Autonomous Robot Navigation System with Learning Based on Deep Q-Network and Topological Maps

Yuki Kato[1], Koji Kamiyama[1] and Kazuyuki Morioka[1]

Abstract— **This paper proposes a autonomous mobile robot navigation system integrating local navigation based on deep reinforcement learning and global navigation based on topological maps. Especially, the proposed system aims to achieve a robot navigation that is adaptive with human traffic in environments crowded with many people. In order to obtain abilities for adapting dynamic obstacles in such environments, a simulator imitating real worlds is required for learning. Then a simulator suitable for learning of robot navigation in crowded environments is developed at first. This simulator is specialized in simulation of two-dimensional shape, and it could achieve learning in shorter time than existing simulators. Next, a learning system based on DDQN was designed for local navigation with avoiding pedestrians. Some simulations and experiments were performed to present effectiveness of the proposed navigation system. Simulation results show that the system learned abilities to go to local destinations with avoiding dynamic obstacles Also, the system was applied to an actual robot system and navigation in a real world was demonstrated.**

I. Introduction

Recently, intelligent robots are expected to be active in not only industrial fields but also various environments coexisting with people such as living support, sports, medical welfare and so on. Navigation of robots is a common problem in these applications. Mainly, localization and path planning must be included in the robot navigation. Grid map-based navigation is widely known as one of effective ways. The grid maps are generally build by using SLAM algorithm [1]. However, building occupancy grid maps in a real world is so cumbersome work. A robot system that don't depend on such precise grid maps are desired. This study aims to propose a navigation system without grid maps. Topological map-based global navigation and deep reinforcement learning-based local navigation are integrated in the proposed system.

Generally, topological map have only geometric positional relationships among waypoints. Such maps are easier to build than occupancy grid maps. Additionally, we apply deep reinforcement learning for local path planning to travel between waypoints. Deep reinforcement learning is a method approximating action value function using neural networks. It has achieved great success in task of playing computer games. This learning method can provide a solution in the problems that a series of successive behaviors is evaluated. Applying deep reinforcement learning to autonomous mobile robot navigation has also great potential.

Also, map-based navigation is not effective in environments crowded with many people. As one of the local path planning methods, DWA (Dynamic Window Approach) [2] is widely known. DWA selects a path based on a cost map representing local obstacles and a motion model of robot. Smooth navigation in crowded environments is almost impossible. Acquiring navigation skills with deep reinforcement learning have a possibility to handle this problem. In this paper, we aim to develop a mobile robot system that includes local navigation by learning behaviors like not only avoiding collision with obstacles but also coordinated with pedestrians. Learning of local navigation, simulation of the method integrating global and local navigations and experiments in a real world are also described.

[1]Meiji University, Graduate School of Advanced Mathematical Sciences, Network Design Program, Japan cs173023@meiji.ac.jp

II. Related Work

A. Autonomous Mobile Robot Navigation based on Deep Reinforcement Learning

There are mainly two kinds of state representations for autonomous mobile robot navigation method based on deep reinforcement learning. One representation is depth image and the other is range data by a laser range finder (LRF).

Zhang *et al.* [3] achieved a navigation system which dosen't need localization, mapping and path planning. They used depth images acquired from Microsoft Kinect as robot states, and Deep Q-Network(DQN) [4] was used to learn strategy moving to the destination. In addition, they proposed successor-feature-based reinforcement learning which adapts quickly to new environments using already learned results.

Tai *et al.* [5] developed a LRF based navigation system. As a learning algorithm, they adopt ADDPG (Asynchronous Deep Deterministic Policy Gradient) that was an improved version of DDPG [6]. ADDPG uses asynchronous data acquired from multiple environments for learning. In this study, they used range data scanned by the laser range finder mounted on the robot. A robot state is a set of 10 range points decimated from raw range points. Robot navigation experiments using learning results were performed in an actual environment. Also, "move-base" that is a well-known navigation method was compared with the proposed method. The experimental results showed that the learning-based system could navigate the robot in real world. On the other hand, "move-base" could not achieve the same tasks because 10 range points are too sparse.

In above studies, robot navigation in static environments were achieved. However, avoidance of dynamic obstacles such as pedestrians has not been mentioned. Our proposed system clearly includes learning of dynamic obstacle avoidance behaviors.

978-1-5386-2263-6/17/$31.00 ©2017 IEEE

B. *Topological Map Based Navigation*

Occupancy grid map based navigation can provide precise localization and path planning. However, grid map based navigation has several problems. Building precise grid maps of wide environments is time-consuming. Also, it is not robust to changes in the shape of environments. For solving these problems, topological map based navigation has been studied actively.

Watanabe $et\ al.$ [7] developed a topological map and road following based navigation system in general outdoor environments. They decided nodes of topological map in advance. Intersections that robots can recognize easily are used as nodes. Navigation was achieved by iterating road following and recognition of nodes by pattern matching. The navigation method is implemented to an actual mobile robot. The robot succeeded in traveling up to 900[m] in an actual outdoor environment.

Cheng $et\ al.$ [8] developed a system that a robot automatically explores indoor environments and generate topological maps including loop closing. Also, in the proposed system Progress Bayesian classifier is applied to recognition of nodes. That achieved high robustness against noise. Since it is a navigation system based on an automatically created topological map, advance preparation cost is very low.

In the topological map-based navigation, traveling between nodes and recognition of nodes are the core of the system. We focus on the traveling between nodes and show how to learn cooperative movements for dynamic obstacles such as human beings by deep reinforcement learning.

III. Proposed System

An autonomous mobile robot is a system that various sensors and motor control units are integrated. Recently, ROS (Robot Operating System) is widely used as development platform for integrating many robot components such as sensors. In the ROS platform, a distributed system is built with independent processes called "node" communicating each other. The proposed system is also developed in the ROS platform. We have mainly developed a robot system including following two nodes as core components.

1) Topological map node: Storing a topological map, planing global paths, and selecting a next destination
2) Local navigation node: Moving between waypoints using control commands acquired by learning

Fig. 1 shows the proposed system configuration in the ROS platform.

The above 1) is considered to be a global path planing in general navigation system. Dijkstra [9] and A* [10] algorithm are widely known as algorithms for global path planning. These are usually used with occupancy grid maps. Contrastingly, our proposed system performes global path planning with topological maps. In addition, topological maps have robustness to changes in the shape of roads between waypoints, because the maps are represented with just nodes and edges. There are various ways to build a topological map. Generally, creating topological maps is low

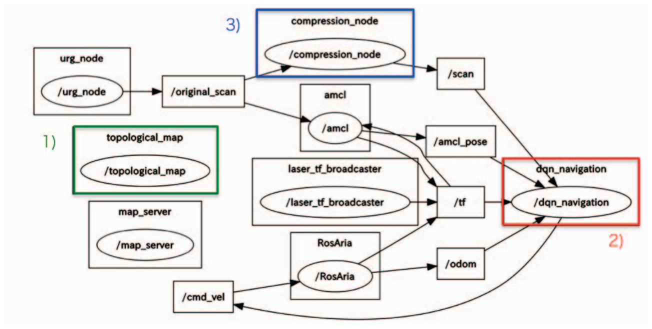


Fig. 1. Proposed system configuration. Node 1) and 2) are core elements of the proposed system respectively. Node 3) obtains a state from LRF raw range data as shown in Fig. 4. The other nodes are existing packages of ROS.

cost work. Automatical map generation methods have been also proposed as mentioned in in Chapter II, B [8]). This study doesn't consider building ways of topological maps. In the proposed system, waypoints are manually selected from floor maps as a simple way.

There are two important elements in a navigation system using topological maps. One is local navigation between waypoints and the other is node recognition. Although there are many robot systems adopting road following method for local navigation, it is not easy to apply the method in the environments where many pedestrians exist. Besides, robots should not make pedestrians feel unsafe because of geting close each other. Cooperating navigation in the environments crowded with people is a challenging problem. Therefore, we aimed to develop a local navigation method in the environments with pedestrians by applying deep reinforcement learning. The above 2) is a local path planning node to implement the proposed local navigation. The details of learning system is mentioned in the next section.

To recognize reaching nodes is a difficult problem in topological map based navigation. Even characteristic nodes such as intersections are difficult to recognize based on shapes due to the influence of dynamic obstacles like humans. Position-based recognition is a more practical way. In this study, Monte Carlo localization [11] is adopted for recognition of reaching nodes. When the robot approaches within 0.5[m] of the radius of the destination node, the system decides that the robot arrived at the destination node. In the ROS platform, Monte Carlo localization as open source is available in AMCL (Adaptive Monte Carlo Localization) package. The proposed system uses AMCL with grid maps for nodes recognition only as preliminary implementation. While traveling between nodes, localization is performed by another method. In the future, a robot system without any grid maps is desired. The other the node recognition algorithm should be applied to the proposed system. For example, precise position estimation systems such as low cost RTK-GPS or QZSS Satellite have possibilities for node recognition in outdoor environments. Then, our proposed system that robots doesn't need occupancy grid maps is completed.

Robot localization while traveling between nodes is indispensable in order to travel toward the right destination. Odometry obtained from the rotary encoder is used for localization. Although it is not practical to travel long distance with only odometry, the traveling distance between nodes is at most 10 meters. After arriving at nodes, the robot positions are corrected using node positions. In this way, the proposed system achieves autonomous robot navigation with topological maps and local navigation using learning results.

IV. Learning System of Local Navigation

A. Purpose of Learning

This study aims to achieve robot navigation between nodes in dynamic environments crowded with many people such as pedestrians not static environments. To travel between nodes, learning of the action toward the target node with avoiding obstacles is performed. Deep reinforcement learning is applied for the proposed learning system.

A state s_t of the robot used for learning is expressed as follows: $s_t = \{s_{laser}, s_{distance}, s_{angle}\}$. s_{laser} is raw range data scanned by a LRF and used to prevent the robot from colliding with obstacles. $s_{distance}$ and s_{angle} represent a distance and a direction to a destination respectively. These two states are calculated from the robot's self-position estimated from odometry, and the world coordinate of target node. From these states, the robot learns actions for travelling between nodes while avoiding obstacles.

In order to learn behaviors for avoiding dynamic obstacles, a simulator in dynamic environments is needed. However, it was found that the execution speed of existing simulators is quite slow against learning speed and it was not suitable for learning in dynamic environments. Then, we develop a original simulator where walking pedestrians can be placed freely at first. Also, the simulator is required to make learning speed shorter.

B. Development of Simurator

Deep reinforcement learning requires a large amount of data for obtaining appropriate behaviors. Especially, obstacle avoidance behaviors must be acquired in learning. To learn avoidance behaviors, a series of actions that collide with obstacles is needed to be iterated. Such actions with collisions can not be performed in in a real world actually although this study aims to achieve the robot navigation system in real world. So it is desirable to train and verify behaviors on the simulator and training results are applied to the actual robot system.

Examples of existing robot simulator include V-REP and Gazebo and so on. However, these perform 3-dimentional simulation and include a lot of unnecessary processing. This made it difficult to iterate learning several times with various learning conditions such as hyperparameters or environments (i.e. walking speed, number of pedestrians). This is why we developed the original 2D mobile robot simulator using a physical operation engine called Box2D. The developed simulator is called "$Fabot2D(Fast\ Robot\ 2D\ Simurator)$".

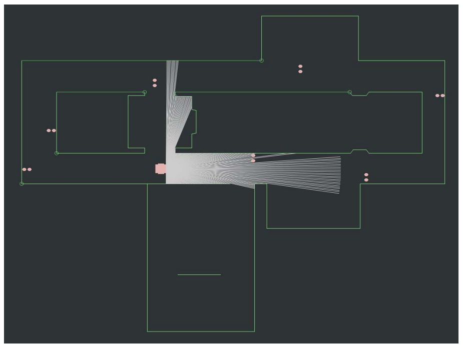


Fig. 2. Simulator environment developed originally. The robot is the differential drive robot with two wheels equipped with a laser range finder. Pseudo pedestrians can be represented by moving a circular object in parallel.

TABLE I
Training speed comparison between V-REP and Fabot2D.

Name	Speed(step/hour)	Learning finish time(hour)
V-REP	$3*10^4$	160
Fabot2D	$6*10^5$	8

Fig. 2 shows a example of a simulation environment of Fabot2D. This example includes 2D structures of the environment and walking pedestrians. Table. I shows a comparison of learning speed between V-REP and Fabot2D under the same learning condition. This experiment is performed under the following conditions. CPU: Intel Core i7-6800K CPU @ 3.40CHz $\times$ 12, GPU: GeForce GTX 1070/PCIe/SSE2. By using Fabot2D, approximately 20 times faster learning than V-REP simulator was achieved. Also, since Fabot2D is implemented so that the environment can be duplicated, this simulator can be applied to asynchronous learning algorithms.

Robots are expected not only to avoid obstacles but also to behave coordinated with humans (i.e. left-hand traffic, following pedestrians, etc.). Therefore, we implemented a function to add objects imitating pedestrians. A set of two pink colored circles in Fig. 2 represents one pedestrians, and each objects move with constant linear motions. By learning in the environment where these pseudo pedestrian objects exist, it is expected that robots can learn behaviors to coordinate with pedestrians.

C. Network Structure

DDQN is a learning method that can suppress overestimation of Q-value in DQN. Since it has been confirmed that performances to many tasks are higher than DQN, this study adopts DDQN for the proposed navigation system.

The structure of the network of DDQN is shown in Fig. 3. The network of DDQN is consisted of an input layer, hidden layers and an output layers. Details of the input / output layer are described in the next section. Generally, it is difficult to theoretically decide the optimum number of units of the hidden layer. Preliminary experiments with changing the number of units of the hidden layer were performed to decide a suitable number. Some learning features were obtained from experimental results. For example, deeper hidden layers

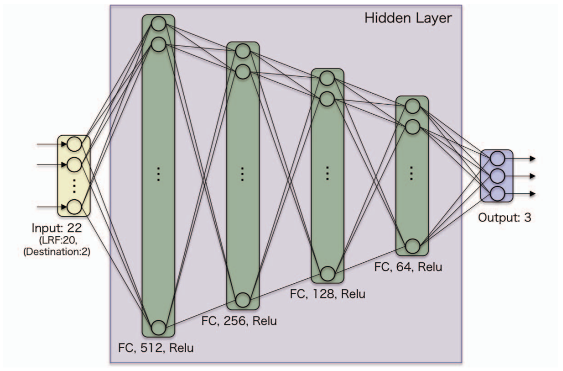


Fig. 3. DDQN network structure. Input is a 22-dimensional vector including 20 ranges from LRF and distance and direction to the destination. Output is a 3-dimensional vector that includes Q-values of three behaviors. Setting of the hidden layer was decided by trial and error.

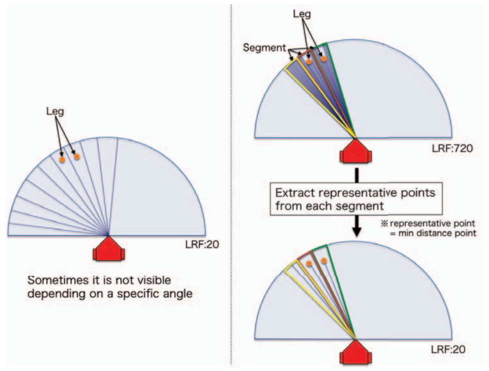

Leg
LRF:20
Sometimes it is not visible depending on a specific angle
Leg
Segment
LRF:720
Extract representative points from each segment
※representative point = min distance point
LRF:20

Fig. 4. State representation of two-dimensional LRF. The smallest range data is selected from raw LRF range data as a representative value in each segment.

make robots travel more safely away from obstacles. On the other hand, too much deeper layers leaded to increase calculation time for learning and decrease generalization performance. Thus, hidden layer setting is as shown in Fig. 3.

D. Modeling of the problem

At first, "action" selected in learning process is described. DDQN provides discrete outputs as behaviors of the robot. In this study, 3 behaviors were predetermined for the output. Then, the number of units of the output layer is three as shown in Fig. 3. The behavior patterns $a_t(i)$ are configured by linear velocity v_{linear} and angular velocity $v_{angular}$ as follows.

$$\begin{aligned} a_t(i) &= (v_{linear}, v_{angular}) \\ &= ((0.5, 0.0), (0.4, 0.5), (0.4, -0.5)) \end{aligned} \tag{1}$$

These behaviors include linear velocities of forward direction because the robot monitors 180 degrees in front of the robot and cannot see anything in the back. In addition, stopping or spinning action would be also essential in the actual robot system. These actions, however, aren't considered for learning in this study because stopping or spinning actions were selected in many cases as learning results in preliminary learning experiments including such stopping and spinning actions.

Next, "state" of the learning process is explained. As described in IV-A, the state of the robot s_t used for the input layer is configured as follows.

1) s_{laser}: selected 20 range values of LRF
2) $s_{distance}, s_{angle}$: distance and direction to the destination

1) is information for recognizing where the robot can travel. In this paper, robots are required not only avoiding walls, but also cooperative behaviors with dynamic obstacles such as pedestrians. So, small obstacles such as human legs must be visible by LRF scanning. On the other hand, the smaller input dimension to the neural network made learning more accurate in preliminary experiments. Therefore, 20 range points measuring small obstacles are selected from 720 range points acquired with two-dimensional LRF. The details of the method is shown in Fig. 4. First, 720 range points of LRF are devided into 20 segments. Each segment have 36(=720/20) range points. Next, a point with the smallest range in each segment is selected as a representative point of the segment. As a result, small obstacles such as human legs are not overlooked and the input dimension can be reduced. Finally, each representative range is normalized to [0.0, 1.0] by setting the maximum observation distance of LRF 10[m].

2) needs for the robot to head for the destination. Assuming that self position and orientation of the robot: $p_r = (x_r, y_r, th_r)$ and the destination: $p_d = (x_d, y_d)$ are known in world coordinate system, the destination in the robot coordinate: $p'_r = (x'_r, y'_r)$ can be calculated from Eq.(2).

$$\begin{pmatrix} x'_r \\ y'_r \end{pmatrix} = \begin{pmatrix} \cos(\theta_r) & -\sin(\theta_r) \\ \sin(\theta_r) & \cos(\theta_r) \end{pmatrix} \begin{pmatrix} x_d - x_r \\ y_d - y_r \end{pmatrix} \tag{2}$$

The distance $s_{distance}$ and the direction $s_{angular}$ to the destination are calculated from Eq.(3). Finally, $s_{distance}$ is normalized to [0.0, 1.0] by Eq.(4) and $s_{angular}$ are normalized to [-1.0, 1.0] divided by π.

$$\begin{aligned} s_{distance} &= \|p'_r\| = \sqrt{((x'_r)^2 + (y'_r)^2)} \\ s_{angular} &= \tan^{-1}(y'_r / x'_r) \end{aligned} \tag{3}$$

$$s'_{distance} = \begin{cases} s_{distance}/20.0 & (s_{distance} \leq 20.0) \\ 1.0 & (else) \end{cases} \tag{4}$$

Finally, "reward" given in the learning process is presented. Rewards in the learning system are set according to Eq.(5) with reference to [5]. When the robot reaches the destination, the arrival reward is given. In this study, the condition ($s_{distance} \leq 0.5$) is regarded as arrival to the destination. When colliding with obstacles such as walls and people, the collision reward is given. In other cases, positive or negative reward is given according to relations between the robot and the destination. The positive reward is given for actions approaching the destination and the negative reward is given for actions to go away from the destination. $s_{distance(t)}$ represents $s_{distance}$ at time "t". c_r is an attenuate hyper-parameter.

$$r_t = \begin{cases} 1.0 & (s_{distance} < 0.5) \\ -1.0 & (collision) \\ (s_{distance(t-1)} - s_{distance(t)}) \cdot c_r & (else) \end{cases} \quad (5)$$

E. Flow of Training

A whole flow of training is summarized in this section. A series of processes from action selection to learning are iterated in the training. This series of processes is called ”step”. A series of steps until the robot arrives at the destination or collides with obstacles from the initial pose are regarded as 1 ”episode”. In the whole flow of training, episodes including several steps are iterated. The combination of the initial pose of the robot and the destination is randomly selected at the beginning of each episode. The initial poses of dynamic objects imitating pedestrians are also initialized at the beginning of the episode. In the action selection, the index number of action i is obtained based on ϵ-greedy algorithm. ϵ-greedy selects an action that takes the maximum Q-value in s_t or a random action with a higher probability at the beginning of learning. As a result of action $a_t(i)$, reward r_t and judgment d_t and the next robot state s_{t+1} are obtaind. Then, these are added to experience memory. Learning is performed by retrieving 32 lists randomly from the experience memory and updating the unit weights of the neural network by the error back propagation algorithm. From these processes, the Q-value of the action which should be taken in s_t is updated to become higher. A brief summary of learning flow in each episode is shown as follows.

1) Decide the initial robot pose and the destination point
2) Initialize human poses as dynamic obstacles
3) Get the state of the robot s_t
4) From ϵ-greedy algorithm, action index i is selected from s_t or randomly
5) Take action $a_t(i)$, and get next state of the robots_{t+1}, reward r_t and end judgment d_t
6) Add the list $(s_t, a_t, r_t, d_t, s_{t+1})$ to experience memory
7) Update weights of neural network by error back propagation method Using 32 lists taken from experience memory
8) d_t is True: back to 1), else: back to 3)

V. Experiment

A. Learning Phase

This study proposed a navigation system integrating global navigation based on topological maps and local navigation based on learning between nodes of topological maps. Then, global topological maps and local behaviors acquired by learning must be prepared in advance.

At first, we provide learning results to acquire local behaviors. There are several general patterns of road shape for robot navigation as mentioned in [7] or [8]. In this paper, we classified them into seven patterns as shown in Table. II.

For robot navigation in environments crowded with people, robot must learn behaviors moving to the destinations in

TABLE II
7 Patterns of the Shape of the Road

Shape	Description
Straight	│
Left Corner	┐
Right Corner	┌
T-Junction	⊤
Left T-Junction	⊣
Right T-Junction	⊢
Crossroad	+

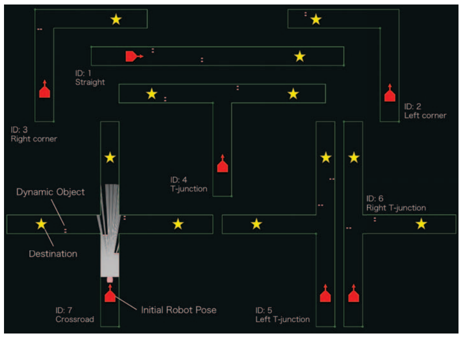


Fig. 5. Simulator environment including seven road patterns used for learning as shown in Table. II. Dynamic objects imitating pedestrians can be placed in a predetermined position. A combination of a initial robot pose (red shape) and a destination (yellow star) is randomly selected in advance in each training episode. Then, initialization of the robot and setting of the destination are performed for training.

above patterns and avoiding pedestrians. This study prepared the environment shown in Fig. 5 for learning of robot navigation behaviors.

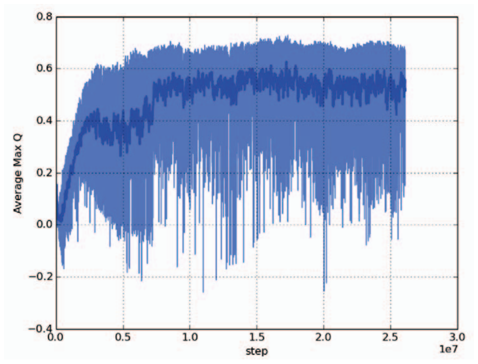
Fig. 6 shows the changes of Q-value and reward in training. In Fig. 6(b), the reward is changing from 1.0 to 1.5 after 6M step(10 hours). It means that the robot could learn how to arrive the destination and avoid dynamic objects in less than half a day.

Next, we made a topological map from a occupancy grid map of a target environment. Nodes and edges of the topological maps are manually selected in the occupancy grid map as preliminary works. The occupancy grid map was generated from gmapping node which is an existing package of ROS. Fig. 7 shows the generated occupancy grid and a topological map in a floor of our university campus.

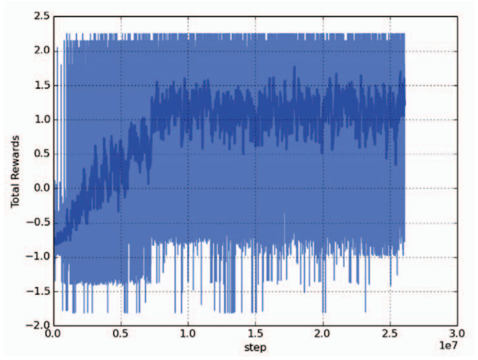
B. Navigation in the Simulator Environment

In order to show the validity of the proposed method, we conducted experiments in a simulator environment. This simulation uses the learning result in environments with general road shapes shown in Table. II for local navigation. Fig. 8 shows the experimental environment on the simulator which includes not only static obstacles but also dynamic obstacles. In this environment, we evaluate a navigation performance whether the robot can travel to destinations without colliding with obstacles.

Fig. 9 shows one of the results of traveling path. This result shows that navigation avoiding dynamic obstacles is possible on the simulator. Recognition of arriving nodes is performed using self-positions acquired by AMCL based on



(a) Average of max Q-value in each step



(b) Total rewards in each step

Fig. 6. Transitions of Q-values and rewards in training. (a) shows the change of averages of max q-values. The robot can become selecting good actions gradually. (b) shows the change of rewards. After 6M steps, the robot can arrive at the destination without colliding with walls and dynamic objects imitating pedestrians.

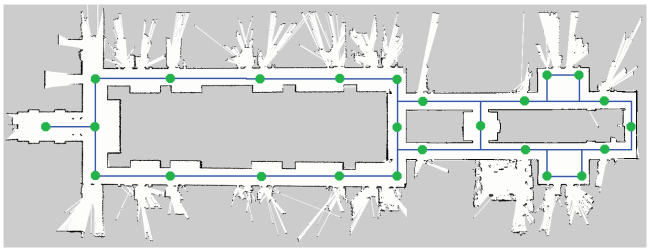


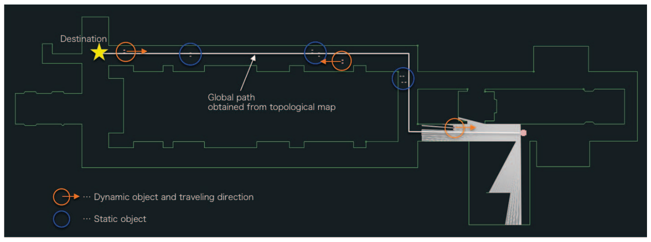
Fig. 7. The occupancy grid and topological map using for learning based navigation. The characteristic locations in the environment manually determined as nodes (green circles). A world coordinate of each node is extracted from the occupancy grid map. Edges are obtained as connections between nodes (blue lines).

grid maps.

Additional experiments are conducted for evaluation of navigation performances in the environment shown in Fig. 8. In the additional experiments, navigations in random paths are iterated 10 times. Completion of the navigation, collisions with static or dynamic pedestrians are evaluated. The initial positions and the destinations of the robot are randomly initialized. Also, static and dynamic pedestrians are randomly placed in the environment. Table. III shows the result of experiments. The robot could not reach the destination just once by colliding with the walls. There were no collisions with static obstacles at all. Collisions with dynamic objects are occurred approximately 40 percent. From these results,

TABLE III
EXPERIMENT ON THE SIMULATOR.

	Time of Success / Attempt	Percentage
navigation	9/10	90%
Static Obstacle Avoidance	10/10	100%
Dynamic Obstacle Avoidance	11/19	58%



Destination
Global path
obtained from topological map
··· Dynamic object and traveling direction
··· Static object

Fig. 8. An experimental environment on the simulator. Static and dynamic objects are placed in the environment. A global path is calculated using a topological map. The robot navigation between nodes are performed with learning results.

the possibility of navigation based on deep reinforcement learning could be shown, though it is not perfect.

C. Navigation in the Real World

The above section showed that navigation is possible on the simulator. On the other hand, navigation in the real world is more difficult than on the simulator because of disturbances. In this section, we evaluate robustness of the proposed system in a real environment.

The robot hardware configuration used in the experiment is shown in Table. IV, and the appearance of the robot is shown in Fig. 10. A LRF is attached to the robot at the height of 125[mm] from the ground so that it is able to detect most of obstacles in the environment. In the experiment, pedestrians in the environment intentionally walked as follows.

- Standing in front of the robot to block traveling in straight direction
- Walking in front of the robot at low speed to encourage overtaking.

The defference between simulator and real world is that the shape of the environment cannot be observed accurately. Additionally, the width and shape of the corridor are also different compared with the learning environment. Nevertheless, Fig. 11 shows success of navigation. This result provides the possibility of applying deep reinforcement learning to autonomous mobile robot.

VI. CONCLUSION

This study proposed a navigation system with local navigation based on deep reinforcement learning and global navigation based on a topological map. By learning in the simulation environment with many pedestrians, the robot learned policies traveling to the local destinations while avoiding dynamic and static obstacles. Also, a global navigation system was achieved by applying the learning results to traveling between nodes in topological maps. Simulation and experiments in a real world show that our proposed system

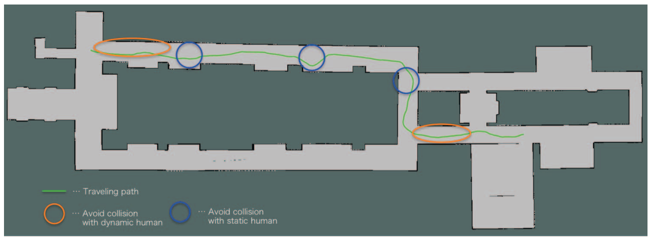

··· Traveling path
··· Avoid collision with dynamic human
··· Avoid collision with static human

Fig. 9. Traveling result in the simulator. This grid map was made beforehand in the simulator environment of Fig. 8. Objects imitating pedestrians were placed manually. The initial pose of the robot and the destination are also decided manually, and navigation was completed on the simulator. The result shows that the robot can acquire abilities for going to destinations and avoiding static and dynamic obstacles.

TABLE IV
HARDWARE CONFIGURATION

	Device Name	Manufacturer
Robot	Pioneer3DX	Adept MobileRobots
Sensor Unit	UTM-30LX	Hokuyo Automatic co.
System Control Unit	PC(dynabook R93/PB)	Toshiba co.

has effectiveness for navigation in a environment with many people.

Several problems are remained in the current system. At first, velocities of the robot and pedestrians must be increased for navigation in real worlds. In this study, the upper limit of the translational velocity of the robot was set to 0.5[m/s]. Also, the pedestrians walked at 0.5[m/s] or less in the simulation and experiments. However, the walking velocity of a general pedestrian is about 1.1[m/s] and variable. The learning system should be improved for addressing such general pedestrians.

Behaviors of the robot are represented as discrete values and have to be determined in advance, since the current learning system uses DDQN algorithm. The robot is required to adjust the moving velocity according to actual pedestrians walking with variable and fast velocities. Therefore, the learning algorithm to output continuous and variable actions should be considered. For example, A3C (Asynchronous Advantage Actor-Critic) [12] is promising for this application.

Next problem is recognition of arriving nodes in topological maps. In topological map-based navigation, recognition of nodes is usually not easy. In this study, the current system compares robot self-positions estimated by AMCL with the coordinates of nodes in the world coordinate system for recognition of arriving nodes. However, this system requires an occupancy grid map. This system aims to achieve a navigation system without occupancy grid maps. Then, the other methods for recognition of arriving nodes must be considered. For example, accurate GPS systems can be used in outdoor environments.

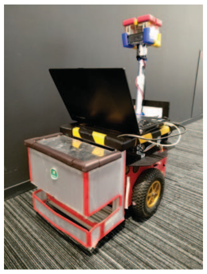


Fig. 10. Appearance of the robot used for experiment.

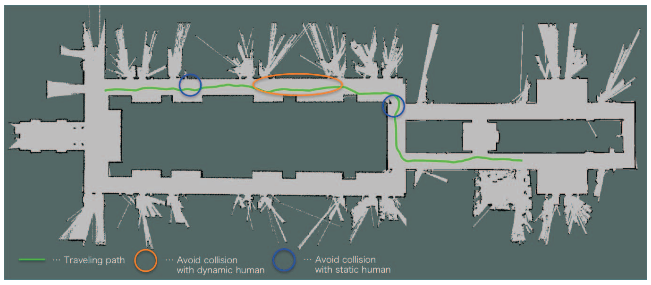

··· Traveling path
··· Avoid collision with dynamic human
··· Avoid collision with static human

Fig. 11. Traveling demonstration in the real world. The initial pose and the destination similar to Fig.9 were set in advance. In this experiment, the subject pedestrians walked with intentional paths so that the robot might select actions avoiding pedestrians. Though there were many different conditions from the simulator environment in the real world, the robot could reach the destination without colliding with the obstacles.

REFERENCES

[1] Giorgio Grisettiyz, Cyrill Stachniss, and Wolfram Burgard. Improving grid-based slam with rao-blackwellized particle filters by adaptive proposals and selective resampling. *Proceedings of the 2005 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 2432–2437, 2005.
[2] Dieter Fox, Wolfram Burgard, and Sebastian Thrun. The dynamic window approach to collision avoidance. *IEEE Robotics & Automation Magazine*, Vol. 4, No. 1, pp. 23–33, 1997.
[3] Jingwei Zhang, Jost Tobias Springenberg, Joschka Boedecker, and Wolfram Burgard. Deep reinforcement learning with successor features for navigation across similar environments. *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2017.
[4] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. *NIPS Deep Learning Workshop 2013*, 2013.
[5] Lei Tai, Giuseppe Paolo, and Ming Liu. Virtual-to-real deep reinforcement learning: Continuous control of mobile robots for mapless navigation. *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2017.
[6] Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. *2016 International Conference on Learning (ICLR)*, 2016.
[7] Atsushi Watanabe, Shigeru Bando, Kazuhiro Shinada, et al. Road following based navigation in park and pedestrian street by detecting orientation and finding intersection. *2011 International Conference on Mechatronics and Automation (ICMA)*, pp. 1763–1767, 2011.
[8] Hongtai Cheng, Heping Chen, and Yong Liu. Topological indoor localization and navigation for autonomous mobile robot. *IEEE Transactions on Automation Science and Engineering*, Vol. 12, No. 2, pp. 729–738, 2015.
[9] Edsger W Dijkstra. A note on two problems in connexion with graphs. *Numerische mathematik*, Vol. 1, No. 1, pp. 269–271, 1959.
[10] Peter E Hart, Nils J Nilsson, and Bertram Raphael. A formal basis for the heuristic determination of minimum cost paths. *IEEE Transactions on Systems Science and Cybernetics*, Vol. 4, No. 2, pp. 100–107, 1968.
[11] Dieter Fox. Kld-sampling: Adaptive particle filters. *Advances in neural information processing systems*, pp. 713–720, 2002.
[12] Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. *Proceedings of International Conference on Machine Learning (ICML)*, pp. 1928–1937, 2016.