# Comparison of End-to-End Continuous and Discrete Motion Planners for Autonomous Mobile Robots

Satoshi HOSHINO[1†] and Joichiro SUMIYOSHI[1]

[1]Department of Mechanical and Intelligent Engineering Graduate School of Engineering Utsunomiya University, Japan
(Tel: +81-28-689-6053; E-mail: hosino@cc.utsunomiya-u.ac.jp)

**Abstract:** For autonomous navigation of mobile robots, obstacle avoidance capability and straight-line stability are essential performance. Robots are required to move with suitable control values regarding velocity and angular velocity. Therefore, we have focused on motion planning techniques and proposed end-to-end discrete motion planners. In one of the motion planners, a deep neural network, DNN, was used to classify inputs into motion commands, such as straight, right, and left. In environments with obstacles, the robot based on the motion planner successfully moved with discretized control values depending on the motion commands. In this paper, we further present an end-to-end continuous motion planner that allows the robot to move with continuous control values. In the navigation experiments, these two motion planners are applied to the robot, and the navigation performance is discussed.

**Keywords:** Mobile Robots, Motion Planning, End-to-End, Autonomous Navigation, Obstacle Avoidance.

## 1. INTRODUCTION

For autonomous navigation of mobile robots, obstacle avoidance capability and straight-line stability are essential performance. Robots are required to move with suitable control values regarding velocity and angular velocity. In this paper, therefore, we focus on motion planning techniques. Previously, artificial potential method [1], dynamic window approach [2], and other motion planning techniques have been proposed. Since mathematical models were used to decide the continuous control values for sensor inputs, these were so-called model-based motion planners. Mobile robots based on the model-based motion planners are, however, sometimes exhibit different behavior depending directly on sensor inputs. Furthermore, it is required to determine the model parameters suitable to navigation environments.

For the drawbacks of the model-based planners, motion planners based on deep learning [3] have attracted attention. The motion planners aim to approximate a relationship between sensor inputs and motor outputs with the use of deep neural networks, DNNs, instead of the mathematical models. These are known as end-to-end motion planners. Compared to model-based motion planners, end-to-end motion planners are less affected by sensor noises, and model parameters are not used. In general, DNNs and convolutional neural networks, CNNs, have been used for regression and classification problems.

In the regression problem, an end-to-end motion planner based on CNN has successfully decided the continuous control value of angular velocity for the image input [4]. Furthermore, a target direction toward a destination has been used as the input in addition to the camera images [5][6]. This motion planner enabled the robot to move along a planned path toward the destination. Instead of camera sensors, 2D LiDARs have been used to obtain range data. The input range data and target direction enabled a robot to move toward the destination while avoiding obstacles [7].

However, the DNNs and CNNs used in the regression problem require a large amount of data for network training. As a result, such data sets increase in the training cost. On the other hand, we have proposed an end-to-end discrete motion planner based on a DNN [8] for the classification problem. **Fig. 1** shows our mobile robot equipped with a 2D LiDAR.


Fig. 1 Mobile robot equipped with 2D LiDAR

In the motion planner, the DNN was used to classify the input range data and target direction into the motion commands, such as straight, right, and left. After that, discrete control values of the velocity and angular velocity depending on the classified motion commands were decided and sent to the robot. **Fig. 2** shows test environments given for autonomous navigation.

(a)Random obstacles (b)Dense obstacles (c)Dead end

Fig. 2 Corridor environments with obstacles

† Satoshi HOSHINO is the presenter of this paper.

According to the discretized control values, the robot based on the end-to-end motion planner successfully moved toward the destination while avoiding the obstacles as shown in Fig. 2(a) and Fig. 2(b). In order to take into account a series of motions for a dead end as shown in Fig. 2(c), moreover, we have proposed another end-to-end discrete motion planner based on a deep recurrent neural network, DRNN [9]. The motion planner enabled the robot to stop at the dead end, rotate at the same position, return to the doorway, and move on the right side of the corridor.

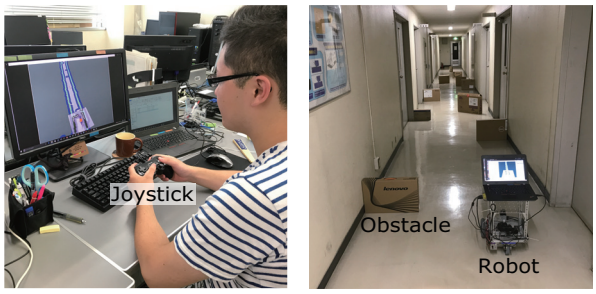
Compared to continuous motion planners for the regression problem, the discrete motion planners require lower training costs. In contrast to the advantage, the navigation performance of the discrete motion planners is a challenging issue. In this paper, therefore, we present continuous and discrete motion planners based on DNNs. For the input, these motion planners decide continuous and discrete control values of the velocity and angular velocity. In the navigation experiments, these two motion planners are applied to the robot. As the navigation performance, obstacle avoidance capability and straight-line stability of the robot are discussed.

## 2. LEARNING FROM DEMONSTRATION

### 2.1. Teaching System

For end-to-end motion planners based on DNNs, we adopt a learning-from-demonstration approach. In this approach, a mobile robot is first controlled by an operator. In so doing, the sensor data and the corresponding control values given by the operator are recorded as the input and output of the DNNs. Eventually, the inputs and outputs are acquired as data sets. After the teaching phase, in the learning phase, the data sets are used for network training.

In the teaching phase, if the operator was just behind the robot, the robot is controlled with unobtainable data by the sensor. In the learning phase, such data sets might be inadequate for network training. For the gap of environmental information between the operator and robot, we have developed a teaching system as shown in **Fig. 3**. In this system, both the operator and robot were allowed to have the same environmental information.




(a)Operator with joystick (b)Controlled robot by operator

Fig. 3 Developed teaching system for learning from demonstration

Since the robot is equipped with the 2D LiDAR, a map of environment is built beforehand, and the robot is allowed to localize the position and orientation in this map. The information is then shared with the operator. In Fig. 3(a), the scanned range data and the localized position of the robot in the occupancy grid map are displayed on the monitor. In Fig. 3(b), the robot based on the human operation is moving forward. In the corridor, obstacles are newly arranged. When the obstacles are detected by the robot, the range data is also denoted in the map and displayed on the monitor. In the teaching phase, the operator in front of the monitor controls the velocity $v$ and angular velocity $\omega$ with the left and right joysticks so that the robot moves while avoiding the obstacles.

### 2.2. Input Data

The moving robot based on the human operation simultaneously records the range data as the input in the training data sets. **Fig. 4** illustrates the scanned range data. In this paper, we assume that the 2D LiDAR has limited scan angle, 180 [deg], and sensing area.

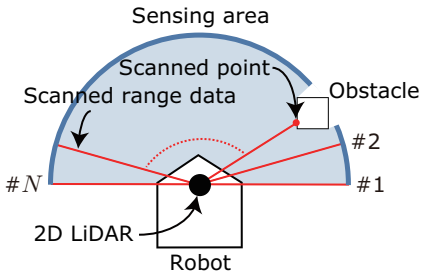
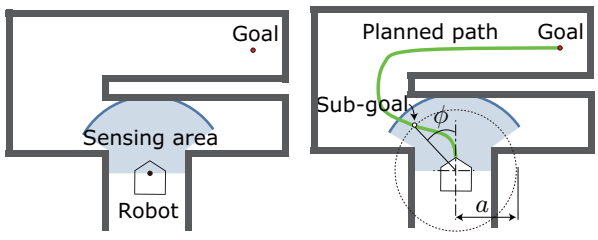



Fig. 4 Top view of scanned range data by 2D LiDAR

Each time the robot scans the environment, $N$(1, 2, $\cdots$, $N$) range data are obtained. If there was an obstacle within the sensing area, a distance between the 2D LiDAR and scanned point of the obstacle is returned as the range data. The range data enables the robot to avoid the obstacle. However, the robot based on this motion might move in a different direction from the destination. Therefore, we use a direction angle as an additional input in the training data sets.

As described in 2.1, the robot localizes the position and orientation in a global map of environment. In this map, a destination is designated by an operator. Thus a direction angle, $\phi$, is derived from the robot and destination. **Fig. 5** illustrates the direction angle.




(a)Scanned range data (b)Sub-goal

Fig. 5 Definition of direction angle between robot and sub-goal

In Fig. 5(a), the robot is allowed to turn right and left according to the symmetrical input range data. In this

case, if a direction angle between the robot and goal destination was used as the additional input, the robot turns right. This is because that the robot moves so as to reduce the direction angle. Therefore, we use a sub-goal to derive the direction angle. In Fig. 5(b), the robot is planning a path toward the goal destination. Furthermore, a dashed circle centered at the robot with a radius of $a$ is drawn. Then, a sub-goal is arranged at the intersection between the planned path and dashed circle. Finally, a direction angle between the robot and sub-goal, $\phi$, is derived. By using the direction angle, $\phi$, as the additional input, the robot is allowed to turn left toward the sub-goal and move toward the goal destination.

### 2.3. Output Data

The operator controls the velocity $v$ and angular velocity $\omega$ of the robot through the joysticks. These control values, $v$ and $\omega$, are sent to the robot for the human operation. Therefore, the moving robot based on the human operation simultaneously records the velocity and angular velocity as the output in the training data sets.

For the continuous motion planner, these continuous values are the output data. For the discrete motion planner, the continuous values of $v$ and $\omega$ are discretized into the three motion commands depending on the combinations as follows:

- if $v > 0$ and $\omega = 0$, then "straight,"
- else if $v > 0$ and $\omega < 0$, then "right," and
- else if $v > 0$ and $\omega > 0$, then "left."

Instead of $v$ and $\omega$, the motion commands are the output data.

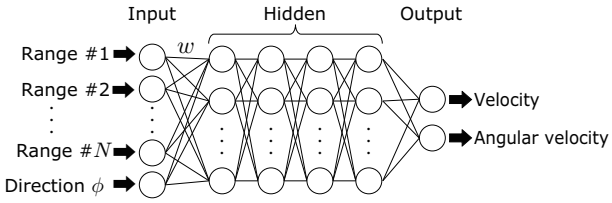
Finally, the training data sets are composed of the input and output data at the same sampling time. In the learning phase, the output data, which are the continuous values of $v$ and $\omega$ and the motion commands, are used as the supervisory signals for network training.

## 3. END-TO-END MOTION PLANNERS

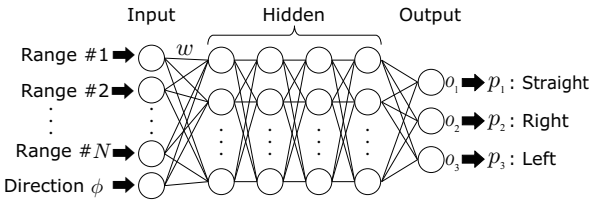
### 3.1. Deep Neural Networks

From DNNs, continuous values and discrete values are derived as the control values of the velocity and angular velocity. These control values are sent to the robot for autonomous navigation. In order to decide the suitable control values for the regression and classification problems, we present two motion planners based on DNNs as illustrated in **Fig. 6**.

The $N$ range data obtained from the 2D LiDAR and direction angle $\phi$ are the input to both the DNNs. Since the 2D LiDAR shown in Fig. 1 has $N$ range data, the input layer is composed of $N + 1$ units. In addition, both of the DNNs are composed of four hidden layers. In Fig. 6(a), since the control targets are the velocity and angular velocity, the output layer is composed of two units. In Fig. 6(b), the output layer is composed of three units corresponding to the number of motion commands. All the units in each layer are fully connected to the ones in the following layer through the connection weight $w$. In each unit, the rectifier is used as an activation function.




(a)Continuous motion planner




(b)Discrete motion planner

Fig. 6 DNNs used in the motion planners

In Fig. 6(a), the continuous values of the velocity $v$ and angular velocity $\omega$ are decided. Thus, $v$ and $\omega$ are directly sent to the robot. In Fig. 6(b), on the other hand, each unit in the output layer computes $o_i$, and $o_i$ is converted into a probability, $p_i$, by the softmax function. As a result, the input data is classified into a motion command with the highest probability. Finally, the velocity $v$ and angular velocity $\omega$ discretized depending on the motion command, i.e. straight, right, or left, are decided and sent to the robot.

### 3.2. Training Data Sets

In the teaching phase, training data sets are acquired as described in 2. The mobile robot as shown in Fig. 1 is used. The robot is equipped with the 2D LiDAR at a height of 0.2 [m]. The sensing range of the 2D LiDAR is 4.0 [m]. The robot motion is controlled by an operator in three environments as shown in **Fig. 7**. The mark "S" indicates the initial position of the robot and "G" is the destination.


(a)1st environment (b)2nd environment (c)3rd environment

Fig. 7 Teaching environments with tables and chairs as obstacles

The number of obstacles and these positions are different in each environment. In Fig. 7(a) and Fig. 7(b), the robot based on the human operation moves toward the destinations while avoiding the obstacles. In Fig. 7(c), since the obstacles are located against the wall, the robot just moves straight toward the destination.

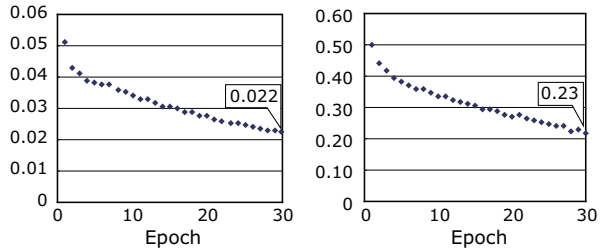
In order for the robot to localize the position and orien-

tation, Monte Carlo Localization, MCL [10], is applied. As for the direction angle, Dijkstra's algorithm [11] is used as a path planner in Fig. 5(b). During the robot is moving, the path planning is constantly executed. Finally, 3000 data sets were acquired.

### 3.3. Network Training

For each input in the data sets, the network training of the continuous motion planner shown in Fig. 6(a) is done by comparing the derived velocity and angular velocity with the supervisory signals. The network training of the discrete motion planner shown in Fig. 6(b) is also done by comparing the classified motion commands with the supervisory signals. The mean square error is used for the continuous motion planner and the cross-entropy is used for the discrete motion planner. In backpropagation, the Adam optimizer [12] is used to update each connection weight $w$. Furthermore, the units in the last hidden layer are dropped out with a probability of 0.5 to reduce overfitting.

In consideration of the LiDAR resolution, the range data in Fig. 4 is given $N = 512$. Thus the input layer in Fig. 6 is composed of 513 units in total. These input data are then compressed down into 500, 200, 100, and 50 units in the hidden layers. **Fig. 8** shows the result of the network training in the learning phase.




(a)Continuous motion planner (b)Discrete motion planner

Fig. 8 Loss transition of DNN for 3000 data sets

As the network training proceeded, the loss value of each DNN was decreased. This result indicates that the DNNs increased the regression and classification accuracies through the learning-from-demonstration approach. Since the loss values at the 30th epoch were minimized, 0.022 and 0.23, the DNNs based on these results are used as the end-to-end continuous and discrete motion planners for autonomous navigation of the robot.

## 4. NAVIGATION EXPERIMENTS

### 4.1. Settings

As well as the learning phase in 3, the same mobile robot equipped with the 2D LiDAR as shown in Fig. 1 is used in this experiment. For obstacle avoidance, the following two end-to-end motion planners based on the DNNs at the 30th epoch in Fig. 8 are applied to the robot:

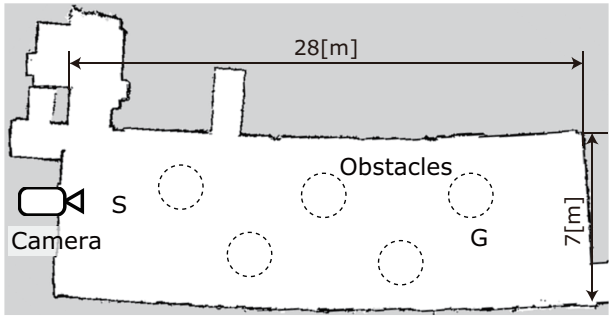
1. continuous motion planner, CMP, — Fig. 6(a) and
2. discrete motion planner, DMP, — Fig. 6(b).

Both the motion planners use the 512 range data and direction angle $\phi$ as the input. The sub-goal in Fig. 5(b) is arranged by the dashed circle with a radius of $a = 1.0$ [m]. Since the CMP decides the continuous control values of the velocity $v$ and angular velocity $\omega$, $v$ [m/s] and $\omega$ [rad/s] are directly sent to the robot. The DMP classifies the input data into any one of the straight, right, or left command. After that, the discretized velocity $v$ and angular velocity $\omega$ depending on the motion commands are decided as follows:

- if "straight," then $v = 0.4$ and $\omega = 0$,
- else if "right," then $v = 0.4$ and $\omega = -0.8$, and
- else if "left," then $v = 0.4$ and $\omega = 0.8$.

These control values of $v$ [m/s] and $\omega$ [rad/s] are then sent to the robot.

**Fig. 9** shows the target environment. While the layout is similar to the ones in Fig. 7(a) and Fig. 7(b), more obstacles are located at different positions in this environment.




(a)Occupancy grid map with newly-added obstacles

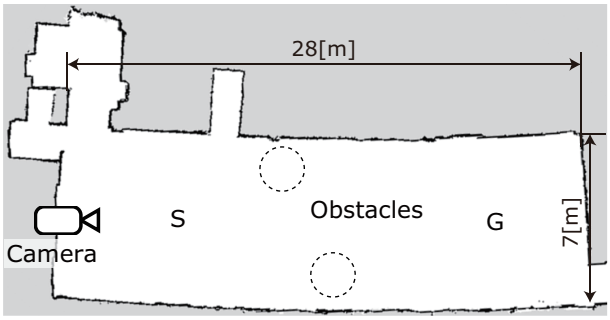

(b)Scene of actual environment taken from "camera" position in Fig. 9(a)

Fig. 9 Navigation environment 1

Fig. 9(a) shows an occupancy grid map of the environment. The obstacles drawn by five dashed circles were not arranged when the robot built the map beforehand. In this map, the robot moves from S toward G. Fig. 9(b) shows the actual environment with the obstacles. In this environment, the robot is required to avoid the obstacles to reach the destination.

**Fig. 10** shows another target environment. As well as the environment shown in Fig. 7(c), the obstacles are located against the wall. Compared to environment 1 shown in Fig. 9, the robot is allowed to reach the destination without avoiding the obstacles in this environment.

Fig. 10(a) shows an occupancy grid map of the environment. The obstacles drawn by two dashed circles were not arranged when the robot built the map beforehand. In this map, the robot moves from S toward G. Fig. 10(b) shows the actual environment with the obstacles.




(a)Occupancy grid map with newly-added obstacles


(b)Scene of actual environment taken from "camera" position in Fig. 10(a)

Fig. 10 Experimental environment 2

### 4.2. Navigation Results

As the navigation results of the robot based on the CMP and DMP, we focus on a series of the localized positions, which is the trajectory. In **Fig. 11**, the trajectories in environment 1 depending on the motion planners are compared. The dashed and solid lines denote the results of the CMP and DMP.

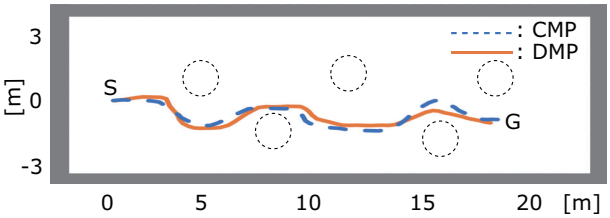
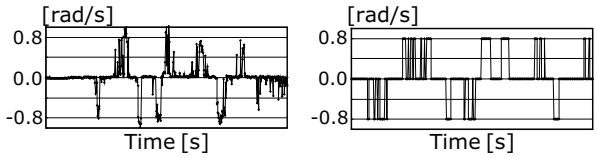



Fig. 11 Comparison of navigation trajectories in environment 1

The robot based on both the CMP and DMP succeeded in moving toward the destination, G, while avoiding the obstacles. In this regard, the robot moved with about the same velocity $v = 0.4$ and spent almost the same amount of time for navigation. For obstacle avoidance, the angular velocity plays an important role more than the velocity. Therefore, we compare the control values based on the CMP and DMP as shown in **Fig. 12**.

While the continuous value of angular velocity was decided in the CMP (see Fig. 12(a)), the discrete value of $-0.8$, $0$, or $0.8$, was decided in the DMP (see Fig. 12(b)). These angular velocities were sent to the robot as the control values. This result indicates that, compared to the DMP, the robot based on the CMP achieved more fluid obstacle avoidance. On the other hand, it is notable that




(a)CMP (b)DMP

Fig. 12 Angular velocity $\omega$ in environment 1

even the robot based on the DMP avoided the obstacles and reached the destination with the discretized angular velocity. Therefore, the sufficient obstacle avoidance capability of the two end-to-end motion planners regardless of the regression and classification problems was shown from the results.

In **Fig. 13**, the trajectories of the robot in environment 2 depending on the motion planners are compared. The dashed and solid lines denote the results of the CMP and DMP.

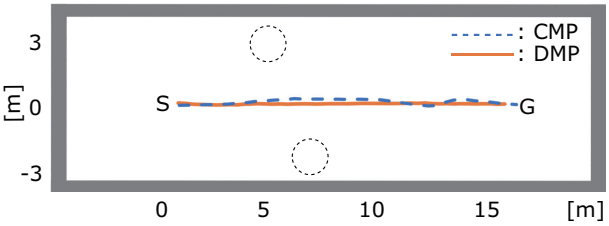
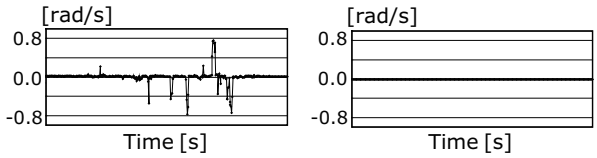



Fig. 13 Comparison of navigation trajectories in environment 2

The robot based on both the CMP and DMP succeeded in moving toward the destination, G. In this regard, however, the robot based on the CMP slightly weaved. On the other hand, the robot based on the DMP moved in a straight line. The robot behavior is due to the angular velocity. Therefore, we compare the control values as shown in **Fig. 14**.



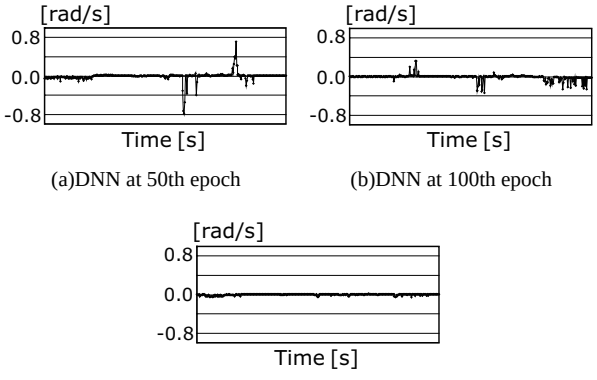

(a)CMP (b)DMP

Fig. 14 Angular velocity $\omega$ in environment 2

As can be seen in Fig. 14(a), although the obstacle avoidance was not required for the robot in this environment, the angular velocity other than $\omega = 0$ [rad/s] was decided in the CMP. This result indicates that the robot was affected by the obstacles within the sensing area even though these were located against the wall. In contrast to the result, the angular velocity of $\omega = 0$ [rad/s] was constantly decided in the DMP as shown in Fig. 14(b). In other words, as the result of the discretized angular velocity, the robot based on the DMP moved with no influence of the obstacles. Therefore, while the CMP resulted in the fluid motion for obstacle avoidance, the higher straight-line stability of the DMP was shown.

### 4.3. Straight-Line Stability

In order to improve the straight-line stability of the CMP, the network training was further conducted. Afterward, the CMP based on the enhanced DNN was applied to the robot. The robot based on the CMP moved in environment 2 as shown in Fig. 10, and successfully reached the destination, G. **Fig. 15** shows the angular velocities based on three DNNs.




(a)DNN at 50th epoch (b)DNN at 100th epoch

(c)DNN at 200th epoch

Fig. 15 Angular velocity $\omega$ of CMP in environment 2

In Fig. 15(a), even though the DNN at the 50th epoch was used in the CMP, the angular velocity other than $\omega = 0$ [rad/s] was decided and sent to the robot. The average absolute value was $0.031$ [rad/s]. In Fig. 15(b), the average absolute value based on the DNN at the 100th epoch was reduced to $0.027$ [rad/s]. However, the angular velocity other than $\omega = 0$ [rad/s] was still decided and sent to the robot. As a result, the robot based on these CMPs slightly weaved in some cases. In Fig. 15(c), finally, the average absolute value based on the DNN at the 200th epoch was reduced to $0.008$ [rad/s]. Thus the robot based on the CMP was enabled to move in a straight line toward the destination.

From the results described above, it was shown that, as the network training further proceeded, the straight-line stability of the CMP approached the one of DMP. In this regard, however, while the DMP was based on the DNN at the 30th epoch, the CMP was based on the DNN at the 200th epoch. This result indicates the effectiveness of the discrete motion planner for autonomous navigation of mobile robots at the lowest possible training cost.

## 5. CONCLUSIONS

In this paper, two end-to-end continuous and discrete motion planners based on DNNs were presented for autonomous navigation of mobile robots. In these motion planners, continuous and discrete velocity and angular velocity were decided and sent to the robot as the control values. Thus, the robot was allowed to successfully move toward the destination while avoiding obstacles. In the navigation experiments, the sufficient obstacle avoidance capability of the two end-to-end motion planners was shown regardless of the regression and classification problems. Moreover, while the continuous motion planner resulted in the fluid motion for obstacle avoidance, the robot slightly weaved due to obstacles even though these were located against the wall. On the other hand, the robot based on the discrete motion planner completely moved in a straight line. Therefore, the higher straight-line stability of the discrete motion planner was shown. Although the straight-line stability of the continuous motion planner was improved with the further enhanced DNNs through the network training, the effectiveness of the discrete motion planner for autonomous navigation of mobile robots at the lowest possible training cost was shown.

## REFERENCES


[1] O. Khatib, "Real-Time Obstacle Avoidance for Manipulators and Mobile Robots," *International Journal of Robotics Research*, Vol. 5, No.1, pp. 90–98, 1986.

[2] W. Burgard *et al.*, "The Dynamic Window Approach to Collision Avoidance," *IEEE Robotics and Automation Magazine*, Vol. 4, No. 1, pp. 23–33, 1997.

[3] G. E. Hinton *et al.*, "A Fast Learning Algorithm for Deep Belief Nets," *Neural Computation*, Vol. 18, No. 7, pp. 1527–1554, 2006.

[4] Y. LeCun *et al.*, "Off-Road Obstacle Avoidance through End-to-End Learning," *Advances in Neural Information Processing Systems*, pp. 739–746, 2005.

[5] A. Carballo *et al.*, "End-to-End Autonomous Mobile Robot Navigation with Model-Based System Support," *Journal of Robotics and Mechatronics*, Vol. 30, No. 4, pp. 563–583, 2018.

[6] S. Seiya *et al.*, "End-to-End Navigation with Branch Turning Support Using Convolutional Neural Network," *IEEE International Conference on Robotics and Biomimetics*, pp. 499–506, 2018.

[7] M. Pfeiffer *et al.*, "From Perception to Decision: A Data-Driven Approach to End-to-End Motion Planning for Autonomous Ground Robots," *IEEE International Conference on Robotics and Automation*, pp. 1527–1533, 2017.

[8] S. Hoshino and J. Sumiyoshi, "End-to-End Discrete Motion Planner based on Deep Neural Network for Autonomous Mobile Robots," *IEEE/SICE International Symposium on System Integration*, pp. 12–17, 2019.

[9] S. Hoshino and J. Sumiyoshi, "End-to-End Discrete Motion Planner based on Deep Recurrent Neural Network for Mobile Robots," *International Conference on Intelligent Autonomous Systems*, 2020 (under review).

[10] D. Fox *et al.*, "Robust Monte Carlo Localization for Mobile Robots," *Artificial Intelligence*, Vol. 128, No. 1-2, pp. 99–141, 2001.

[11] E.W. Dijkstra, "A Note on Two Problems in Connexion with Graphs," *Numerische Mathematik*, pp. 269–271, 1959.

[12] D. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization," *International Conference for Learning Representations*, 2015.