2020 IEEE International Conference on Robotics and Automation (ICRA)
31 May - 31 August, 2020. Paris, France

Cooperative Multi-Robot Navigation in Dynamic Environment with Deep Reinforcement Learning

Ruihua Han, Shengduo Chen and Qi Hao*

***Abstract*— The challenges of multi-robot navigation in dynamic environments lie in uncertainties in obstacle complexities, partially observation of robots, and policy implementation from simulations to the real world. This paper presents a cooperative approach to address the multi-robot navigation problem (MRNP) under dynamic environments using a deep reinforcement learning (DRL) framework, which can help multiple robots jointly achieve optimal paths despite a certain degree of obstacle complexities. The novelty of this work includes threefold: (1) developing a cooperative architecture that robots can exchange information with each other to select the optimal target locations; (2) developing a DRL based framework which can learn a navigation policy to generate the optimal paths for multiple robots; (3) developing a training mechanism based on dynamics randomization which can make the policy generalized and achieve the maximum performance in the real world. The method is tested with Gazebo simulations and 4 differential drive robots. Both simulation and experiment results validate the superior performance of the proposed method in terms of success rate and travel time when compared with the other state-of-art technologies.**

I. Introduction

Multi-robot systems are advantageous in performing complex tasks such as surveillance and rescue, formation and exploration, cooperative manipulation [1]. However, when the number of robots increases, more technical challenges arise in perception, navigation, and formation control. Many approaches have been developed to solve MRNP such as simultaneous localization and mapping (SLAM) based planning [2]–[4], velocity obstacles (VO) based velocity selection [5]–[7], based on assumptions such as fully environment modeling, perfect sensing, which are unfeasible in the dynamic environment, as shown in Fig. 1.

Recently, reinforcement learning (RL) based approaches have been developed to solve the navigation problem in dynamic environments [8]. Nevertheless, there are still technical challenges for them to solve the multi-robot navigation problem in complex environments, including:

1) **Inefficient Target Location Allocation**. Usually, multiple robots have several target locations; how to allocate target locations to robots will determine the travel time of the group.
2) **Lack of Cooperation Scheme**. For multiple robots, they could experience different situations and conditions; how to develop a cooperative architecture to take full advantage of the accumulative experiences to achieve an optimal navigation policy.
3) **Ineffective Transfer from Simulation to the Real World**. How to develop a mechanism that can constantly improve the system parameters in simulation to achieve an optimal policy workable in the real world.

*This work is partially supported by the National Natural Science Foundation of China (No: 61773197); the Science and Technology Innovation Committee of Shenzhen City (No: CKFW2016041415372174,GJHZ20170314114424152); and the Nanshan District Science and Technology Innovation Bureau (No: LHTD20170007); and the Intel ICRI-IACV Grant 2019.

The authors are with Department of Computer Science and Engineering, Southern University of Science and Technology, SUSTech-Haylion Center for Intelligent Transportation, Shenzhen Research Institute for Trustworthy Autonomous Systems, Shenzhen, Guangdong, 518055, China. {`hanruihuaff@gmail.com haoq@sustc.edu.cn`} *Corresponding author

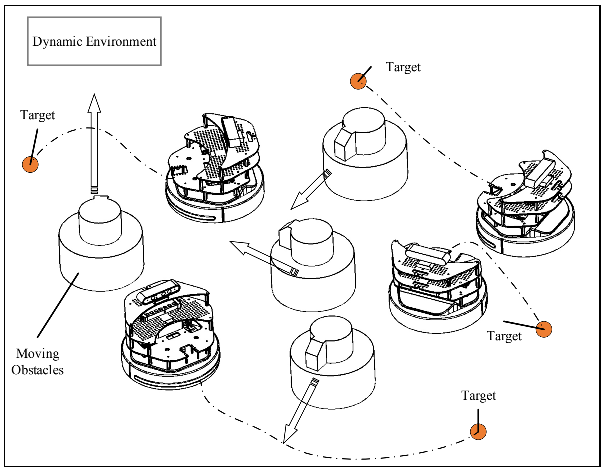

Fig. 1. Multi-robot navigation in dynamic environment

Some approaches focus on learning the collision avoidance policy with pre-allocated goal locations from the raw sensor measurements of individual robots using the DRL framework [9]–[11]. Other approaches learn the navigation policy only for the single robot or in the static environment [12], [13]. Generally speaking, these approaches cannot adequately address the MRNP. RL methods are also developed to achieve cooperative schemes among multiple robots such as parameter sharing policy [14], [15], but they are unable to allocate target locations optimally, leading to low navigation efficiency. Besides, many expensive efforts have been made to collect huge amounts of data for the transfer learning from simulation to the real world [16], [17].

In this paper, we propose a DRL framework to the MRNP in dynamic environments, where a cooperative architecture is developed to combine the target location allocation and collision avoidance into the training process to learn the navigation policy. We also develop a set of randomized dynamics models during the training process in the simulator to alleviate the mismatch between simulation and the real

TABLE I
RELATED WORK

	Approach	Algorithm	Strength(s)	Limitation(s)
RL based Policy	RL based navigation	Q-learning	Can tackle unknown environment with moving obstacles and targets	Only consider single robot; assumption of perfect sensing
	SLCAP	PPO	Able to handle large robot system and complex environment	Only consider the range sensor when detects the obstacles
	GA3C-CADRL	A3C	Able to tackle obstacles with various number and velocity	Policy update asynchronously is inefficient
	IDRL	DQN & DDPG	Avoid the collision and select the target simultaneously	Only avoid the static obstacles
Sim-to-Real Transfer	Inverse dynamics model	-	Able to tackle various dynamics model with superior performance	Require the collection of the real world data
	Dynamics randomization	-	Able to make policy generalized to adapt the real world	Require a specific dynamics model

world. The main contributions of this work include:

1) Developing a cooperative architecture that can utilize the accumulative experiences of all robots for target location allocation.
2) Developing a training algorithm to learn the navigation policy, including target location allocation and collision avoidance based on the DRL framework.
3) Developing a transfer mechanism from simulation to the real world during the simulation training, which can help achieve a better-generalized navigation policy and maximize the performance of robots in the real world.

The rest of paper is organized as follows. Section II discusses the related approaches to solve the MRNP. Section III describes the system setup and the problem statement. Section IV describes the framework of DRL and its configuration. Section V presents the dynamics parameters applied in the training process. Section VI presents the target location allocation and the policy training algorithm. Section VII provides the simulation and experiment results. Section VIII concludes the paper and outlines future work.

II. RELATED WORK

RL-based approaches have demonstrated many advantages in solving the navigation problem under dynamic environments [8]. For example, Q-learning based approaches are proposed to solve the mobile robot navigation problem in unknown dynamic environments [12]. However, those approaches only deal with single robot navigation and assume perfect sensing, which limits their applications. A use of the actor-critic method referred to as multi-agent deep deterministic policy gradient (MADDPG) has been proposed, which allows the policies to use extra information to ease training [18]. Deep neural networks have been proposed to learn a collision avoidance policy without perfect sensing assumptions, which can directly map the raw sensor measurements to the robot's actions [9]. This approach is extended to a decentralized sensor-level collision avoidance policy (SLCAP) for large-scale robot systems using Proximal Policy Optimization (PPO) [15]. However, those approaches heavily rely on range sensors that cannot detect all types of obstacles; besides, it will incur high hardware cost and energy consumption for their applications in large scale multi-robot systems.

Another RL-based collision avoidance (CADRL) approach has been developed by training the value network, which can find a collision-free velocity vector for multiple agents [10]. Its extension GA3C-CADRL applies the Asynchronous Advantage Actor-Critic (A3C) to learn multi-agent collision avoidance policy [11]. However, in these methods, all the robots update the policy with many threads asynchronously, which is inefficient during the training process for homogeneous robots. Besides, these methods only consider the collision avoidance policy with pre-allocated target locations, which is only a part of the multi-robot navigation problem. An interlaced DRL (IDRL) approach has been proposed to combine the dynamic target selection and collision avoidance into the navigation policy [19]. It uses a Deep Q-Network (DQN) to learn the target selection policy and uses the deep deterministic policy gradient (DDPG) to learn the collision avoidance policy simultaneously. However, this approach only deals with the environment with static obstacles. Meanwhile, a deep inverse dynamics model to decide what the action should be in the real world based on the simulator policy is proposed in [16]. However, this approach needs data collection from the real world to train the inverse dynamics model constantly, which is inefficient for the navigation problem. A randomized dynamics model during the training process has been proposed to achieve a generalized policy without using any real-world data [20]. Nevertheless, this approach requires a complicated dynamics model for training. The summary of these approaches is listed in Table I.

In this work, we focus on tackling the MRNP in dynamic environments with multiple moving obstacles. We develop a cooperative DRL based framework to learn the navigation policy based on the collective experiences of all robots. The policy is shared and updated by all robots simultaneously. Unlike SLCAP, this policy uses the poses and velocities of robots as input, which can be implemented for various types of sensors. Besides, a simple randomized dynamics model for navigation is developed during the simulation training process to make the trained policy more adaptive in the real world.

III. SYSTEM SETUP AND PROBLEM STATEMENT

A. System Framework

The system framework of our multi-robot navigation system is shown in Fig. 2. The sensors for obstacle detection

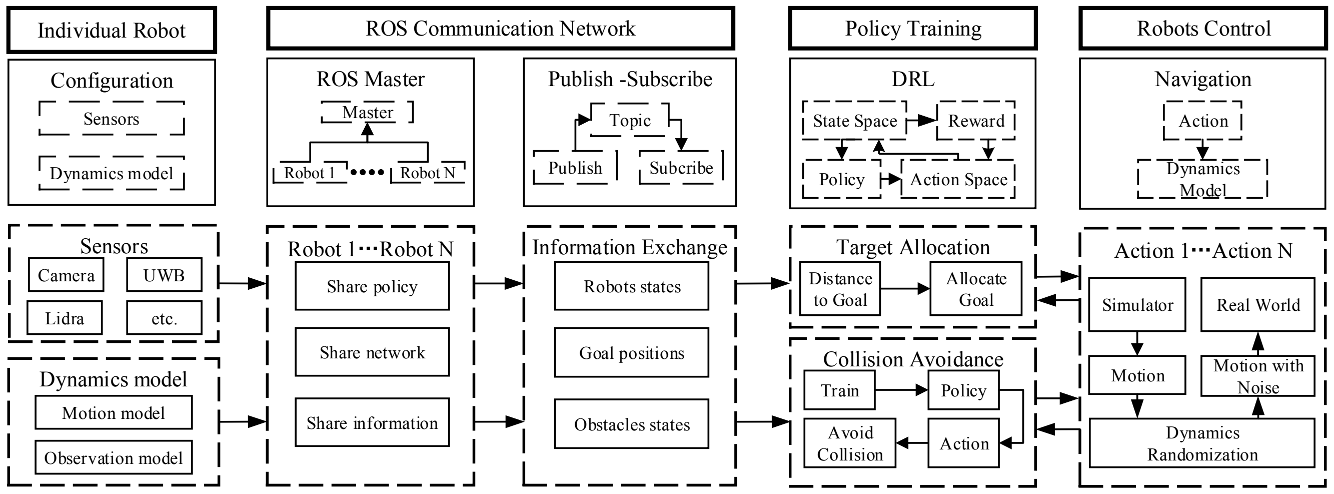

Fig. 2. System framework of multi-robot navigation policy

include Lidar, camera, Ultra-Wide Band (UWB). A group of robots in the same network will use the same ROS master to guarantee the messages passing. Based on the accumulative information from all robots, the target location allocation, which can make navigation more efficient and collision avoidance are combined in the DRL framework to train the navigation policy. The policy is shared and updated by all robots simultaneously. In the simulation, the randomized dynamics models are applied during the training process to make policy more adaptive to the real world.

B. Problem Statement

The MRNP is actually a Partially Observable Markov Decision Process (POMDP) that a group of robots navigate to the target location in dynamic environment with collision avoidance and minimum time cost. The state vector of the single robot $\mathbf{s}$ at time t is composed of Cartesian coordinates, orientation and speed, $\mathbf{s}_r^t = [p_{rx}^t, p_{ry}^t, \theta_r^t, v_{rx}^t, v_{ry}^t]$. Specifically, all robots have the same radius r. The state of moving obstacles includes Cartesian coordinates, speed and radius, $\mathbf{s}_o^t = [p_{ox}^t, p_{oy}^t, v_{ox}^t, v_{oy}^t, r_o]]$. The target position of robots can be denoted as $\mathbf{s}_g = [p_{gx}, p_{gy}]$. In the condition of these previous states, the navigation policy generates the actions a at each time step for robots, $\mathbf{a}^t \sim \pi_\theta(\mathbf{a}^t|\mathbf{s}_r^t, \mathbf{s}_o^t, \mathbf{s}_g)$. Where, θ is the parameter of policy. The action is composed of transitional and rotational velocity of differential drive robots, $\mathbf{a}^t = [v_t^t, v_r^t]$. Thus, for a group of N robots navigating in dynamic environment with M moving obstacles, the optimization objective should be:

$$\begin{aligned} &\underset{\pi_\theta}{\operatorname{argmin}}\ \mathbb{E}\left[T|\pi_\theta, \mathbf{s}_{r,1:N}, \mathbf{s}_{o,1:M}, \mathbf{s}_{g,1:N}\right], \\ &\quad s.t. \quad \forall i,j \in [1,N], k \in [1,M] \\ &\qquad\qquad d_{rr,i,j} > 2r_r \\ &\qquad\qquad d_{ro,i,k} > r_r + r_o \\ &\qquad\qquad d_{g,i} < d_{\min} \end{aligned} \tag{1}$$

where, $d_{rr,i,j}$ represents the distance between two arbitrary robots, $d_{ro,i,k}$ denotes the distance between robots and obstacles. These two distances are used to judge the collision. The distance $d_{g,i}$ is used to judge whether the robot i reaches the goal position depending on whether it is less than the minimum distance $d_{\min}$. The travel time T denotes the moment when all the robots reach their own target position, respectively, without any collision. The goal of this paper is to find an optimal navigation policy shared and updated by all robots in a dynamic environment by minimizing the expectation of travel time T.

IV. Deep Reinforcement Learning Framework

The framework of our DRL with actor-critic model is shown in Fig. 3. Firstly, the target locations are allocated to each robot. Secondly, each robot receives a state from a state space $\mathcal{S}$ and select an action from an action space $\mathcal{A}$, following the stochastic policy π_θ. Finally, the action leads to a new environment state and reward. Specifically, the critic part computes the value V of taking that action at that state to evaluate the action. The actor part updates its policy parameters using the value V.

A. State Space and Action Space

The state space can be divided into four parts: the current robot's state $\mathbf{s}_r^t$, the other robots' state $\tilde{\mathbf{s}}_r^t$, the obstacles' state $\mathbf{s}_o^t$ and the target position $\mathbf{s}_g^t$ as mentioned in Section III, $\mathbf{s}^t = [\mathbf{s}_r^t, \tilde{\mathbf{s}}_r^t, \mathbf{s}_o^t, \mathbf{s}_g^t]$. The obstacles state $\mathbf{s}_o^t$ is aggregated from the observation of all robots.

The action space is a probability distribution of continuous actions including transitional and rotational velocity, $\mathbf{a}^t = [v_t^t, v_r^t] \sim \mathcal{S}$. At each time step, in the condition of state $\mathbf{s}^t$, robots select the highest probability of action from an action space to navigate following the stochastic policy. Consider the real world situation that the robot only moves forward because of the limited view of observation, the range of transitional velocity is limited between 0 and 0.4 m/s. Similarly, given the obstacles detection speed, the rotational velocity is limited between -1 and 1 rad/s.

B. Reward Design

The goal of the DRL algorithm is to maximize the reward of a series of actions. Therefore, for robot i at time t, the reward r_i^t is designed as $r_i^t = r_{g,i}^t + r_{c,i}^t$, where, $r_{g,i}^t$ is

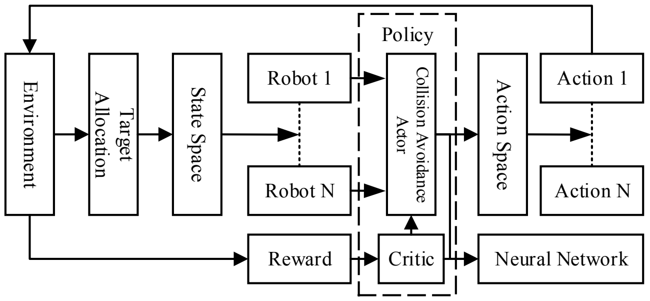

Fig. 3. Framework of deep reinforcement learning for navigation policy

the award when the robot reaches or approaches its target location which is judged by distance d_g^t. Otherwise, the reward is related to the change of d_g^t:

$$r_{g,i}^t = \begin{cases} 5 & if d_g^t < 0.1 \\ 1 + d_g^t * 0.5 & if 0.1 < d_g^t < 0.4 \\ 10 * (d_g^{t-1} - d_g^t) & otherwise \end{cases}. \quad (2)$$

$r_{c,i}^t$ is the punishment when there are collision among robots or obstacles judged by the distance d_{rr}^t and d_{ro}^t:

$$r_{c,i}^t = \begin{cases} -10 & if \quad d_{rr}^t < 2 * r_r \\ -10 & if \quad d_{ro}^t < r_r + r_o \\ 0 & otherwise \end{cases}. \quad (3)$$

C. Actor and Critic

The actor part that mapping from the state to action is designed as a 3-hidden-layer neural network. The input state is fed into the three fully-connected layers with 256, 128, and 64 rectifier units separately. The output layer is also a fully-connected layer with four units and different activation for different parts of action space. The sigmoid and Hyperbolic tangent activations are applied to output the mean of transitional and rotational velocity separately. Specifically, the clip function is also used to constrain these two velocities in range (0, 0.4) and (-1, 1). The final activation softplus is used to generate the standard deviation of these two velocities. The mean and standard deviation compose the Gaussian distribution of actions. Robots select the action $\mathbf{a}^t$ sampled from the Gaussian distribution at each time step t.

The critic part is the state value function $V(\mathbf{s}^t)$ and is designed as a 2-hidden-layer neural network, including two fully-connected layers with 128 and 64 rectifier units. The output layer only has one unit to generate value with linear activation.

V. Dynamics Randomization

The policy trained in the simulator may have superior performance. However, the performance is hard to remain in the real world because of the ubiquitous uncertainties in the process of robot control and obstacle detection. In a simulator, the dynamics model of robots obeys differential drive robot model; however, instead of the precise data read from the simulator, there are numerous sensor noises in the real world. Thus, inspired by [20], we apply the randomized dynamics model into the training process to make the navigation policy adaptive to the real world. During each episode, a series of dynamics parameters λ are selected uniformly from a range γ and keep the value until the end of this episode. The dynamics parameters λ for differential drive robots include:

- Noise in transitional velocity, ξ_1
- Noise in rotational velocity, ξ_2
- Noise in the position (coordinates) of robots, ξ_3
- Noise in position (coordinates) of obstacles, ξ_4
- Noise in the measurements of obstacles ξ_5
- Mass of robots m_r

where the noises are all Gaussian noise with a zero mean and specific variance. At each time step, the noise is sampled from the Gaussian distribution with a fixed variance. ξ_1 and ξ_2 is the action noise from the chassis motor. It will affect the motion model of differential drive robots. ξ_3 and ξ_4 is the noise in external sensors like Ultra-Wide Band (UWB), which can afford the global position of robots and obstacles. ξ_5 is the noise in obstacle detection derived from the onboard sensors, including the range and relative bearing. The mass of robots m_r can affect the dynamics model like inertia and friction and remains constant during an episode.

VI. Navigation Policy Training

A. Target Location Allocation Algorithm

Usually, multiple robots have several corresponding target positions. Each robot has a fixed target will be inefficient when the robot has to move to the further target instead of the closer one. In the same network, the global coordinates of all robots and target locations can be acquired at each time step. Thus, we develop a simple target location allocation algorithm depending on the distance between robots and targets described in Algorithm 1. The core principle is to calculate all the permutations of the allocation scheme between robots and targets to find the permutation with the shortest total distance length. In consideration of computation cost, this policy is performed with a time step interval T_a decided by the number of robots and computer performance.

Algorithm 1 Target Location Allocation Algorithm

```
1: Receive the state of $N$ robots and goal positions, $\mathbf{s}_{r,1:N}^t$, $\mathbf{s}_{g,1:N}^t$
2: for Each permutation in the permutations of $N$ robots and $N$ targets do
3:     for robot $i = 1, ..., N$ do
4:         Calculate the distance $d_i^t$ between robot $i$ and its target
5:     end for
6:     Calculate the total distance length of all robots $d_{total}^t$
7: end for
8: Select the permutation with shortest distance length
```

B. Training Algorithm

Those robots in a dynamic environment are homogeneous; the information of poses and velocities is used as the input of the training algorithm. All the robots share and update the navigation policy simultaneously. After the target location allocation, the robots should move to the goal position following the collision avoidance policy. Because of the continuous action space and complex environment, we apply the PPO [21], [22] with actor-critic style to learn the navigation policy.

The summary of the training algorithm is described in Algorithm 2. Firstly, at the start of the episode, the environment resets to an initial state. Dynamics parameters λ are uniformly sampled from a range γ. Secondly, the states of the robots $\hat{\mathbf{s}}_r^t$ are received with noise ξ_3, ξ_4, ξ_5. The goal position is allocated for each robot depending on target location allocation policy. Thirdly, the action is sampled from policy π_θ with highest probability in the condition of current states, $\mathbf{a}_i^t \sim \pi_\theta(\mathbf{a}_i^t|\hat{\mathbf{s}}_i^t)$. Specifically, before the action is published to the robot, noise ξ_1, ξ_2 is applied. Simultaneously, the action $\mathbf{a}_i^t$ without noise, state $\hat{\mathbf{s}}_i^t$ and reward r_i^t are collected to compute the estimated advantage $\hat{A}_i$ used by [23]. Finally, according to the estimated advantage $\hat{A}_i$ and policy π_θ, surrogate loss $L^{CLIP}(\theta)$ and state value loss $L^V(\psi)$ are constructed. $L^{CLIP}(\theta)$ is a clipped surrogate objective proposed in [21]. It is optimized by Adam optimizer and learning rate l_a for K epochs. $L^V(\psi)$ is the loss of state value function $V(\mathbf{s}^t)$ constructed by square of advantage $\hat{A}_i^t$. It is also optimized by Adam optimizer and learning rate l_v for L epoch.

Algorithm 2 Policy Training with PPO

```
1: Initialize neural network $\pi_\theta$
2: for episode=1,2,... do
3:     Reset the environment with the initial state, $\mathbf{s}_{init}$
4:     Sample the dynamics parameters $\lambda$ from a range $\gamma$ uniformly, $\lambda \sim \gamma$
5:     for robot $i$=1,2,... do
6:         Receive state $\mathbf{s}_i^t$, select the goal position $\mathbf{s}_{g,i}^t$
7:         Add noise, $\hat{\mathbf{s}}_i^t \sim \mathbf{s}_i^t + [\xi_3, \xi_4, \xi_5]$
8:         Sample action $\mathbf{a}_i^t \sim \pi_\theta(\mathbf{a}_i^t|\hat{\mathbf{s}}_i^t)$
9:         Add noise, $\hat{\mathbf{a}}_i^t \sim \mathbf{a}_i^t + [\xi_1, \xi_2]$
10:        Publish $\hat{\mathbf{a}}_i^t$ to robot $i$
11:        Collect state $\hat{\mathbf{s}}_i^t$, reward $r_i^t$ and $\mathbf{a}_i^t$ for $T_i$ time steps
12:        Compute advantage estimates $\hat{A}_i^1, \ldots, \hat{A}_i^T$
13:    end for
14:    Optimize surrogate loss $L^{CLIP}(\theta)$ wrt $\theta$, with Adam optimizer and learning rate $l_a$ for $K$ epochs
15:    $\theta_{\text{old}} \leftarrow \theta$
16:    Optimize value loss $L^V(\psi)$ wrt $\psi$, with Adam optimizer and learning rate $l_v$ for $L$ epochs
17:    $\psi_{old} \leftarrow \psi$
18: end for
```

VII. Experiments and Results

A. Simulation Setup

Numerous robotics simulators can be used to simulate the navigation of multiple robots such as Matlab, ROS Stage, ARGoS. In this paper, we use Gym-Gazebo [24] to simulate the dynamic obstacles and differential drive robots, Turtlebot. Gym-Gazebo integrates Gym API, Gazebo simulator, and ROS to build a 3D robotics platform for RL. Specifically, OpenAI Gym is a popular kit for RL research and Gazebo is a 3D robotics simulator with a physics engine and realistic rendering. This platform reduces the difference between the simulator, and the real world considerably.

As shown in Fig. 4, the training scenario is composed of several Turtlebot models and green circles with the corresponding number as goal positions for robots. Specifically, parts of Turtlebot models are formulated as dynamic obstacles with stochastic motion. Each stochastic trajectory is achieved by using the periodically changed velocity to generate as many as possible scenarios for robots to tackle. The key hyper-parameter and dynamics parameters during the training process are listed in Table II. To accelerate reward converge and compare the performance, the policy is trained with three stages. We first pre-train the policy without dynamic parameters and target allocation, i.e., Pre-Policy. Based on this Pre-Policy, we train the TA-Policy with the target location allocation algorithm. Finally, the dynamic parameters are applied during the training process to achieve the DP-Policy. The navigation policy is trained using Algorithm 2 on a computer with an Nvidia GTX 1070 graphics card and the six times accelerated Gazebo simulator; the whole training process takes about 36 hours.

B. Experiment Setup

The hardware experiment that implements navigation policy on a differential drive robot (Turtlebot) in a dynamic environment has been performed to demonstrate the performance of our policy in the real world. The configuration of Turtlebot and the obstacles is illustrated in Fig. 5. The dynamic obstacles are formulated with a kobubi base and block. To acquire the state of the environment, we use the UWB base station to build a global coordinate system for

TABLE II

PARAMETERS DURING TRAINING PROCESS

Parameters	Value (Range)
l_a in line 14	0.0005
K in line 14	15
l_v in line 16	0.001
L in line 16	10
ϵ in $L^{CLIP}(\theta)$	0.2
ξ_1, ξ_2	variance [0.01, 0.1]
ξ_3	variance [0.005, 0.05]
ξ_4	variance [0.005, 0.05]
ξ_5	variance [0.01, 0.1] for range, [0.01, 0.05] for bearing
m_r	[5, 6.5] kg

TABLE III

PERFORMANCE COMPARISON OF DIFFERENT POLICIES

Metrics	Method	Simulation				Real World			
		1	2	3	4	1	2	3	4
Success Rate	ORCA	1.0	0.97	0.96	0.88	0.91	0.85	0.76	0.69
	Pre-Policy	1.0	1.0	1.0	0.95	0.84	0.78	0.68	0.61
	TA-Policy	1.0	1.0	1.0	0.98	0.81	0.81	0.71	0.66
	DP-Policy	1.0	1.0	0.94	0.91	1.0	1.0	0.91	0.88
Extra Time	ORCA	4.54	6.84	10.56	15.65	6.46	10.69	15.64	20.64
	Pre-Policy	3.25	5.26	7.63	10.63	7.56	10.32	14.44	17.24
	TA-Policy	3.25	4.26	6.21	8.54	6.94	9.24	11.62	14.35
	DP-Policy	4.18	6.45	9.33	10.98	4.89	7.12	9.69	11.84

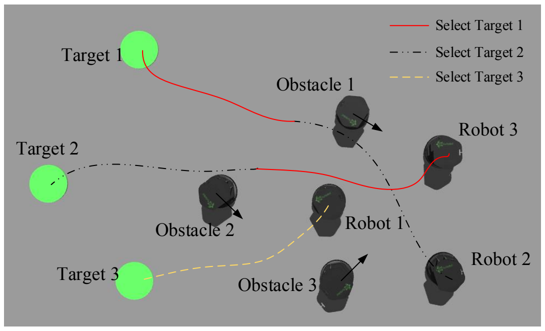

Fig. 4. Simulation scenario

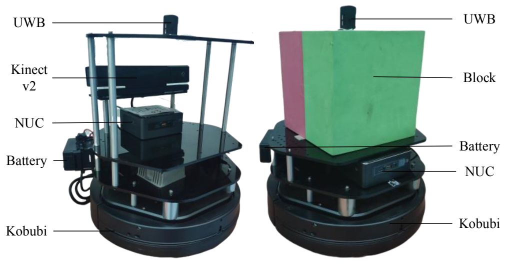

Fig. 5. Configuration of real robots and obstacles

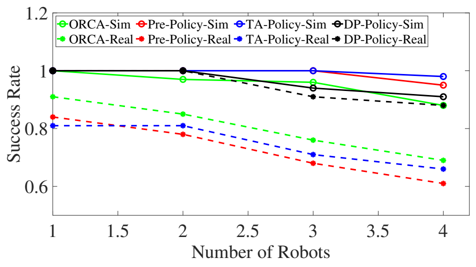

Fig. 6. The comparison of various policies in terms of success rate

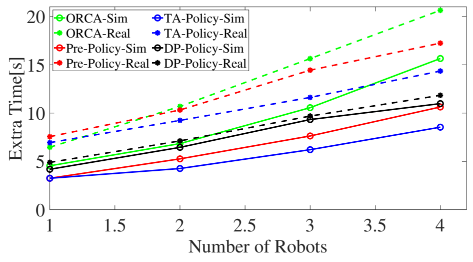

Fig. 7. The comparison of various policies in terms of extra time

all robots and obstacles. The state of the dynamic obstacles is derived from the fusion of UWB and Kinect v2, which is an RGB-D camera. The master control is a mini PC called Intel NUC kit to run the policy in the ROS architecture.

C. Results and Discussions

To validate the performance, we use two performance metrics: success rate and extra time [15]. The success rate is the ratio of the number of robots arriving at their goal positions without any collision over the total number of robots. Extra time measures the difference between the average travel time over all robots and the lower bound of the travel time. We compare four policies, including ORCA [5], Pre-Policy, TA-Policy, and DP-Policy. ORCA is a popular VO-based policy used as a baseline for performance comparison. These policies are tested in the simulator and the real world for 30 times with 3 moving obstacles and the robots with the number from 1 to 4 in each case. The paths of 3 obstacles are randomized with a fixed velocity.

The results are listed in Table III. In the simulation experiments, the TA-Policy has the best performance. Compared to the Pre-Policy, its performance of extra time decreases by about 14.32 percent on average. However, in the real world, its performance decreases heavily, about a 24.91 percent decrease in the success rate and a 96.39 percent increase in extra time. The performance decrease occurs on other policies except for the DP-Policy, which has a only 1.5 percent decrease and a 10 percent increase as shown in Fig. 6 and Fig. 7. Although the DP-Policy takes more travel time in the simulator, it has the slightest performance decrease in the real world. Thus, it has the best performance in terms of the success rate and extra time in the real world. Specifically, the performance of ORCA in the real world decreases a lot because this approach is sensitive to the uncertainty in the location readings from the UWB tag, leading to serious collisions. The results demonstrate the superior performance of our policy in both the simulator and real world.

VIII. Conclusion

In this paper, a DRL based navigation policy for multiple robots in a dynamic environment is proposed. All the robots in the same network can exchange information, and the target location allocation policy is proposed depending on the total distance. The PPO algorithm for multi-robot is developed to train the navigation policy based on the accumulative information collected by all robots in a simulator. The randomized dynamics parameters are used during the training process to make policy adaptive to the real world without any requirement of real data collection. The experiment results show that the target location allocation algorithm can save extra travel time about 14.32 percent in comparison with other methods. With the randomized dynamics parameters, the performance deterioration in real-world applications is alleviated from a 24.91 percent decrease in the success rate and 96.36 percent increase in travel time to 1.62 percent and 9.76 percent, respectively. Future work includes addressing more complex dynamic environments and using fewer external sensors. Moreover, a distributed training framework will be developed to reduce the training time.

References

[1] R. N. Darmanin and M. K. Bugeja, “A review on multi-robot systems categorised by application domain,” in *2017 25th Mediterranean Conference on Control and Automation (MED)*. IEEE, 2017, pp. 701–706.

[2] H. Durrant-Whyte and T. Bailey, “Simultaneous localization and mapping: part i,” *IEEE robotics & automation magazine*, vol. 13, no. 2, pp. 99–110, 2006.

[3] D. Fox, J. Ko, K. Konolige, B. Limketkai, D. Schulz, and B. Stewart, “Distributed multirobot exploration and mapping,” *Proceedings of the IEEE*, vol. 94, no. 7, pp. 1325–1339, 2006.

[4] R. Reid, A. Cann, C. Meiklejohn, L. Poli, A. Boeing, and T. Braunl, “Cooperative multi-robot navigation, exploration, mapping and object detection with ros,” in *2013 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 2013, pp. 1083–1088.

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

[6] D. Bareiss and J. van den Berg, “Generalized reciprocal collision avoidance,” *The International Journal of Robotics Research*, vol. 34, no. 12, pp. 1501–1514, 2015.

[7] J. E. Godoy, I. Karamouzas, S. J. Guy, and M. Gini, “Implicit coordination in crowded multi-agent navigation,” in *Thirtieth AAAI Conference on Artificial Intelligence*, 2016.

[8] T. T. Nguyen, N. D. Nguyen, and S. Nahavandi, “Deep reinforcement learning for multi-agent systems: A review of challenges, solutions and applications,” *arXiv preprint arXiv:1812.11794*, 2018.

[9] P. Long, W. Liu, and J. Pan, “Deep-learned collision avoidance policy for distributed multiagent navigation,” *IEEE Robotics and Automation Letters*, vol. 2, no. 2, pp. 656–663, 2017.

[10] Y. F. Chen, M. Liu, M. Everett, and J. P. How, “Decentralized non-communicating multiagent collision avoidance with deep reinforcement learning,” in *2017 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2017, pp. 285–292.

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

[12] M. A. K. Jaradat, M. Al-Rousan, and L. Quadan, “Reinforcement based mobile robot navigation in dynamic environment,” *Robotics and Computer-Integrated Manufacturing*, vol. 27, no. 1, pp. 135–149, 2011.

[13] J. Lin, X. Yang, P. Zheng, and H. Cheng, “End-to-end decentralized multi-robot navigation in unknown complex environments via deep reinforcement learning,” in *2019 IEEE International Conference on Mechatronics and Automation (ICMA)*. IEEE, 2019, pp. 2493–2500.

[14] J. K. Gupta, M. Egorov, and M. Kochenderfer, “Cooperative multi-agent control using deep reinforcement learning,” in *International Conference on Autonomous Agents and Multiagent Systems*. Springer, 2017, pp. 66–83.

[15] P. Long, T. Fanl, X. Liao, W. Liu, H. Zhang, and J. Pan, “Towards optimally decentralized multi-robot collision avoidance via deep reinforcement learning,” in *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2018, pp. 6252–6259.

[16] P. Christiano, Z. Shah, I. Mordatch, J. Schneider, T. Blackwell, J. Tobin, P. Abbeel, and W. Zaremba, “Transfer from simulation to real world through learning deep inverse dynamics model,” *arXiv preprint arXiv:1610.03518*, 2016.

[17] A. A. Rusu, M. Vecerik, T. Rothörl, N. Heess, R. Pascanu, and R. Hadsell, “Sim-to-real robot learning from pixels with progressive nets,” *arXiv preprint arXiv:1610.04286*, 2016.

[18] R. Lowe, Y. Wu, A. Tamar, J. Harb, O. P. Abbeel, and I. Mordatch, “Multi-agent actor-critic for mixed cooperative-competitive environments,” in *Advances in Neural Information Processing Systems*, 2017, pp. 6379–6390.

[19] Y. Jin, Y. Zhang, J. Yuan, and X. Zhang, “Efficient multi-agent cooperative navigation in unknown environments with interlaced deep reinforcement learning,” in *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2019, pp. 2897–2901.

[20] X. B. Peng, M. Andrychowicz, W. Zaremba, and P. Abbeel, “Sim-to-real transfer of robotic control with dynamics randomization,” in *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2018, pp. 1–8.

[21] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, “Proximal policy optimization algorithms,” *arXiv preprint arXiv:1707.06347*, 2017.

[22] N. Heess, S. Sriram, J. Lemmon, J. Merel, G. Wayne, Y. Tassa, T. Erez, Z. Wang, S. Eslami, M. Riedmiller *et al.*, “Emergence of locomotion behaviours in rich environments,” *arXiv preprint arXiv:1707.02286*, 2017.

[23] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu, “Asynchronous methods for deep reinforcement learning,” in *International conference on machine learning*, 2016, pp. 1928–1937.

[24] I. Zamora, N. G. Lopez, V. M. Vilches, and A. H. Cordero, “Extending the openai gym for robotics: a toolkit for reinforcement learning using ros and gazebo,” *arXiv preprint arXiv:1608.05742*, 2016.