (obstacle radius in Fig. 3) is set to 100, and the cost of collision radius (collision radius in Fig. 3) is set to 99. Then, the cost of the range beyond collision radius is set based on Eq. (1) [9].

$$C_{\text{cell}}(\boldsymbol{x}) = (100 - 1) \times \exp\left(-1.0 \times w \times (d - r)\right) \quad (1)$$

Where $C_{\text{cell}}(\boldsymbol{x})$ is the cost of the cell in the cost map corresponding to pose $\boldsymbol{x}$. w is the weighting parameter and set to 1.066, and is determined so that Obstacle Cost of inflation radius in Fig. 3 is 20.0. Also, d and r correspond to the distance and collision radius in Fig. 3, respectively.

The inflation radius of obstacle costs is determined separately for the forward direction, backward direction, and orthogonal direction. Then, the obstacle cost value is set so that the cost at the inflated boundary becomes 20.0.

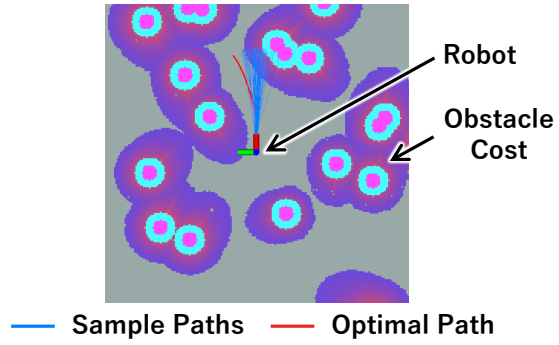


Fig. 4. Proposed motion planning using MPPI based on the cost map.

The inflation radius r_f in the forward direction of the obstacle cost illustrated in Fig. 2 is determined based on Eq. (2). The inflation radius r_b in the backward direction and the inflation radius r_o in the orthogonal direction are determined based on Eq. (3).

$$r_f = l_{\min} + (l_{\max} - l_{\min}) \left(\alpha \frac{r_{\text{obs}}}{r_{\max}} + \beta \frac{v_{\text{obs}}}{v_{\max}} \right) \quad (2)$$

$$r_b = r_o = s_{\min} + (s_{\max} - s_{\min}) \frac{r_{\text{obs}}}{r_{\max}} \quad (3)$$

Where $l_{\max}$ is the maximum value of the inflation radius in the forward direction, and $l_{\min}$ is the minimum value. $s_{\max}$ is the maximum value of the inflation radius in the backward direction and the orthogonal direction, and $s_{\min}$ is the minimum value. Furthermore, r_{obs} is the distance between the robot and the obstacle, v_{obs} is velocity of the obstacle, and α and β are weighting parameters. For normalization, the distance between the robot and the obstacle is divided by $r_{\max}$, the upper limit of obstacle distance reflected in cost maps, and velocity on the obstacle is divided by $v_{\max}$, its maximum value.

C. Motion Planning using MPPI

Fig. 4 shows the behavior of motion planning using MPPI based on obstacle costs with the proposed method. The state transition model of the robot uses a differential drive model, and the cost function is designed as shown in Eq. (4).

$$C(\boldsymbol{x}_t) = \gamma \frac{C_{\text{goal}}(\boldsymbol{x}_t)}{C_{\text{goal\_max}}} + \delta |v_t| \frac{C_{\text{map}}(\boldsymbol{x}_t)}{C_{\text{map\_max}}} \quad (4)$$

Where γ and δ are weighting parameters. $\boldsymbol{x}_t$ is the state of the robot at time t for each sample, represented by $(x_t, y_t, \theta_t, v_t, \omega_t)$. x_t, y_t, θ_t are the robot pose, v_t, ω_t are velocity and angular velocity of the robot. $C(\boldsymbol{x}_t)$ is the state cost, and the terminal cost uses the same.

The goal cost term $C_{\text{goal}}(\boldsymbol{x}_t)$ and the obstacle cost term $C_{\text{map}}(\boldsymbol{x}_t)$ are designed as shown in Eq. (5) and Eq. (6), respectively.

$$C_{\text{goal}}(\boldsymbol{x}_t) = \|\boldsymbol{x}_t - \boldsymbol{x}_{\text{goal}}\|_{\text{xy\_dist}} \quad (5)$$

$$C_{\text{map}}(\boldsymbol{x}_t) = \begin{cases} \infty & (C_{\text{cell}}(\boldsymbol{x}_t) \geqq 99) \\ C_{\text{cell}}(\boldsymbol{x}_t) & (C_{\text{cell}}(\boldsymbol{x}_t) < 99) \end{cases} \quad (6)$$

Where in Eq. (4), for normalization relative to Eq. (5) and Eq. (6), each is divided by the maximum distance to the goal $C_{\text{goal\_max}}$ and the maximum obstacle cost $C_{\text{map\_max}}$, respectively.

In Eq. (4), in order to prevent the robot from passing through at high speed when the cost map is filled with obstacle costs, the obstacle cost term $C_{\text{map}}(\boldsymbol{x}_t)$ is multiplied by the absolute value of velocity $|v_t|$. If this is not done, when the obstacle cost $C_{\text{map}}(\boldsymbol{x}_t)$ becomes similarly large for all samples (i.e., when the map is filled with obstacle costs), the control input "accelerates" will be selected to reduce the goal cost $C_{\text{goal}}(\boldsymbol{x}_t)$. On the other hand, multiplying by the absolute value of velocity $|v_t|$ imposes a larger penalty when passing through areas with high obstacle costs at high speeds. This makes the control input select either "slows down" or "avoids areas with high obstacle costs".

In Eq. (5), a penalty is imposed based on the goal cost with greater distances given greater penalties. $\boldsymbol{x}_{\text{goal}}$ is the goal pose, and $\|\boldsymbol{x}_t - \boldsymbol{x}_{\text{goal}}\|_{\text{xy\_dist}}$ is the distance between the robot and the goal at time t for the sample in the xy-plane. Local Goal is given as this goal.

In Eq. (6), a larger penalty is given for passing through areas with higher obstacle costs. $C_{\text{cell}}(\boldsymbol{x}_t)$ is the obstacle cost assigned to the cost map cell that corresponds to the robot pose at time t of the sample. In case $C_{\text{cell}}(\boldsymbol{x}_t) \geqq 99$, the robot is within the collision radius shown in Fig. 3, and an ∞ cost is set because the path will collide with an obstacle. On the other hand, in case $C_{\text{cell}}(\boldsymbol{x}_t) < 99$, the robot is outside the collision radius, so the cost assigned to the cost map is used as the cost.

In MPPI, random samples of control inputs $\boldsymbol{V}_k = [\boldsymbol{v}_0, \boldsymbol{v}_1, \ldots, \boldsymbol{v}_{T-1}]$, with Gaussian noise added up to time T, are sampled K times. For the K sample sequences $[\boldsymbol{V}_k]_{k=1}^{K}$ created, the cost $S(\boldsymbol{V}_k)$ of control inputs up to time T is calculated using the cost function defined in Eq. (4), as shown in Eq. (7).

$$S(\boldsymbol{V}_k) = \sum_{t=0}^{T} C(\boldsymbol{x}_t) \tag{7}$$

Then, the weight $\boldsymbol{w}_k$ for each sample $\boldsymbol{V}_k$ is calculated as shown in Eq. (8).

$$\boldsymbol{w}_k = \frac{1}{\eta} \exp\left(-\frac{1}{\lambda}\, S(\boldsymbol{V}_k) - \sum_{t=0}^{T-1} (\hat{\boldsymbol{u}}_t - \tilde{\boldsymbol{u}}_t)^{\mathsf{T}}\, \Sigma^{-1} \boldsymbol{v}_t\right) \tag{8}$$

Where η is the normalization term, λ is the temperature parameter, and Σ is the covariance matrix of the Gaussian distribution generating the random samples. Also, $\hat{\boldsymbol{U}}_t = [\hat{\boldsymbol{u}}_0, \hat{\boldsymbol{u}}_1, \ldots, \hat{\boldsymbol{u}}_{T-1}]$ is the initial estimated solution, and $\tilde{\boldsymbol{U}}_t = [\tilde{\boldsymbol{u}}_0, \tilde{\boldsymbol{u}}_1, \ldots, \tilde{\boldsymbol{u}}_{T-1}]$ is the nominal input sequence.

Finally, the optimal solution $\boldsymbol{U}_t{}^*$ is found using the weighted average as shown in Eq. (9).

$$\boldsymbol{U}_t{}^* = \sum_{k=1}^{K} \boldsymbol{w}_k \boldsymbol{V}_k \tag{9}$$

The implementation in this paper uses the prediction horizon $T = 0.02$ [s] $\times$ 50 [times] $= 1$ [s], the sample size $K = 2048$, the temperature parameter $\lambda = 0.2$, and the parameter $\sigma = 3.0$ for the sample covariance matrix Σ. Also, the initial estimated solution $\hat{\boldsymbol{U}}_t$ uses the previous optimal solution, and the nominal input sequence is $\tilde{\boldsymbol{U}}_t = 0$.

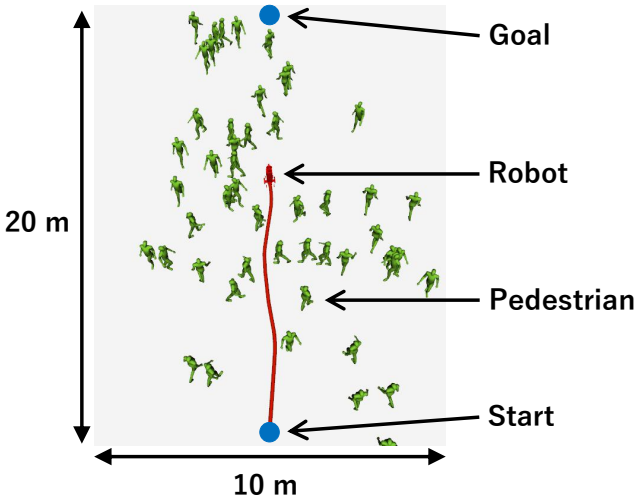


Fig. 5. Experimental environment: pedsim_ros [25] is used for a pedestrian simulator.

IV. Experiments

A. Experimental Conditions

We conducted pedestrian avoidance experiments in crowded environments using a simulator to evaluate the avoidance performance of the proposed method. We used pedsim_ros [25] for a pedestrian simulator. Pedestrians moved based on Social Force Model [26], and they took avoidance actions not only toward other pedestrians but also toward a robot. Also, the poses and velocity of pedestrians at each time were obtained true values from the simulator.

Fig. 5 shows the experimental environment. The number of pedestrians was 50, and the area within which pedestrians could move was set to 20 m in length and 10 m in width.

When creating the cost map, the weighting parameters in Eq. (2) were set to $\alpha = 0.5$ and $\beta = 0.5$. In addition, the maximum and minimum values of inflation radius in Eq. (2) and Eq. (3) were set to $l_{\text{max}} = 2.0$ [m], $l_{\text{min}} = 1.0$ [m], $s_{\text{max}} = 1.2$ [m], and $s_{\text{min}} = 0.7$ [m]. The weighting parameters of the cost function in Eq. (4) were set to $\gamma = 4.0$ and $\delta = 5.0$.

Fig. 6 shows cost maps created using each method. The types of cost maps are: 1. Collision Radius Only (Fig. 6 (a)), 2. Circular Inflation (Fig. 6 (b)), 3. Considering Velocity (Fig. 6 (c)). Also, the obstacle costs indicated in Fig. 6 were set for current pedestrian poses.

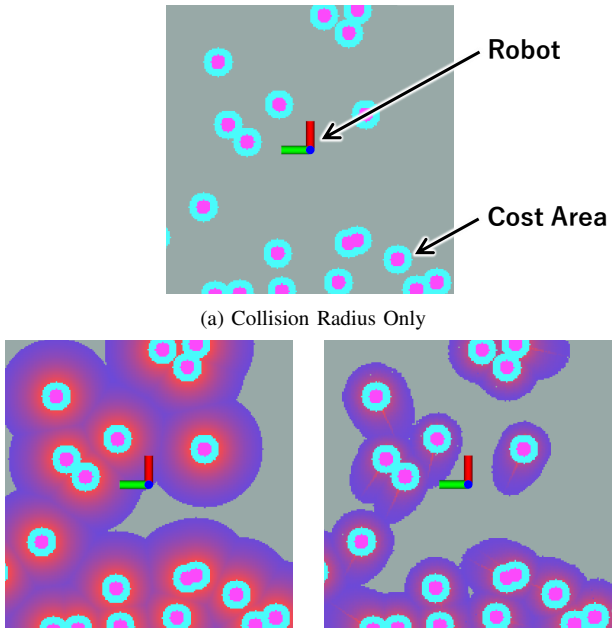
As methods for comparison in experiments, the types of cost maps and the uses of predicted poses are: 1. Collision Radius Only (Fig. 6 (a)) without predicted poses, 2. Circular Inflation (Fig. 6 (b)) without predicted poses, 3. Considering Velocity (Fig. 6 (c)) without predicted poses, 4. Considering Velocity (Fig. 6 (c)) with predicted poses (ours). In case 4. Considering Velocity with predicted poses, the shape of obstacle costs was the same as in Fig. 6 (c), but the cost

TABLE I

EVALUATION OF MOVING OBSTACLE AVOIDANCE FOR EACH COST MAP TYPE

Cost Map Type	SR [%] ↑	PL [m] ↓	NT [s] ↓	OD [m] ↑
Collision Radius Only	69	**20.1**	**18.7**	1.74
Circular Inflation	79	20.2	34.7	1.80
Considering Velocity (w/o Predicted Poses)	87	20.3	25.0	**1.81**
Considering Velocity (w/ Predicted Poses, Ours)	**94**	20.2	24.8	**1.81**

SR: Success Rate, PL: Path Length, NT: Navigation Time, OD: Obstacle Distance

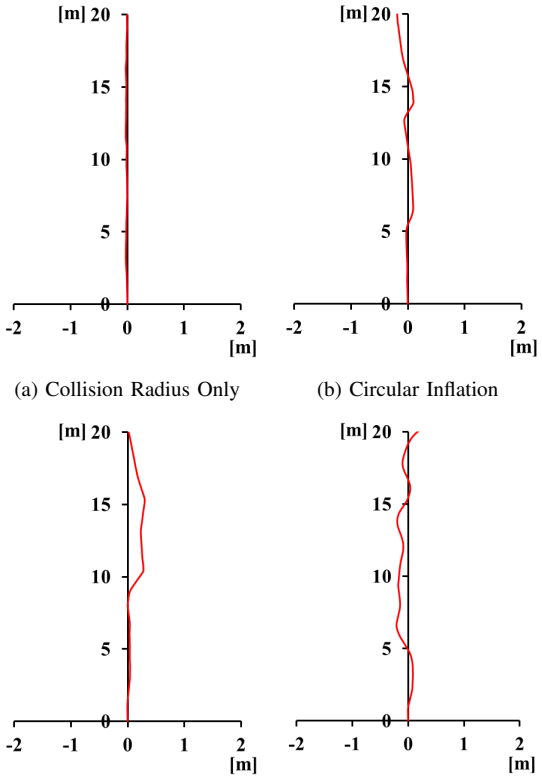


(a) Collision Radius Only

(b) Circular Inflation

(c) Considering Velocity (Ours)

Fig. 6. Cost maps created using each method. Obstacle cost colors correspond to the plot in Fig. 3.



(a) Collision Radius Only

(b) Circular Inflation

(c) Considering Velocity (w/o Predicted Poses)

(d) Considering Velocity (w/ Predicted Poses, Ours)

Fig. 7. Robot trajectories of each cost map type.

center was the predicted pose rather than the current pose. Additionally, in case 2. Circular Inflation, inflation radius was set to 2.0 m, the same as the maximum value of the proposed method.

B. Experimental Results

Experiments were conducted 100 times for each condition. The evaluation metrics were success rate (SR), path length (PL), navigation time (NT), and average obstacle distance to the closest pedestrian (OD). A collision was defined as occurring when the distance between the pedestrian and the robot was less than the collision radius. Any result involving a collision with a pedestrian was considered a failure. For items other than the success rate, the average value was calculated only for successful results.

Table I shows evaluation results of moving obstacle avoidance for each cost map type. In addition, Fig. 7 shows an example of robot trajectories of each cost map type.

Our proposed method achieved the highest success rate (SR) among the four methods. In addition, focusing on the average obstacle distance (OD), all three methods with inflated obstacle costs, except for Collision Radius Only, kept similar distances. Among these three methods, focusing on the path length (PL) and the navigation time (NT), our method reduced both path length and navigation time. On the other hand, comparing the proposed method with Collision Radius Only, both path length and navigation time of ours were increased. However, this is a result of the robot taking avoidance actions against pedestrians. Comparing Fig. 7 (a) and Fig. 7 (d) also indicates that Collision Radius Only takes almost no avoidance action, while our proposed method takes avoidance action.

V. CONCLUSION

In this paper, we aimed to realize stable avoidance behavior for moving obstacles, and proposed a motion planning method using MPPI based on obstacle cost maps. The proposed method created cost maps considering velocity and predicted poses of moving obstacles and formulated cost functions of MPPI that appropriately avoid obstacles. Furthermore, we conducted pedestrian avoidance experiments in

crowded environments using a simulator, and demonstrated the usefulness of the proposed method for moving obstacle avoidance. In the future, we plan to conduct further research to improve avoidance performance and verify it in the real world.

ACKNOWLEDGMENT

This study was conducted under Research Cluster for Autonomous Robotic Systems at Meiji University. We sincerely thank you for this support.

REFERENCES

[1] Grady Williams, Paul Drews, Brian Goldfain, James M. Rehg, and Evangelos A. Theodorou: "Aggressive Driving with Model Predictive Path Integral Control", *Proc. of IEEE Int. Conf. on Robotics and Automation (ICRA)*, 2016.

[2] Dieter Fox, Wolfram Burgard, and Sebastian Thrun: "The Dynamic Window Approach to Collision Avoidance", *IEEE Robotics and Automation Magazine*, vol. 4, no. 1, pp. 23–33, 1997.

[3] Johann Borenstein and Yoram Koren: "The Vector Field Histogram: Fast Obstacle Avoidance for Mobile Robots", *IEEE Trans. on Robotics and Automation*, vol. 7, no. 3, pp. 278–288, 1991.

[4] Oussama Khatib: "Real-Time Obstacle Avoidance for Manipulators and Mobile Robots", *Proc. of IEEE Int. Conf. on Robotics and Automation (ICRA)*, 1985.

[5] Vinu Maddumage, Sarath Kodagoda, Marc G. Carmichael, Amal Gunatilake, Karthick Thiyagarajan, and Jodi Martin: "Relative Velocity-based Reward Model for Socially-aware Navigation with Deep Reinforcement Learning", *Proc. of IEEE Int. Conf. on Robotics and Automation (ICRA)*, 2025.

[6] Asako Kanezaki, Jirou Nitta, and Yoko 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.

[7] Kohei Matsumoto, Yuki Hyodo, and Ryo Kurazume: "Crowd-aware Robot Navigation with Switching between Learning-based and Rule-based Methods using Normalizing Flows", *Proc. of IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2024.

[8] Mai Nishimura and Ryo Yonetani: "L2B: Learning to Balance the Safety-efficiency Trade-off in Interactive Crowd-aware Robot Navigation", *Proc. of IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2020.

[9] David V. Lu, Dave Hershberger, and William D. Smart: "Layered Costmaps for Context-sensitive Navigation", *Proc. of IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2014.

[10] Chin-Sheng Chen and Si-Yu Lin: "Costmap Generation based on Dynamic Obstacle Detection and Velocity Obstacle Estimation for Autonomous Mobile Robot", *Proc. of Int. Conf. on Control, Automation and Systems (ICCAS)*, 2021.

[11] David Wilkie, Jur van den Berg, and Dinesh Manocha: "Generalized Velocity Obstacles", *Proc. of IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2009.

[12] Zhiqiang Jian, Songyi Zhang, Lingfeng Sun, Wei Zhan, Nanning Zheng, and Masayoshi Tomizuka: "Long-term Dynamic Window Approach for Kinodynamic Local Planning in Static and Crowd Environments", *IEEE Robotics and Automation Letters*, vol. 8, no. 6, pp. 3294–3301, 2023.

[13] Steve Macenski, Francisco Martín, Ruffin White, and Jonatan G. Clavero: "The Marathon 2: A Navigation System", *Proc. of IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2020.

[14] Steve Macenski, Tom Moore, David V. Lu, Alexey Merzlyakov, and Michael Ferguson: "From the Desks of ROS Maintainers: A Survey of Modern & Capable Mobile Robotics Algorithms in the Robot Operating System 2", *Robotics and Autonomous Systems*, vol. 168, pp. 1–22, 2023.

[15] Kohei Honda, Naoki Akai, Kosuke Suzuki, Mizuho Aoki, Hirotaka Hosogaya, and Hiroyuki Okuda: "Stein Variational Guided Model Predictive Path Integral Control: Proposal and Experiments with Fast Maneuvering Vehicles", *Proc. of IEEE Int. Conf. on Robotics and Automation (ICRA)*, 2024.

[16] Takato Miura, Naoki Akai, Kohei Honda, and Susumu Hara: "Spline-Interpolated Model Predictive Path Integral Control with Stein Variational Inference for Reactive Navigation", *Proc. of IEEE Int. Conf. on Robotics and Automation (ICRA)*, 2024.

[17] Qiang Liu and Dilin Wang: "Stein Variational Gradient Descent: A General Purpose Bayesian Inference Algorithm", *Proc. of Int. Conf. on Neural Information Processing Systems (NIPS)*, 2016.

[18] Mizuho Aoki, Kohei Honda, Hiroyuki Okuda, and Tatsuya Suzuki: "Switching Sampling Space of Model Predictive Path-Integral Controller to Balance Efficiency and Safety in 4WIDS Vehicle Navigation", *Proc. of IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2024.

[19] Aamodh Suresh and Carlos Nieto-Granda: "Reactive Robot Navigation using Behavioral Risk Perception for Uncertain Dynamic Obstacles", *Proc. of Int. Conf. on Ubiquitous Robots (UR)*, 2024.

[20] Ihab S. Mohamed, Mahmoud Ali, and Lantao Liu: "Chance-Constrained Sampling-based MPC for Collision Avoidance in Uncertain Dynamic Environments", *IEEE Robotics and Automation Letters*, vol. 10, no. 7, pp. 7492–7499, 2025.

[21] Christoph Schöller, Vincent Aravantinos, Florian Lay, and Alois Knoll: "What the Constant Velocity Model Can Teach Us about Pedestrian Motion Prediction", *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 1696–1703, 2020.

[22] Bogdan Vlahov, Jason Gibson, Manan Gandhi, and Evangelos A. Theodorou: "MPPI-Generic: A CUDA Library for Stochastic Trajectory Optimization", *arXiv:2409.07563*, 2024.

[23] mppi_playground. `https://github.com/kohonda/mppi_playground`

[24] python_simple_mppi. `https://github.com/MizuhoAOKI/python_simple_mppi`

[25] pedsim_ros. `https://github.com/srl-freiburg/pedsim_ros`

[26] Dirk Helbing and Péter Molnár: "Social Force Model for Pedestrian Dynamics", *Physical Review E*, vol. 51, no. 5, pp. 4282–4286, 1995.