A Hierarchical Deep Reinforcement Learning Framework with High Efficiency and Generalization for Fast and Safe Navigation

Wei Zhu and Mitsuhiro Hayashibe

***Abstract*—We present a hierarchical deep reinforcement learning (DRL) framework with prominent sampling efficiency and sim-to-real transfer ability for fast and safe navigation: the low-level DRL policy enables the robot to move towards the target position and keep a safe distance to obstacles simultaneously; the high-level DRL policy is supplemented to further enhance the navigation safety. We select a waypoint located on the path from the robot to the ultimate goal as the sub-goal to reduce the state space and avoid sparse reward. Moreover, the path is generated based on either a local or a global map, which can significantly improve the sampling efficiency, safety, and generalization ability of the proposed DRL framework. Additionally, a target-directed representation for the action space can be derived based on the sub-goal to improve the motion efficiency and reduce the action space. In order to demonstrate the eminent sampling efficiency, motion performance, obstacle avoidance, and generalization ability of the proposed framework, we implement sufficient comparisons with the non-learning navigation methods and DRL-based baselines, with videos, data, code, and other supplemental material shown on our website[1].**

***Index Terms*—Hierarchical deep reinforcement learning, sampling efficiency, fast and safe navigation, sim-to-real.**

I. Introduction

FOR autonomous navigation issues, mobile robots are required to get close to the target and keep safe distance to obstacles. To this end, either the traditional non-learning map-based methods, such as dynamic window approach (DWA) [1], or the state-of-the-art deep reinforcement learning (DRL) frameworks [2]–[5], need comprehensive evaluation criteria that can balance the importance of obstacle avoidance and target reaching. It is however challenging to adjust the weights of the factors used in these criteria. Therefore, we propose a hierarchical DRL-based motion planner that can comprehensively take these criteria into consideration and achieve fast and safe navigation. Moreover, the motion planner acquired by the DRL framework is able to generate a novel and optimal navigation policy by interacting with environments whereas the traditional non-learning approaches are inclined to make mobile robots move along the pre-established trajectories which may be non-optimal with respect of path length and motion efficiency.

The representation for the action space, state space, and immediate reward function of DRL frameworks can make a significant difference to the sampling efficiency and generalization ability. For the DRL-based navigation, the action space is generally composed by linear and angular velocities [2], [3]. However, randomly generating either continuous or discrete velocities during the DRL training process may degrade the sampling and learning efficiency. Furthermore, the navigation environments are generally complicated because of the diversified locations, sizes, and shapes of obstacles, the time-varying robot states, and the different goal positions. Therefore, it is especially difficult to represent the state space and reward function for DRL frameworks. For instance, the data from either LiDAR or vision sensors used to detect obstacles have high dimensions and broad ranges [4]. When the final goal is far from the mobile robot, the reward function would be extremely sparse, which makes it prohibitively difficult to fully explore the environment. Furthermore, the complex obstacle distribution between the robot and the final goal makes it challenging to reasonably and comprehensively define the reward function. Even though the well-learned navigation policy can be successfully deployed on the same or similar environments as the training scenarios, it is tremendously difficult to generalize the policy into more variant environments.

To improve the sampling efficiency and generalization ability of the DRL model, we select a waypoint (namely sub-goal) which is close to the robot and located on a feasible path from the robot to the final goal. The definitions of DRL elements are based on the sub-goal rather the final goal. More specifically, the path is planned using either a fully-known global map or a real-time local map that is updated and maintained with the simultaneous localization and mapping (SLAM) algorithm [6]. Since the mobile robot explores the environment during the DRL training, it is reasonable and natural to update a map for path generation, thus to select an accessible waypoint for reducing the representation complexity of DRL elements. Since we only consider the neighboring surroundings around the robot and encourage the robot to move towards an accessible sub-goal, the generalization ability of the DRL framework is significantly improved. Based on the sub-goal, we define a target-directed representation for the action space that can further improve the motion efficiency for navigation and the sampling efficiency for the DRL model.

This work was supported in part by the Japan Society for the Promotion of Science (JSPS) KAKENHI under Grant JP20KK0256.

The authors are with the Department of Robotics, Graduate School of Engineering, Tohoku University, 980-8579, Sendai, Japan. zhu.wei.r5@dc.tohoku.ac.jp

[1]https://github.com/zw199502/RL_navigation

It is generally challenging for sim-to-real transfer even though the DRL-based navigation methods have achieved some real applications [2]–[4] because simulation environments are far different from real scenarios. The real mobile robots used to validate the proposed DRL framework are a wheeled bipedal robot [7] and a quadruped robot, whose dynamics are completely different from the dynamics of the differential mobile robot used in the simulation. In order to achieve the sim-to-real transfer, we add noises to the action during the training to learn a policy that can deal with uncertainties and fine-tune the acceleration for both the simulated and real mobile robots because of their dynamic differences. Additionally, we also add mild noises to the action during validations in case of the frozen motion in certain extreme situations.

The main contributions of this work are: (1) creating a hierarchical DRL framework that can comprehensively consider fast and safe navigation; (2) introducing a sub-goal which is able to improve the sampling efficiency and generalization ability for the DRL model; (3) deploying the simulated policy on physical robots in the real world; (4) making our project open-sourced one the website[1].

II. RELATED WORKS

There is a large body of work on navigation for wheeled mobile robots, using either LiDAR or vision sensors [8], [9]. For traditional non-learning map-based methods [10], [11], a pre-built global map is required for path planning. However, mapping is generally cumbersome when environments frequently change. When the global map is already known, a plenty of path planning literature can either produce a global path from the initial robot position to the final goal such as the A-star algorithm [12] and the rapidly-exploring random tree (RRT) method [13], or yield a local planner that can generate velocity commands and handle certain disturbances such as moving, removing or adding obstacles. Among the local planners, the dynamic window approach (DWA) as well as its variations is widely leveraged for obstacle avoidance [1], [14]. Nevertheless, the commonly-used global planners are unable to generate optimal paths because they search a feasible path from the map via a heuristic method. Moreover, the local planners need careful tuning for the coupled factors involving the cost weights of collision, path following, motion speed, etc.. For certain simple situations, if the environment does not change and the global path avoids all obstacles, some path tracking methods can be directly used to follow the path without the concern of collision [15], [16]. The aforementioned methods assume that a global map is already known, which is impractical when frequently changing the environment. One alternative is to simultaneously update a map in real time and navigate in the partially-known surroundings [17], which solves the problem of off-line time-consuming map building process at the expense of degrading the safety and optimality of navigation. Another solution is completely free of maps. In [18], a motion planner can generate collision-free action by the current and predicted motion states of surrounding obstacles. For dynamic environments, the optimal reciprocal collision approach (ORCA) [19] and its extensions [20], [21] can ensure safe navigation but rely heavily on the overall information of the surroundings.

Recently, deep learning, especially DRL-based approaches have been widely applied for map-free navigation because of the powerful representation capability of deep neural networks and the strong exploration, interaction and self-learning ability of reinforcement learning (RL). In [22], a deep neural network structure is used to predict future relevant events, such as collision and terrain properties. The off-line gathered data with LiDAR and IMU sensors are required to update the deep neural networks. Similar collision prediction method is leveraged for safe navigation in [23]. However, collecting supervised data is arduous and it is challenging to generalize the prediction model in completely different scenarios because the off-line data are generally gathered from specific environments.

To improve the generalization ability and avoid manually collecting supervised data, researchers are focusing on autonomous interaction with environments and self-learning approaches, especially on RL frameworks. In [24], the DRL training is firstly implemented in simulation, where a great deal of vision data can be gathered to optimize the navigation strategy, then the simulated policy is directly deployed in the real world with similar settings as the simulation. Similarly, ref. [25] successfully transfers the learned policy into novel scenarios by approximating the reward function with a linear combination of learned features. Besides the vision-based DRL navigation policies which are sensitive to light intensity, LiDAR sensors are also broadly leveraged to gather environment information [2], [3], [26]. One crucial problem for these DRL frameworks is the inefficient sampling and learning because of the complexity of environments. Therefore, some studies are aimed for improving the sampling efficiency. For instance, ref. [27] leverages the prior demonstrations and ref. [28] combines the model-based and model-free RL frameworks. However, the generalization capability is deteriorated because of the dependency on specific environments. Furthermore, it is still challenging to transfer the DRL-based navigation frameworks from simulation to the real world because the uncertainties in real scenarios can not be accurately represented in simulation. Moreover, most of real applications utilize wheeled mobile robots with stable motion properties [2], [3], [24]. Some sim-to-real applications are implemented on humanoid robots with omni-directional motion ability [29], [30]. However, it is rare to deploy the DRL-based navigation policy on wheel bipedal robots with unstable motion characteristics [7]. In this work, we achieved the sim-to-real transfer on a physical wheeled bipedal robot with unstable motion and a quadruped robot.

III. HIERARCHICAL DRL FRAMEWORK

We are aimed to propose a deep reinforcement learning approach with high training efficiency and generalization ability in respect to different surroundings and robot platforms for fast and safe navigation in complex environments. To this end, a two-layer DRL framework shown in Fig. 1 is constructed, wherein the low-level DRL policy is responsible for producing quick motion while the high-level DRL policy is supplemented

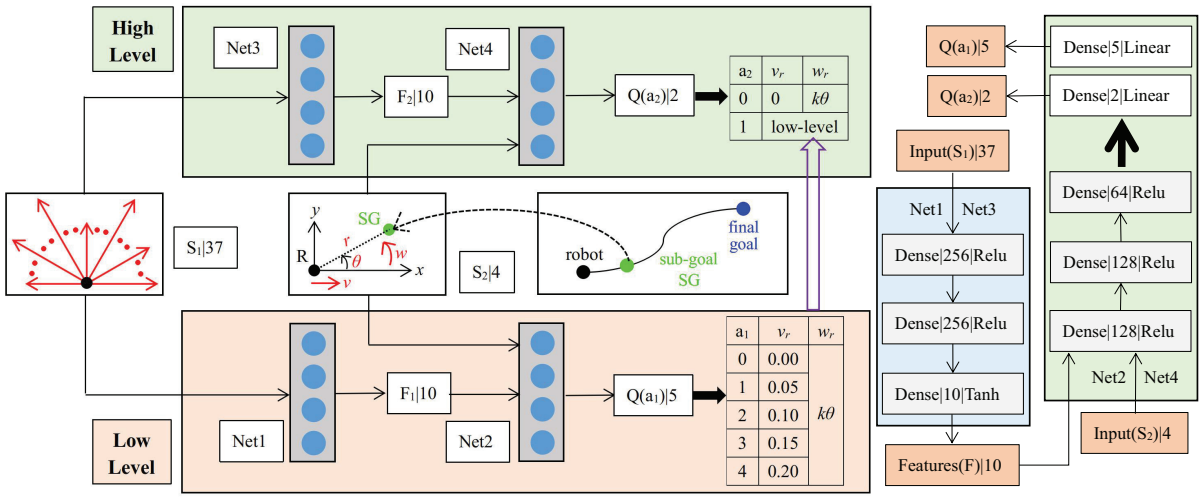


Fig. 1: Hierarchical DRL framework. The low-level DRL policy is used for fast motion while the high-level DRL strategy is aimed at safe obstacle avoidance. Both the low- and high- level DRL policies have the same deep neural network structure and share the identical state input including a 37-dimension laser scan, robot's linear and angular velocities (v and w), and the position of the sub-goal in the robot frame (r and θ). Differently, the low-level DRL policy outputs five specific actions while the high-level DRL policy produces two abstract choices, one of which projects to the low-level DRL policy.

to further improve safety of obstacle avoidance. Because we choose the sub-goal which is a waypoint located on the path from the robot to the final goal, the observation space of RL is significantly narrowed. Moreover, the path is generated via traditional global path planning approaches which take consideration of both obstacles and the final goal position, the sampling space is therefore further reduced. Thanks to the introduction of the sub-goal, the training efficiency of our DRL framework is far higher than pure DRL methods. Besides the reduction of observation space, the scope of exploratory action is narrowed because the deep neural networks only output discrete linear velocity while the angular velocity is proportional to the orientation of the sub-goal in the robot frame. Additionally, the proposed DRL framework has great generalization ability not only for various environments but also for different robot platforms for three reasons: (1) a sub-goal located on a feasible path is selected to represent the DRL elements; (2) one part of observation is a high dimensional laser scan whose features are extracted via deep neural networks; (3) the actions are generalized linear and angular velocities which are further transformed into actuator commands.

A. Problem Formulation

For the navigation task in environments with dense obstacles, the mobile robot needs to make decisions to avoid obstacles and reach targets according to the states of the robot as well as the surrounding information gathered by attached sensors. We formulate this task as a Markov decision process (MDP) defined by a tuple $< S, A, T, R, \gamma >$. S is the state space including the states of the robot, surrounding obstacles, and the target position in the robot frame. A stands for the action space which is composed of the linear and angular velocities of the mobile robot. T represents the state transition, that is the next state $\boldsymbol{s}'$ is generated by the current action $\boldsymbol{a}$ and state $\boldsymbol{s}$. Such a state transition is implied by a physical simulator Gazebo. R, with the actual value r, denotes the immediate reward after executing $\boldsymbol{a}$ in $\boldsymbol{s}$, which is related to the distance from the robot to its surrounding obstacles and the distance from the robot to the target point. γ is the discount factor that indicates the changing reward in the time domain. The goal of this task is to acquire a policy π to maximize the expectation of the long-term cumulative reward $V_\pi(\boldsymbol{s})$:

$$\begin{aligned} \pi^* &= \underset{\pi}{\operatorname{argmax}} V_\pi(\boldsymbol{s}), \\ V_\pi(\boldsymbol{s}) &= E_\pi\left[\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} | S_t = \boldsymbol{s}\right] \\ &= \sum_{\boldsymbol{a}\in A} \pi(\boldsymbol{a}|\boldsymbol{s}) Q_\pi(\boldsymbol{s}, \boldsymbol{a}), \end{aligned} \tag{1}$$

where Q is the action-value function. We leverage Deep Q-learning networks (DQNs) to formulate this optimization problem [31].

B. Low-level DRL

State space. Unlike the existing researches which use sparse laser scan as one part of state space, such as ref. [2] utilizes a 10-dimensional laser scan and ref. [3] selects a 13-dimensional laser scan, we choose a dense 37-dimensional laser scan $\boldsymbol{s}_l$ to extensively perceive surroundings. Moreover, instead of directly using this scan like ref. [2], [3], we pre-process it with deep neural networks, then 10 abstract features are extracted so as to generalize surroundings. For the feature extraction, we utilize deep neural networks to map the relatively high-dimension raw laser scan to a low-dimension feature vector which can indicate the abstract obstacle features with the possibly authentic collision information. In fact, we can increase the dimensions of the raw laser scan with more details at the

expense of more complex neural networks and longer training time. Moreover, rather than directly reaching the final goal, the robot is required to move close to the waypoint (namely sub-goal) one by one and gradually arrive at the ultimate goal location. Consequently, we define another part of the state space as the distance r from the sub-goal to the robot and the sub-goal orientation θ relative to the moving direction of the robot. In this case, the state space can be tremendously narrowed. Finally, we regard the current linear and angular velocities v and w of the robot as the third part of state because they are likely to predict future states of the robot. Therefore, the ultimate state $\boldsymbol{s}$ is defined as follows:

$$\boldsymbol{s} = \{\boldsymbol{s}_l|37, r, \theta, v, w\} \in S. \tag{2}$$

To reduce the state space thus to improve training efficiency of DRL, we add certain constraints for these state elements. Firstly, the robot only needs to take close obstacles into consideration for local motion planning, so the maximum range of each laser scan beam is set as $l_{\max}$. Secondly, the maximum distance is defined as $r_{\max}$ with details introduced in Section III-D and the orientation of the sub-goal θ is free on the whole 2D plane. Finally, the ranges of v and w are limited according to the definition of the action space.

Action space. For the DRL-based navigation, the actions are generally either continuous or discrete linear and angular velocities of mobile robots [2], [3]. Randomly generating these two velocities during the trial and error process may cause inefficient and invalid sampling, thereby degrading the learning efficiency of DRL models. To improve the sampling efficiency, we propose a target-directed representation form for the actions, which is defined by two parts. (1) The angular velocity w is unrelated to the action space of DRL but proportional to the sub-goal orientation θ, which is represented as follows:

$$w = k\theta, \tag{3}$$

with k being a positive control gain. Such a traditional P controller can exclude one action variable from the DRL model, thereby exponentially reducing the exploratory action space for the DRL framework. Moreover, the motion orientation of the mobile robot is directed to the sub-goal, thus to significantly improve the motion efficiency and avoid certain invalid angular velocities. (2) The action space of RL only consists of five non-negative discrete linear velocities which is defined as follows:

$$a_l \in \{0, 0.05, 0.1, 0.15, 0.2\} m/s \in A_l. \tag{4}$$

We assume the robot does not move backward and the maximum translation speed is $0.2m/s$ for the mobile robots we use in simulation and real worlds. Please note that we will decrease the maximum translation speed for the wheeled bipedal robot for stable motion and balance keeping. In this work, we define the action as discrete choices. Please note that continuous action can be beneficial for more smooth motion but an action network is required such as the deterministic deep policy gradient (DDPG) algorithm [32] and the soft-actor-critic (SAC) framework [33]. The discrete action is commonly used in the value-based reinforcement learning frameworks such as the DQN algorithm we utilize in this work. Although the discrete action may cause less smooth motion, the network framework is more simple. We choose the discrete action space only for simplicity. We think the continuous action can also work for the navigation task.

Reward function. The navigation objective is to simultaneously avoid obstacles and move close to the target point. Let r_o be the minimum distance from the robot to its surrounding obstacles which is measured by the laser sensor. Let r_t be the current distance from the robot to the sub-goal and r_{t-1} be the last distance. Let r_c be the collision distance. Then, we define the reward function R_l as below:

$$R_l = R_{lc} + R_{lg} + R_{la} \tag{5}$$

$$R_{lc} = \begin{cases} -200 & r_o < r_c & \text{(6a)} \\ -0.05 r_c / r_o & r_c \leqslant r_o \leqslant l_{\max} & \text{(6b)} \\ 0 & r_o > l_{\max} & \text{(6c)} \end{cases}$$

$$R_{lg} = \begin{cases} 0 & r_c > r_o & \text{(7a)} \\ 180(r_{t-1} - r_t) & r_c \leqslant r_o & \text{(7b)} \end{cases}$$

$$R_{la} = \begin{cases} 0 & a_l = 0.1, 0.15, 0.2 & \text{(8a)} \\ -4(0.1 - a_l) & a_l = 0, 0.05 & \text{(8b)} \end{cases}$$

R_{lc} is the low-level collision reward, R_{lg} represents the goal distance reward, and R_{la} stands for the action reward. On the one hand, if the robot collides with obstacles, a large punishment with -200 will be given. Furthermore, we also mildly punish the behavior which leads to approaching obstacles. On the other hand, a positive reward will be obtained if the robot gets closer to the sub goal. To avoid rotation in place or slow motion, we discourage small linear velocities. To summarize, the definition of reward function can simultaneously encourage the robot to move close to the sub-goal as fast as possible and keep safe distance from obstacles.

Neural networks. Instead of directly using the 37-dimensional laser scan, we firstly process it with a three-layer neural network framework displayed in Fig. 1 to extract a low-dimensional feature vector which can potentially describe the information about the free and occupied space. Then, we combine this feature vector with other four state elements in (2) to construct a new input for another three-layer neural network framework shown in Fig. 1, which finally outputs five Q values corresponding to the five discrete linear speeds. During the training, the discount factor γ is 0.99, the learning rate α is set as 0.0001, and the batch size is 64, respectively.

C. High-level DRL

When we evaluated the pre-learned low-level policy, we found that the robot tended to collide with obstacles if the orientation of the sub-goal θ is far from zero. We think such a situation is caused by the long turning arc length corresponding to the large θ. For safer navigation, we replenish a high-level DRL framework that can learn a policy to avoid such collisions while keeping the motion efficiency of the low-level DRL framework.

State space, neural networks and action space. The description of the high-level DRL framework for the state space and neural networks is identical to that of the low-level

framework while the definition of action space is different. The low-level policy is learned in advance and then regarded as one possible action for the high-level DRL policy. Moreover, we individually select the first element from the low-level DRL action space (4) as the other element to further improve the safety of obstacle avoidance. Therefore, the action space of the high-level DRL framework is summarized as below:

$$a_h \in \{0, 1\} \in A_h. \tag{9}$$

If $a_h = 0$, the linear velocity is zero. When $a_h = 1$, the linear velocity is generated by the pre-learned low-level policy. In both cases, the angular velocity is same as (3). Although the case of $a_h = 0$ may degrade the motion efficiency because the robot only rotates in place, it enables the robot to avoid the collisions caused by certain large turning arc length.

Reward function. Because the high-level policy is aimed for safer navigation while keeping the motion efficiency of the low-level policy, we therefore define the reward function as below:

$$R_h = R_{hc} + R_{ha} \tag{10}$$

$$R_{hc} = \begin{cases} -200 & r_o < r_c \quad (11a) \\ -1.2 r_c / r_o & r_c \leqslant r_o \leqslant l_{\max} \quad (11b) \\ 0 & r_o > l_{\max} \quad (11c) \end{cases}$$

$$R_{ha} = \begin{cases} 0 & a_h = 1, r_c \leqslant r_o \quad (12a) \\ -r_o & a_h = 0, r_c \leqslant r_o \quad (12b) \end{cases}$$

R_{hc} is the high-level collision reward while R_{ha} represents the action reward. Same as the R_l defined in (5), we heavily punish the behavior which causes collisions. But differently, we magnify the factor from 0.05 to 1.2 so as to further make the robot get far from obstacles. The action reward in (12) is used for avoiding possible long-time rotation in place and frozen motion, thereby maintaining the motion efficiency of the low-level policy.

D. Selection of Sub-goal

One of the key contributions in this work is the selection of sub-goal. Most of navigation studies with DRL are final-goal directed [2]–[4], that is the state space S and the immediate reward R are related to the final goal, which causes several serious flaws. Firstly, if the final goal is far from the robot, it will be extremely difficult to thoroughly explore all possible state space because covering the whole state space generally requires prohibitively long interaction with the environment. Secondly, it is challenging to reasonably define the reward function for two main reasons. (1) The large exploration space may cause exceptionally sparse reward distribution, thereby seriously influencing the sampling efficiency. (2) Most of studies directly define the reward function based on the straight line length from the robot to the final goal while ignoring the possible effects of the obstacles between the robot and the final goal. We also found that the robot tended to collide with obstacles when using such kind of reward functions. Thirdly, directly using final goal may potentially increase the representation complexity of the action space A because the policy networks, which generally project states into actions, are required for actor-critic (AC) based RL frameworks. In this case, the sampling efficiency is further deteriorated.

To effectively solve these problems, we select a sub-goal which is close to the robot rather than directly using the final goal. Instead of discarding past laser scans like how most of studies do [2]–[4], we utilize these historic laser data to update a grid occupation map with the SLAM algorithm [6]. Then we can generate a global path from the robot to the final goal using global path planners. The path is generally non-optimal because of the non-fully explored surroundings but feasible for the local utilization. Therefore, we can select a waypoint which is close to the robot from this path at the beginning of each training iteration. Through moving close to the waypoint one by one, the robot can successfully reach the final target. Because both the SLAM algorithm and the path planning method are general for diversified environments, the proposed hierarchical DRL framework can be directly applied in the new scenarios which are never explored ahead during the training. Besides, the waypoint significantly narrows the state space and avoids the sparse reward, which enables fast learning for the DRL model.

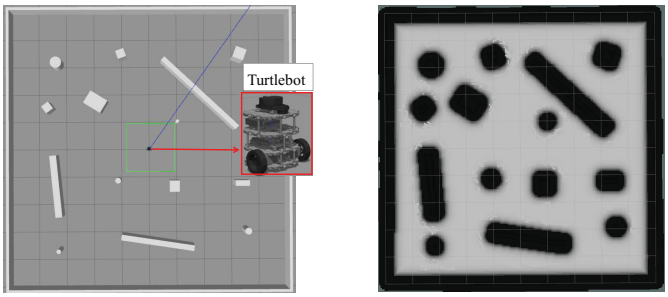
IV. EXPERIMENTS

To validate the predominant sampling efficiency, fast motion and eminent success rate of target reaching, and high generalization in respect of different robot platforms and surroundings for the proposed hierarchical DRL framework, we respectively implemented corresponding experiments and compared with existing approaches, with the overall video shown on the website[2].

A. Auxiliary Implementation Tools

To avoid frequently updating and resetting a non-fully-known global map during the training process, we assume that the global map is already off-line built utilizing the SLAM algorithm with the ROS gmapping package[3], and then we can on-line plan a global path with the ROS A-star package[4] and select a waypoint close to the robot as the sub-goal for improving the sampling efficiency. After training, we testified that the learned policy could be successfully deployed as well in the environments without off-line pre-built global maps. For instance, all real implementations are free of off-line global maps, thus our method can be quickly deployed in various real worlds without pre-building global maps by manual operations. In simulation, we obtain the odometer information with Gazebo-ROS APIs while it is collected with the internal wheel odometry when operating the wheel bipedal robot, and the external motion capture system for the quadruped robot, respectively. For the wheel odometry, we leverage the ROS extended Kalman filter (EKF) package[5] to reduce the drift by fusing the IMU data and the wheel odometer information. For the traditional non-learning based comparison method, we utilize the ROS move_base package[6].

[2]https://youtu.be/S95BQDEiTE0
[3]http://wiki.ros.org/gmapping
[4]http://wiki.ros.org/nav_core
[5]http://wiki.ros.org/robot_localization
[6]http://wiki.ros.org/move_base



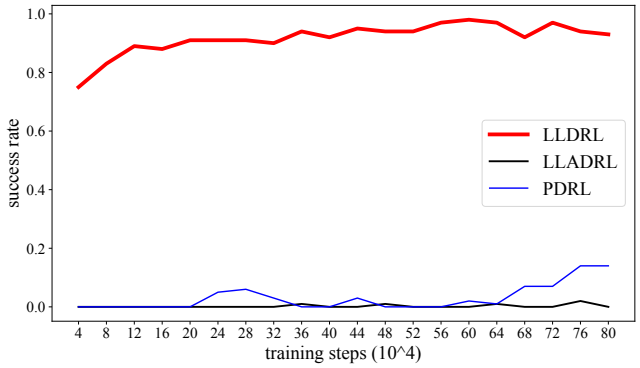
(a) Gazebo visualization (b) Occupation map

Fig. 2: Training environment. The Turtlebot mobile robot moves at 10Hz control frequency in a rectangle with 10×10m size. The laser sensor, with 10Hz scanning frequency, is installed on the top of Turtlebot. The left figure visualizes the Gazebo simulation environment while the right figure displays the occupation map pre-built with the SLAM algorithm.

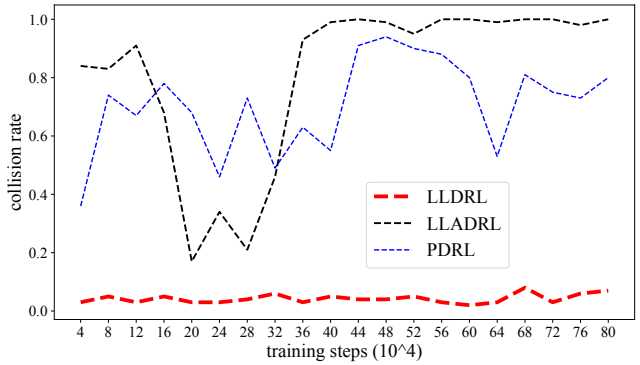
B. Training Environment Settings

We use Gazebo as the physical simulator. The size, shape, and location of obstacles are randomized in a 10×10m enclosed rectangle shown in Fig. 2. At the beginning of each training iteration, the robot pose and the final goal position are randomly chosen from the free space of the global map to explore the environment as thoroughly as possible. The laser sensor, with 901 beams ranging from $-\pi/2$ to $\pi/2$ rad relative to the translation direction of the robot, is attached on the top of the robot. All 901 laser beams are used for updating the minimum distance (r_o) from the robot to its surrounding obstacles and building a global map with the SLAM algorithm. 37 evenly spaced laser beams are selected for composing the state space of the DRL model. We assume that the global map is already off-line built with the SLAM algorithm, and then we can on-line plan a global path and select a waypoint close to the robot as the sub-goal for improving the sampling efficiency. The control frequency is 10Hz which is the maximum sampling frequency of the real LiDAR sensor we use in the real world. The high-level and the low-level DQN policies are executed at 10Hz. Empirically, we update the sub-goal every 2 seconds. Moreover, the global path is also updated every 2 seconds because we only partially know the global environment.

For better comparison with baselines, we train all DRL frameworks 800 thousand steps. Because the control frequency is 10Hz, the theoretical time scale is about 22 hours. However, the real training time is around 7 days because the simulation is paused after each step for updating the deep neural networks. We found that updating policy cost tens of milliseconds while pausing and restarting the Gazebo took prohibitively long time (about half second), which is the major reason of the time-consuming training. Besides, unlike Mujoco, the Gazebo simulation can not be accelerated even runs with more time than the theoretical time because of the computer hardware limitations, which is another reason of the long-time training.



(a) Success rate



(b) Collision rate

Fig. 3: The comparison of the training efficiency. 100 tests are executed every 40 thousand training steps. The test results include successful navigation, collision and time out that is the robot does not reach the final goal within a given time. The top figure shows the success rate while the bottom figure illustrates the collision rate.

C. Simulation Evaluations

Firstly, to validate the prominent sampling efficiency of the proposed low-level DRL (LLDRL) framework, we implemented a pure DRL (PDRL) baseline as the comparison. PDRL directly uses the final goal to represent the state space and the reward function. Moreover, to verify the significant effect of the action definition of our method, we introduced another DRL-based baseline which is similar to LLDRL except that the angular velocity is set as a discrete set, namely LLADRL, rather than the equation (3) utilized in the proposed approach. Because the goal is far from the robot, PDRL can not directly utilize the angular velocity defined in (3). Instead, the angular velocity is discretized as $-\pi/2 + \pi/6 \cdot i$ rad/s with $i = 0, 1, .., 6$. The linear velocity choice is same as the action space (4). Therefore, the action choices of PDRL are 35. The training result is shown in Fig. 3, where LLDRL only needs around 120 thousand training steps to reach a stable and high success rate of navigation while PDRL can only achieve 14% success rate even after 800 thousand training steps. Meanwhile, the LLADRL method yields the worst result, only with 2% success rate.

Secondly, we demonstrated that the proposed hierarchical DRL (HDRL) framework could encourage fast motion and

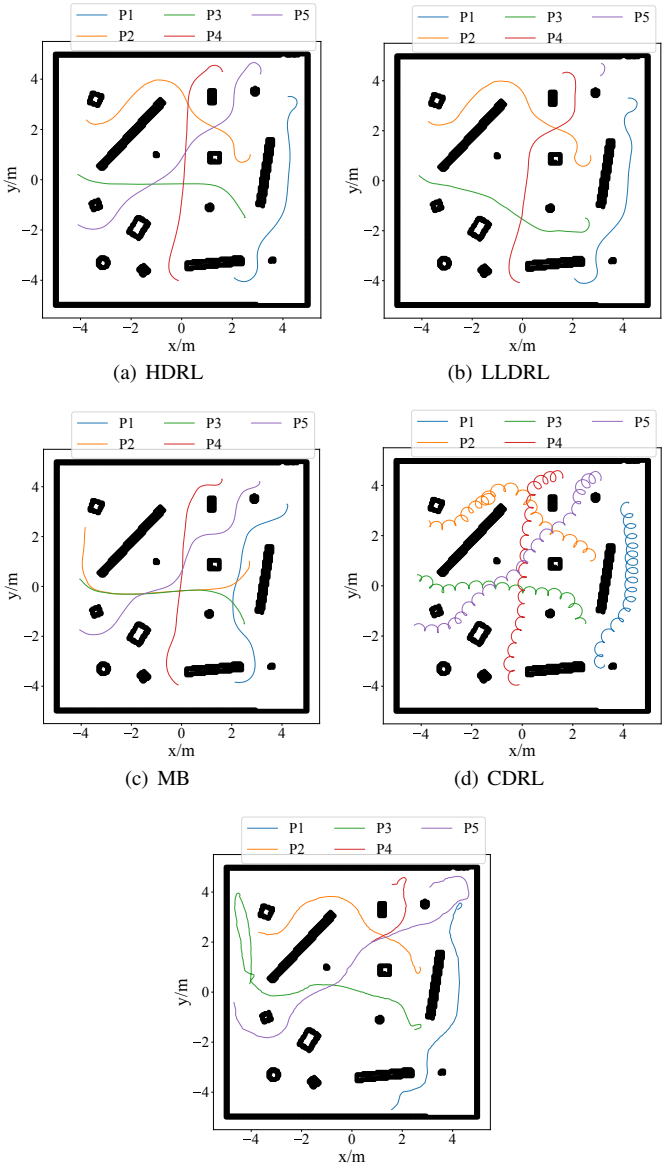
TABLE I: The comparisons of the success rate of target reaching and the motion efficiency. The average goal reaching time used to imply the motion efficiency is calculated only from the tests where all five methods succeed in the target reaching.

	HDRL	LLDRL	MB	CDRL	DDRL
success rate (%)	**100**	93	**100**	70	87
average time (s)	**28.0**	29.7	34.4	117.2	65.4

TABLE II: Average speed (unit: cm/s) of each path for the proposed method and the move_base approach.

	P1	P2	P3	P4	P5	Average
HDRL	**18.7**	**17.6**	**17.6**	**18.8**	**18.4**	**18.2**
MB	15.8	13.5	12.4	16.6	13.4	14.3

boost the success rate of target reaching by comparing with the traditional move_base[6]. (MB) navigation method, another two DRL-based baselines namely CDRL and DDRL respectively, and LLDRL. CDRL originates from [2] with continuous linear and angular velocities while DDRL is derived from [3] where the angular velocity is discrete and the linear velocity keeps constant. We tested these five frameworks 100 times in the same environment as the training one shown in Fig. 2(a). The comparison results are shown in TABLE. I, where HDRL achieved the highest success rate while taking the shortest average motion time. During the testing, we found that the robot tended to rotate in place when using the baseline CDRL, which caused the lowest success rate and took the longest average time for successful target reaching. The phenomenon of rotating in place also happened in [2]. We think that it is because of the action space of 2-dimensional continuous velocity that makes it difficult to quickly acquire a stable policy. The baseline DDRL achieved relatively higher success rate and shorter motion time due to the relatively simple action space, but it brought about frequent jitter on the down side, which degraded the motion efficiency and the target reaching performance. Additionally, the success rate was increased to 100% from 93% when using HDRL, which demonstrated that the high-level DRL policy did play a significant role for performance improvement. Even though MB achieved 100% target reaching, its motion efficiency was obviously lower than HDRL. We think that the move_base method excessively emphasizes the collision avoidance and path tracking at the expense of motion efficiency while our method acquires a novel and optimal motion policy during the training and learning process. For further visualizing comparisons, we selected five paths for each framework shown in Fig. 4. The paths in Fig. 4(d) displayed the phenomenon of rotation in space and the paths in Fig. 4(e) were less smooth than those paths in Fig. 4(a)-4(c). The initial turning arc length of the paths in Fig. 4(b) was generally longer than that of the paths in Fig. 4(a), which caused one failure shown in Fig. 4(b). The average motion speed of these five paths for HDRL and MB is shown in TABLE. II, which demonstrated that



(a) HDRL (b) LLDRL (c) MB (d) CDRL (e) DDRL

Fig. 4: Path visualization. The paths of LLDRL are similar to those of HDRL but the path five of LLDRL enters the fatal collision area. The paths of the baseline CDRL illustrate the inefficient motion of rotation in place while the paths of the baseline DDRL display the vibrant jitter which degrades the motion efficiency.

our method could generate faster motion than the traditional move_base approach even though both achieved smooth and safe navigation at the similar level.

Thirdly, the generalization ability in respect to various environments were proved by deploying the proposed hierarchical DRL policy in two completely diverse and complex scenarios (shown in Fig. 5) which had never been used for the DRL training before. We implemented 100 tests for each environment. The success rate of target reaching in the environment A was 99% and B reached 100%. We visualized 10 paths for each scenario displayed in Fig. 6. At the beginning of

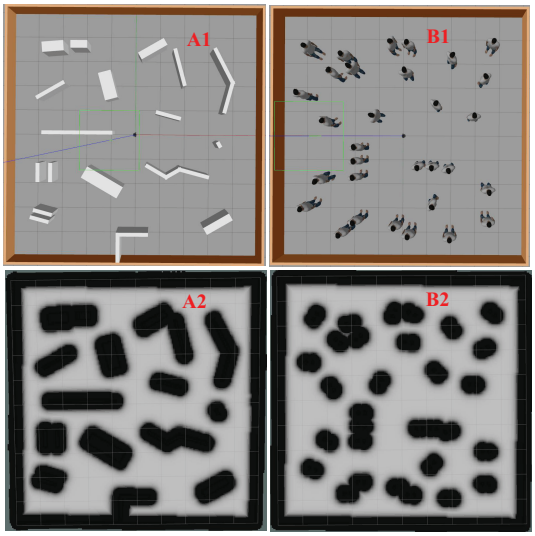
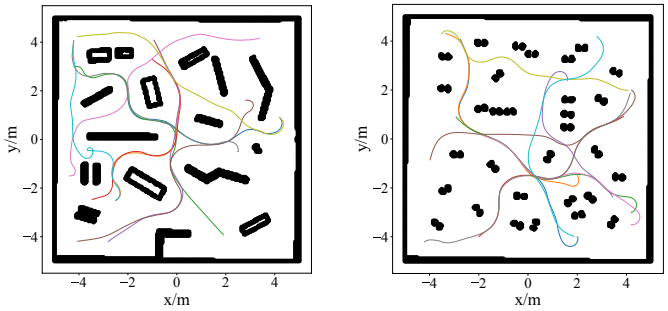


Fig. 5: New environments. Neither the two environments are explored with the proposed DRL framework. A1 and B1 display Gazebo scenarios while A2 and B2 are corresponding occupation maps built with the SLAM algorithm. Compared with the training environment shown in Fig. 2, the obstacles in these two environments are much more dense and exhibit more complex topological structure, such as the obstacle cluster.



(a) Paths in environment A (b) Paths in environment B

Fig. 6: Successful navigation paths in two novel worlds totally different from the training environment.

some paths, the curvature radius was very small in order to quickly avoid collision and keep motion efficiency because the initial orientation of the sub-goal deviated hugely from the moving direction of the robot. However, we still found that the robot collides with the obstacle at the very beginning in the environment A. We think the initial location of the robot is so close to obstacles that there is no enough time and distance space for collision avoidance.

D. Sim-to-Real Transfer

Finally, we implemented the hierarchical DRL framework on a physical wheeled bipedal robot and a quadruped robot to validate the capability of sim-to-real transfer and generalization ability in respect to different robot platforms shown in Fig. 7. The wheeled bipedal robot is developed by HEBI

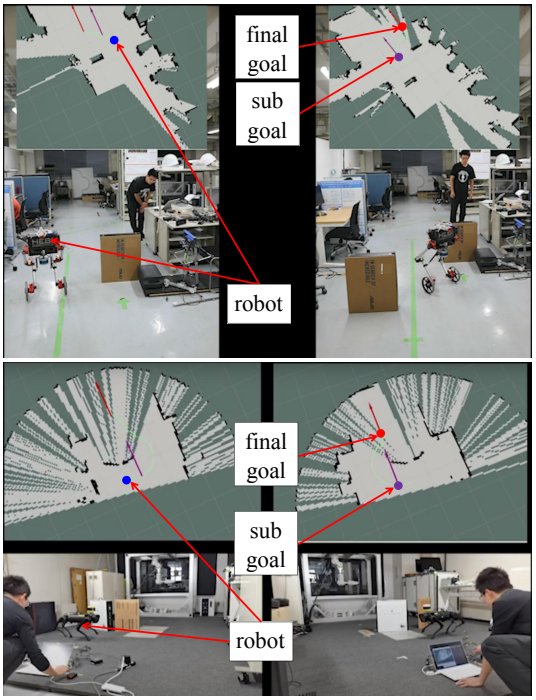


Fig. 7: Sim-to-real transfer in diverse scenarios with different robot platforms. The localization for the wheeled bipedal robot is based on the wheel odometer and the Kalman filter while the quadruped robot relies on the external motion capture because the embedded odometer drifts heavily.

robotics[7] and we installed a Velodyne LiDAR sensor under the robot chassis. We use the quadruped robot from unitree[8], with the LiDAR sensor attached on the robot head. Because the dynamics of the bipedal robot [34] is completely different from that of the wheeled mobile robot used in simulation, we separately trained a hierarchical DRL framework. More specifically, we still utilized the wheeled mobile robot for the training in simulation. Differently, the action space of the low-level DRL framework was reduced to three choices (0.0, 0.05, 0.1)m/s since the small linear velocity is beneficial for the balancing control of the bipedal robot. Moreover, we supplemented acceleration limitations for the linear and angular velocities because of the motion constraints of the wheeled bipedal robot. For the quadruped robot with the stable motion property, we directly deployed the simulated policy in the real world without any fine-tuning. As shown in Fig. 7, we tested the simulated DRL policy in diverse real scenarios with different robot platforms, where the global maps were initially unknown and simultaneously updated with the SLAM algorithm in real time. Thanks to the partially-known global map, a rough path from the robot to the final goal was planned and updated on-line. Based on this path, a sub-goal close to the robot was selected for the proposed DRL policy. Additionally, we performed another five practical experiments in more realistic scenarios to further demonstrate the generalization ability in the physical world, with the video

[7]https://www.hebirobotics.com/
[8]https://www.unitree.com/

shown on the website[9].

V. CONCLUSIONS

We presented a hierarchical deep reinforcement learning framework for navigation issues, with the low-level DRL policy being aimed for fast and safe navigation, and the high-level DRL strategy for further enhancing safety. The proposed framework demonstrated the eminent advantages in terms of high sampling efficiency. Moreover, the framework was able to achieve fast and safe navigation in diverse environments because we directly utilized the sub-goal close to the robot rather than the ultimate goal and proposed a target-directed representation for the action space based on the sub-goal. Additionally, our method could be deployed on different robot platforms even with complex dynamics and be transferred to real worlds without globally known maps.

These advantages were demonstrated by sufficient simulation comparisons and real implementations. Firstly, the sampling efficiency was validated by comparing with the pure DRL model. Secondly the motion efficiency and the collision avoidance safety were confirmed via comparing with the commonly-used move_base approach and another two DRL-based baselines. Finally, the generalization ability with respect to environments and robot platforms was affirmed through deploying the hierarchical DRL policy on novel and complicated environments using a wheeled mobile robot, a wheeled bipedal robot with complicated dynamics and a quadruped robot. For our current work, we assume that the environment is static, we therefore will try to extend our method in the dynamic environments with crowded pedestrians in future.

REFERENCES

[1] D. Fox, W. Burgard, and S. Thrun, "The dynamic window approach to collision avoidance," *IEEE Robotics and Automation Magazine*, vol. 4, no. 1, pp. 23-33, 1997.

[2] T. Lei, G. Paolo, and M. Liu, "Virtual-to-real deep reinforcement learning: Continuous control of mobile robots for mapless navigation," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2017.

[3] M. Enrico, and A. Farinelli, "Discrete deep reinforcement learning for mapless navigation," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2020.

[4] J. Jin, N. Nguyen, N. Sakib, et al., "Mapless navigation among dynamics with social-safety-awareness: a reinforcement learning approach from 2D laser scans," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2020.

[5] K. Wu, H. Wang, M. A. Esfahani, et al., "Learn to navigate autonomously through deep reinforcement learning," *IEEE Transactions on Industrial Electronics*, vol. 69, no. 5, pp. 5342-5352, 2022.

[6] T. Bailey, and H. Whyte, "Simultaneous localization and mapping (SLAM): Part II," *IEEE Robotics and Automation Magazine*, vol. 13, no. 3, pp. 108-117, 2006.

[7] R. Fahad, and M. Hayashibe, "Towards robust wheel-legged biped robot system: Combining feedforward and feedback control," in *IEEE/SICE International Symposium on System Integration (SII)*, 2021.

[8] A. Pfrunder, P. Borges, A. Romero, et al., "Real-time autonomous ground vehicle navigation in heterogeneous environments using a 3D LiDAR," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2017.

[9] G. Desouza, and A. Kak, "Vision for mobile robot navigation: A survey," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 24, no. 2, pp. 237-267, 2002.

[9]https://youtu.be/FSm-EamibPI

[10] D. Filliat, and J. Meyer, "Map-based navigation in mobile robots: I. a review of localization strategies," *Cognitive Systems Research*, vol. 4, no. 4, pp. 243-282, 2003.

[11] Y. Wang, and W. Chen, "Hybrid map-based navigation for intelligent wheelchair," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2011.

[12] C. W. Warren, "Fast path planning using modified A* method," in *IEEE International Conference on Robotics and Automation (ICRA)*, 1993.

[13] J. J. Kuffner and S. M. LaValle, "RRT-connect: An efficient approach to single-query path planning," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2000.

[14] M. Missura and M. Bennewitz, "Predictive collision avoidance for the dynamic window approach," in *International Conference on Robotics and Automation (ICRA)*, 2019.

[15] H. Wang, C. Hu, W. Cui and H. Du, "Multi-objective comprehensive control of trajectory tracking for four-in-wheel-motor drive electric vehicle with differential steering," *IEEE Access*, vol. 9, pp. 62137-62154, 2021.

[16] M. B. Radac and T. Lala, "Hierarchical Cognitive Control for Unknown Dynamic Systems Tracking," *Mathematics*, vol. 9, no. 21, article no. 2752, 2021.

[17] G. Oriolo, G. Ulivi, and M. Vendittelli, "Real-time map building and navigation for autonomous robots in unknown environments," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 28, no. 3, pp. 316-333, 1998.

[18] D. H. Lee, S. S. Lee, C. K. Ahn, P. Shi and C. C. Lim, "Finite distribution estimation-based dynamic window approach to reliable obstacle avoidance of mobile robot," *IEEE Transactions on Industrial Electronics*, vol. 68, no. 10, pp. 9998-10006, 2021.

[19] J. Berg, S. Guy, M. Lin, and D. Manocha, "Reciprocal n-body collision avoidance," *Robotics research*, Springer, Berlin, Heidelberg, 2011.

[20] J. Berg, J. Snape, S. J. Guy and D. Manocha, "Reciprocal collision avoidance with acceleration-velocity obstacles," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2011.

[21] K. Guo, D. Wang, T. Fan and J. Pan, "VR-ORCA: Variable responsibility optimal reciprocal collision avoidance," *IEEE Robotics and Automation Letters*, vol. 6, no. 3, pp. 4520-4527, 2021.

[22] G. Kahn, P. Abbeel, and S. Levine, "BADGR: An autonomous self-supervised learning-based navigation system," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 1312-1319, 2021.

[23] L. Zhang, R. Zhang, T. Wu, R. Weng, M. Han and Y. Zhao, "Safe reinforcement learning with stability guarantee for motion planning of autonomous vehicles," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 32, no. 12, pp. 5435-5444, 2021.

[24] Y. Zhu, R. Mottaghi, E. Kolve, et al., "Target-driven visual navigation in indoor scenes using deep reinforcement learning," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2017.

[25] J. Zhang, J. Springenberg, J. Boedecker, et al., "Deep reinforcement learning with successor features for navigation across similar environments," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2017.

[26] C. Sampedro, H. Bavle, A. Ramos, et al., "Laser-based reactive navigation for multirotor aerial robots using deep reinforcement learning," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2018.

[27] M. Pfeiffer, S. Shukla, M. Turchetta, et al., "Reinforced imitation: Sample efficient deep reinforcement learning for mapless navigation by leveraging prior demonstrations," *IEEE Robotics and Automation Letters*, vol. 3, no. 4, pp. 4423-4430, 2018.

[28] G. Kahn, A. Villaflor, B. Ding, et al., "Self-supervised deep reinforcement learning with generalized computation graphs for robot navigation," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2018.

[29] K. Tsunekawa, F. Leiva, and J. Solar, "Visual navigation for biped humanoid robots using deep reinforcement learning," *IEEE Robotics and Automation Letters*, vol. 3, no. 4, pp. 3247-3254, 2018.

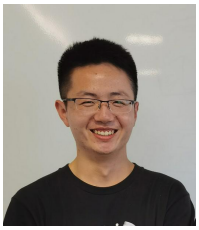
[30] W. Zhu, X. Guo, D. Owaki, K. Kutsuzawa and M. Hayashibe, "A survey of sim-to-real transfer techniques applied to reinforcement learning for bio-inspired robots," *IEEE Transactions on Neural Networks and Learning Systems*, doi: 10.1109/TNNLS.2021.3112718.

[31] V. Mnih, K. Kavukcuoglu, D. Silver, et al., "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529-533, 2015.

[32] Y. Duan, X. Chen, R. Houthooft, et al., "Benchmarking deep reinforcement learning for continuous control," in *International Conference on Machine Learning (ICML)*, pp. 1329-1338, 2016.

[33] T. Haarnoja, H. Tang, P. Abbeel, and S. Levine, "Reinforcement learning with deep energy-based policies," in *International Conference on Machine Learning (PMLR)*, 2017.

[34] F. Raza, W. Zhu, and M. Hayashibe, "Balance stability augmentation for wheel-legged biped robot through arm acceleration control," *IEEE Access*, vol. 9, pp. 54022-54031, 2021.



Wei Zhu received the bachelor's degree in intelligent science and technology from the School of Computer Science and Control Engineering, and the master's degree in Control Science and Engineering from the School of Artificial Intelligence, Nankai University, Tianjin, China, in 2017 and 2020 respectively. He is currently pursuing the Ph.D. degree with the Department of Robotics, Tohoku University, Sendai, Japan.

His research interests include deep reinforcement learning, snake-like robots, wheeled bipedal robots, and autonomous navigation.

Mitsuhiro Hayashibe (Senior Member, IEEE) received the B.S. degree from the Tokyo Institute of Technology, in 1999, and the M.S. and Ph.D. degrees from the Graduate School of Engineering, The University of Tokyo, in 2001 and 2005, respectively. From 2001 to 2006, he was an Assistant Professor with the Department of Medicine, Jikei University School of Medicine.

In 2007, he was a Postdoctoral Fellow with the Institut National de Recherche en Informatique et en Automatique (INRIA) and the Laboratoire Informatique, de Robotique et de Microelectronique de Montpellier (LIRMM), CNRS/University of Montpellier, France. Since 2008, he has been a tenured Research Scientist with INRIA and the University of Montpellier. Since 2012, he has been a Visiting Researcher with the RIKEN Brain Science Institute and TOYOTA Collaboration Center. Since 2017, he has been a Professor with the Department of Robotics, Tohoku University, and founder of the Neuro-Robotics Laboratory. He is currently a Senior Member of the IEEE Engineering in Medicine and Biology Society. He was a recipient of the 15th Annual Delsys Prize 2017 for Innovation in Electromyography from De Luca Foundation, USA. He serves as the Technical Activity Board Member for the International Foundation of Robotics Research (IFRR).