




Robot Subgoal-guided Navigation in Dynamic Crowded Environments with Hierarchical Deep Reinforcement Learning

Tianle Zhang , Zhen Liu* , Zhiqiang Pu , Jianqiang Yi , Yanyan Liang , and Du Zhang

Abstract: Although deep reinforcement learning has recently achieved some successes in robot navigation, there are still unsolved problems. Particularly, a robot guided by a distant ultimate goal is easy to get stuck in danger or encounter collisions in dynamic crowded environments due to the lack of long-term perspectives. In this paper, a novel subgoal-guided approach based on two-level hierarchical deep reinforcement learning with spatial-temporal graph attention networks (ST-GANets), called SG-HDRL, is proposed for a robot navigating in a dynamic crowded environment with autonomous obstacles, e.g., crowd. Specifically, the high-level policy, that models the spatial-temporal relation between the robot and the obstacles using the obstacles' trajectories by the designed high-level ST-GANet, generates intermediate subgoals from a longer-term perspective over higher temporal scales. The subgoals give a favorable and collision-free direction to avoid encountering danger or collisions while approaching the ultimate goal. The low-level policy, that similarly implements the designed low-level ST-GANet to implicitly predict the obstacles' motions, takes the subgoals as short-term guidance through an intrinsic reward incentive to generate primitive actions for the robot. Simulation results demonstrate that SG-HDRL using ST-GANets has better performances compared with state-of-the-art baselines. Furthermore, the proposed SG-HDRL is deployed to an experimental platform based on omnidirectional cars, and experiment results validate the effectiveness and practicability of the proposed SG-HDRL.

Keywords: Collision avoidance, graph attention networks, hierarchical deep reinforcement learning, robot navigation.

1. INTRODUCTION

Recently, with the rapid development of artificial intelligence [1], autonomous mobile robots [2-4] have more and more promising broad applications in real life, such as service robots [5], logistic robots [6], etc. In these applications, robot navigation is a fundamental and critical problem that needs to be solved. The most common case is to make decisions in dynamic crowded environments with autonomous obstacles. In particular, robots need to not only avoid static obstacles, but also avoid dynamic autonomous obstacles, such as pedestrians and other autonomous robots. These obstacles have their own policies and intents, and can make autonomous decisions constantly. Meanwhile, they are not able to form some kind of communication with robots to cooperatively avoid collisions with each other. In addition, robots mostly need to

face the challenge of navigation in crowded environments with a large number of dense obstacles, e.g., crowd, which requires that robot navigation has a long-term perspective. Therefore, finding collision-free, time efficient paths in dynamic crowded environments remains challenging for robot navigation.

Related works of robot navigation can be roughly classified into model-based and learning-based approaches [7-9]. The model-based approaches need to predict the future states of the environment using domain knowledge, including reaction-based methods and trajectory-based methods. The former adopts one-step interaction rules based on geometry or physics to avoid collisions with obstacles [10-12]. These methods are short-sighted as they only consider the next step state, which easily leads robots to generate oscillatory and unnatural behaviors. In contrast, the trajectory methods compute plans

Manuscript received February 23, 2022; revised April 29, 2022; accepted July 28, 2022. Recommended by Associate Editor Yuhu Wu under the direction of Senior Editor Euntae Kim. This work was supported by the National Key Research and Development Program of China under Grant 2018AAA0101005, in part by the Strategic Priority Research Program of Chinese Academy of Sciences under Grant No. XDA27030204, in part by the External Cooperation Key Project of Chinese Academy Sciences (No. 173211KYSB20200002), in part by the Foundation under Grant 2019-JCJQ-ZD-049, and in part by the National Natural Science Foundation of China (62073323).

Tianle Zhang, Zhen Liu, Zhiqiang Pu, and Jianqiang Yi are with the Institute of Automation, Chinese Academy of Sciences, Beijing 100190, China, and the School of Artificial Intelligence, University of Chinese Academy of Sciences, Beijing 100049, China (e-mails: tianle-zhang@outlook.com, {liuzhen, zhiqiang.pu, jianqiang.yi}@ia.ac.cn). Yanyan Liang and Du Zhang are with the Faculty of Innovation Engineering, Macau University of Science and Technology (M.U.S.T), Macau, China (e-mails: {yyliang, duzhang}@must.edu.mo).

* Corresponding author.

on a longer-term perspective to generate smoother paths, but they are computationally expensive and require more knowledge of unobservable states [13,14]. Furthermore, these model-based approaches typically need to depend on obstacles' precise control models, which are difficult to obtain in practice, especially in dynamic crowded environments.

Recent studies show that the learning-based approaches have great potential in robot navigation [15,16]. These approaches mainly adopt deep reinforcement learning methods to learn a value or policy function that implicitly encodes navigating behaviors [17,18]. They focus on building an end-to-end network mapping from the states of environments to actions [19-21]. To better understand the environments, some of them model relations between a robot and obstacles. Everett *et al.* use Long Short-Term Memory (LSTM) [22] to process the observation information of an arbitrary number of obstacles, which packages all obstacles' information [23]. This method ignores that different obstacles have different effects on the robot. To deal with this shortcoming, Chen *et al.* model different relations between a robot and obstacles through attention mechanisms [24]. Although the learning-based approaches have made progress, there are still the following unsolved issues:

- 1) These approaches only use distant ultimate goals to guide a robot. In fact, the initial position of the robot is usually far away from the ultimate goal, and the goal has no meaningful guidance for collision avoidance and easily enables the robot to encounter danger or collisions due to the lack of long-term perspectives. Subgoals can be used as short-term guidance goals, which give a favorable and collision-free direction for approaching the ultimate goal from a long-term perspective [25,26]. This is of great significance for efficient and safe robot navigation.
- 2) These approaches only considers the states of obstacles in current step, which is difficult to predict the motions of the obstacles without analyzing the obstacles' trajectories. This leads to generating short-sighted behaviors for the robot. The trajectories of the obstacles can explicitly reflect their behavior policies and intents. By understanding the trajectories of the obstacles, the robot can predict the obstacles' motions to avoid collisions with them safely.

For the first issue, one efficient way to generate subgoals for robot navigation is to leverage the concept of hierarchical deep reinforcement learning (HDRL), in which multiple layers of policies are trained to perform decision-taking and control at consecutively higher levels of temporal and behavioral abstraction [27]. However, most works of HDRL do not involve robot navigation or collision avoidance problems [28,29]. For the second issue, the trajectories of obstacles contain spatial and temporal information in robot navigation problems. A natural idea is to

model the spatial-temporal relation between the robot and obstacles to analyze the obstacles' motions, including the intents and policies.

Motivated by the above discussions, a new subgoal-guided approach based on two-level HDRL with spatial-temporal graph attention networks (ST-GANets), called SG-HDRL, is proposed for a robot navigating in a dynamic crowded environment with autonomous obstacles. In particular, the high-level policy, that models the spatial-temporal relation between the robot and the obstacles from the obstacles' trajectories by the designed high-level ST-GANet, generates intermediate subgoals from a longer-term perspective by implementing in higher temporal scales. The subgoals give a favorable and collision-free direction to avoid getting stuck in danger or encountering collisions for approaching the ultimate goal. The low-level policy, that similarly implicitly predicts the obstacles' motions by the designed low-level ST-GANet, uses the subgoals as short-term guidance through an intrinsic reward incentive to make motion decisions for the robot. In summary, the main contributions in this paper are listed as follows:

- 1) Differing from guidance with only one ultimate goal, a novel subgoal-guided approach with two-level HDRL, named SG-HDRL, is proposed to generate the subgoals by the high-level policy to guide robot navigation from a longer-term perspective.
- 2) In both the two hierarchical policies, differing from just building the spatial structure relation between the robot and the obstacles with attention mechanism, a spatial-temporal graph attention network, ST-GANet, is designed to model their spatial-temporal relation by using the trajectories of the obstacles for predicting the obstacles' motions.
- 3) The execution and training of the two-level policies in different temporal scales are realized in simulations, and SG-HDRL using ST-GANets has a significant improvement in most evaluation criteria compared with the existing methods. Furthermore, an experimental platform based on omnidirectional cars is build to demonstrate the proposed SG-HDRL, and the experiment results show that SG-HDRL has satisfactory performance.

2. PROBLEM FORMULATION

In this paper, as shown in Fig. 1, the problem of safely and efficiently navigating one robot to its goal position using a subgoal-guided manner in a dynamic crowded environment that is populated by n autonomous obstacles who have decision-making abilities is addressed. The autonomous obstacles are able to make their own decisions according to their intentions. More formally, the geometry of the robot and the obstacles is modeled as a disc

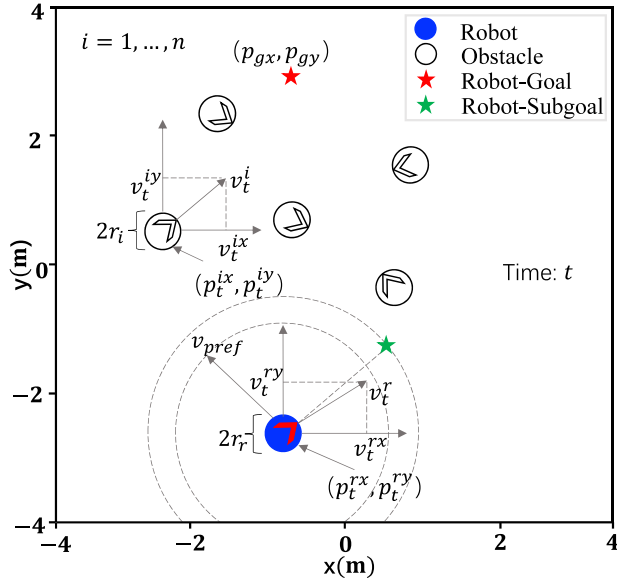


Fig. 1. Robot navigation in dynamic crowded environments with autonomous obstacles.

with an actual shape radius. In each time t , the robot can obtain itself state $s_t^r = [p_g, v_{pref}, p_t^r, v_t^r, r_r]$ and observation state $s_t^o = [s_t^{o1}, \dots, s_t^{on}]$, where $s_t^{oi} = [p_t^i, v_t^i, r_i]$, $i = 1, \dots, n$. Herein, its own state contains the goal position $p_g = [p_{gx}, p_{gy}]$, the preferred speed v_{pref} , the current position $p_t^r = [p_t^{rx}, p_t^{ry}]$, the current velocity $v_t^r = [v_t^{rx}, v_t^{ry}]$ and radius r_r for the robot. The preferred speed v_{pref} is the maximum speed with which the robot would move under the condition that no other obstacles are on its way. The observation state includes the states of the obstacles observed by the robot, which contains the current position $p_t^i = [p_t^{ix}, p_t^{iy}]$, the current velocity $v_t^i = [v_t^{ix}, v_t^{iy}]$ and radius r_i for obstacle i . Meanwhile, the moving trajectories of the obstacles, including the states from time $t - T$ to time t , are considered and define the history state of the robot as $\tau_{(t,T)} = [s_t^s, s_t^o, \dots, s_{t-T}^o]$, where T represents the length of historical time taken into account. Besides, the dynamic of the robot is modeled as a first-order integrator. Hence, the velocity of the robot is taken as the primitive action, i.e., $a_t = v_t$. For the navigation problem, a navigating scheme $\pi : (\tau_{(t,T)}) \mapsto a_t$ can be developed through minimizing the expected time $\mathbb{E}[t_g]$ to the goal under the condition of avoiding collision with the obstacles [17]

$$\underset{\pi(\tau_{(t,T)})}{\operatorname{argmin}} \quad \mathbb{E}[t_g | \tau_{(t,T)}, \pi], \quad (1)$$

$$\text{s.t.} \quad \|p_t^r - p_t^i\|_2 > r_r + r_i \quad \forall t, \quad (2)$$

$$p_g^r = p_g^o, \quad (3)$$

$$p_t^r = p_{t-1}^r + \Delta t \cdot \pi(\tau_{(t-1,T)}), \quad (4)$$

$$p_t^i = p_{t-1}^i + \Delta t \cdot \tilde{\pi}_i, \quad i = 1, \dots, n, \quad (4)$$

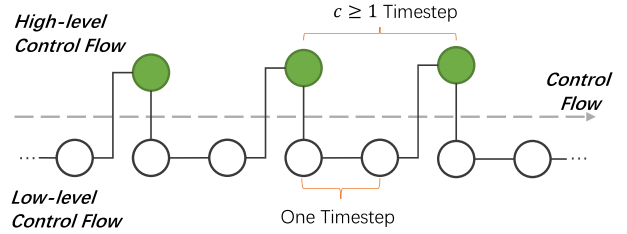


Fig. 2. Hierarchical control flow.

where (2) is the constraint of avoiding collision, (3) is the constraint of reaching the goal, (4) is the kinematics constraint, $\tilde{\pi}_i$ is the moving policy of obstacle i . It can be seen that the expectation in (1) is closely related to the obstacles' policies that are unknown for the robot.

The navigation problem is difficult to be solved directly by optimization methods, while it can be expressed as a sequential decision-making problem, which can be addressed in the deep reinforcement learning (DRL) framework. In this paper, a novel HDRL method is designed to develop a two-level navigating scheme $\pi(\tau_{(t,T)})$ that includes high-level policy $\pi^h : (\tau_{(t,T)}) \mapsto a_t^h$ and low-level policy $\pi^l : (\tau_{(t,T)}, s_t^{sg}) \mapsto a_t$. Herein, $a_t^h = \Delta p_t^r$ is the high-level policy's action, where Δp_t^r represents the relative position with the robot. The high-level policy's action is used to generate a subgoal $p_t^{sg} = \Delta p_t^r + p_t^r$. Meanwhile, s_t^{sg} is the relative position between the robot and the subgoal at time t . Furthermore, in both the high-level policy and the low-level policy, ST-GANet is designed to model the spatial-temporal relation between the robot and the obstacles, which implicitly predicts the obstacles' motions for helping to solve the problem in (1).

HDRL is a promising approach which can extend traditional DRL methods to solve more difficult tasks. In the context of HDRL, multiple layers of policies are trained to perform decisions and controls at different levels of temporal and behavioral abstraction [27]. As shown in Fig. 2, the hierarchical control flow of two-level HDRL is presented. The high-level control flow represents that the high-level policy makes decisions over multiple timesteps, and the low-level control flow represents that the low-level policy are implemented to generate primitive actions every timestep. In the hierarchical policies, of which only the lowest policy applies primitive actions to the environment, higher levels are able to be trained to plan over a longer temporal scale [30]. In addition, the high-level policy learning need to extend traditional Markov decision process to semi-Markov decision process as it may last for multiple timesteps. Recently, some progress has been made in HDRL [31,32]. In order to learn different levels of temporal abstraction, these works consider that a high-level policy is learned to obtain goals over multiple timesteps, and a low-level policy is trained over one timestep to achieve high-level goals. In this paper, two-

level HDRL is used in robot navigation settings and a subgoal-guided approach is proposed to solve robot navigation in dynamic crowded environments.

3. APPROACH

In this section, a new subgoal-guided navigation approach based on two-level HDRL using ST-GANets, called SG-HDRL, is designed for a robot navigating in a dynamic crowded environment. The approach adopts the high-level policy to generate the subgoals, and the low-level policy to produce primitive actions for the robot. In the following, firstly, the overall design of SG-HDRL is given. Then, ST-GANet structure, high-level and low-level policy components are presented in detail. Finally, the training algorithm of SG-HDRL is shown.

3.1. Overall design of SG-HDRL

As shown in Fig. 3, the overall structure of SG-HDRL mainly consists of two components: 1) The high-level policy component with ST-GANet is designed to automatically generate subgoals every c timesteps from a longer-term perspective; 2) The low-level policy component with ST-GANet uses the subgoals as short-term guidance to make motion decisions for the robot navigation every timestep. Moreover, the high-level and low-level policy components both adopt the ST-GANet structure, which is designed to model the spatial-temporal relation between the robot and the obstacles for predicting the obstacles' motions.

3.2. Spatial-temporal graph attention network structure

In reality, a multi-agent system can be naturally described as a graph structure which implies relations in the spatial domain. Meanwhile, graph neural network (GNN)

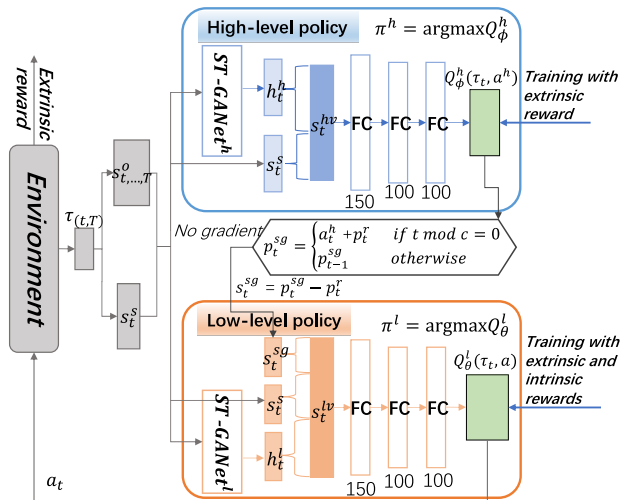


Fig. 3. The overall structure of SG-HDRL.

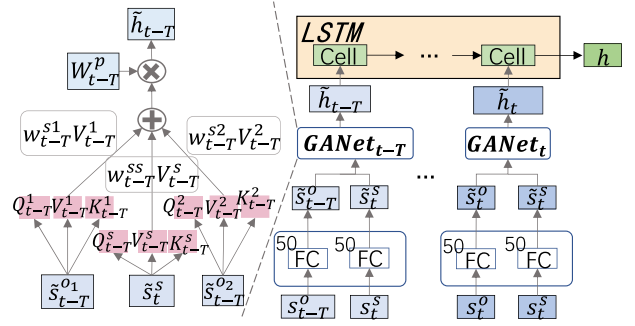


Fig. 4. Spatial-temporal graph attention network structure.

is widely used and deeply studied to process graph-structured data [33]. In the robot navigation task, the robot and autonomous obstacles form a multi-agent system. A natural idea is that their spatial relations can be modelled through GNN. Moreover, GNN with attention mechanism can selectively extract different spatial relations. In this paper, a graph attention network (GANet) based on the attention mechanism Transform [34] is designed to model the spatial relations between the robot and obstacles. Furthermore, modeling the relations between the robot and the obstacles over the past period of time is greatly helpful to predict the motions of the obstacles. Long Short Term Memory (LSTM) [22] is a mature network for processing timing series data. Naturally, LSTM can be used to model and process the temporal relations between the robot and the obstacles from time $t - T$ to current time t .

Therefore, as shown in Fig. 4, a ST-GANet composed of one LSTM and multiple GANets is designed to model the spatial-temporal relation between the robot and obstacles, and extract the influence relation embedding of the obstacles on the robot, which implicitly infers the obstacles' motions. Besides, the high-level ST-GANet and the low-level ST-GANet are two networks with the same structure and different parameters for the high-level and low-level policies, respectively.

In each ST-GANet, firstly, the spatial relations between the robot and the obstacles from time $t - T$ to time t are modeled using multiple GANets with the same structure and different parameters, and the spatial relation embeddings (i.e., $\tilde{h}_{t-T}, \dots, \tilde{h}_t$) are extracted. Then, these spatial relation embeddings are fed into LSTM in the order of time. Finally, the LSTM's final hidden state h is regarded as the influence relation embedding of the obstacles on the robot, which is a fixed-length embedding, reflecting the influence of the robot's surrounding environment (obstacles) on the robot. Moreover, this influence relation embedding (i.e., h) implicitly represents the spatial-temporal relation between the robot and the obstacles, which implicitly predicts the obstacles' motions.

In the following, how to model the spatial relation at each time by one GANet is mainly presented. As shown on

the left side of Fig. 4, the modeling process of the spatial relation between the robot at time t and the obstacles at time $t - T$ is given. This modeling process consists of state encoding and a GANet.

State encoding: Firstly, the observation state $s_{t-T}^o = [s_{t-T}^{o_1}, \dots, s_{t-T}^{o_n}]$ and its own state s_t^s of the robot are encoded as the state embeddings (i.e., $\tilde{s}_{t-T}^o = [\tilde{s}_{t-T}^{o_1}, \dots, \tilde{s}_{t-T}^{o_n}]$, \tilde{s}_t^s) of the same length through two fully-connected (FC) networks, respectively.

GANet: Then, the state embeddings (i.e., \tilde{s}_{t-T}^o , \tilde{s}_t^s) are fed into a GANet. In the GANet, the calculation process mainly includes the following four steps.

a) First of all, each obstacle i computes a query $Q_{t-T}^i = W_{t-T}^q \tilde{s}_{t-T}^{o_i}$, value $V_{t-T}^i = W_{t-T}^v \tilde{s}_{t-T}^{o_i}$ and key $K_{t-T}^i = W_{t-T}^k \tilde{s}_{t-T}^{o_i}$ vectors where $W_{t-T}^q, W_{t-T}^v, W_{t-T}^k$ are learnable parameter matrices.

b) Similarly, the robot also need to compute a query $Q_{t-T}^s = W_{t-T}^q \tilde{s}_t^s$, value $V_{t-T}^s = W_{t-T}^v \tilde{s}_t^s$ and key $K_{t-T}^s = W_{t-T}^k \tilde{s}_t^s$ vectors.

c) After receiving the query-value pair (Q_{t-T}^j, V_{t-T}^j) from the obstacles and the robot ($j \in \{i, s\}$), the robot firstly assigns weight $w_{t-T}^{s_j}$ to each of the state information from the obstacles and the robot

$$w_{t-T}^{s_j} = \text{softmax}\left(\frac{Q_{t-T}^j (K_{t-T}^s)^T}{d_K}\right), \quad (5)$$

where d_K is the dimensionality of the key vector and $(\cdot)^T$ is a transpose operation.

d) The robot then aggregates all the state information by calculating a weighted sum of the values (i.e., V_{t-T}^j) of all the obstacles and itself, and then performing a linear transformation, i.e.,

$$\tilde{h}_{t-T} = W_{t-T}^p \sum w_{t-T}^{s_j} V_{t-T}^j, \quad (6)$$

where W_{t-T}^p is also a learnable parameter matrix, \tilde{h}_{t-T} is a spatial relation embedding which implicitly represents the spatial influence relation between the robot and the obstacles.

3.3. High-level policy component

As shown in the high-level policy component of Fig. 3, the high-level policy $\pi^h(\tau_{(t,T)})$ is a greedy policy about a subgoal action-value function Q_ϕ^h , i.e.,

$$\pi^h(\tau_{(t,T)}) \in \underset{a^h}{\operatorname{argmax}} Q_\phi^h(\tau_{(t,T)}, a^h), \quad (7)$$

where $Q_\phi^h(\tau_{(t,T)}, a^h)$ is parameterized by ϕ using a high-level policy network, mapping from the history state space to high-level action space.

High-level policy network structure: The high-level policy network mainly consists of a high-level ST-GANet and one three-FC-layer network. This network takes the history state as input and outputs high-level action values of the robot. In particular, the history state $\tau_{(t,T)}$ of

the robot is fed into the high-level ST-GANet (i.e., *ST-GANet^h*). The ST-GANet outputs the influence relation embedding of the obstacles on the robot, i.e., h_t^h . Then, the embedding is concatenated with the robot's own state s_t^s to form a mixing vector s_t^{hv} , which contains the knowledge of the robot's state and the surrounding environment. The mixing vector is fed into the three-FC-layer network, and the outputs are high-level action values of the robot.

High-level action: In addition, the high-level action $a_t^h = \Delta p_t^r = [p_t^{hx}, p_t^{hy}]$ is the relative position with the robot, which is not a real subgoal. Hence, the subgoal $p_t^{sg} = p_t^r + a_t^h$ is equal to the high-level action plus the robot's current position, and is updated after executing the high-level policy every c timesteps, otherwise it remains unchanged.

Extrinsic reward function: In this work, a sparse extrinsic reward function $R_E(s_t, a_t)$ is defined to guide the robot navigation. Specifically, the extrinsic reward function is specified to award the robot for reaching its goal (3), and penalize the robot for getting too close or colliding with the autonomous obstacles (2)

$$R_E(s_t, a_t) = \begin{cases} 10, & \text{if } p_{t+1}^r = p_g, \\ -2.5, & \text{else if } d_t^{min} < 0, \\ (d_t^{min} - D) \cdot 5\Delta t, & \text{else if } d_t^{min} < D, \\ 0, & \text{otherwise,} \end{cases} \quad (8)$$

where d_t^{min} is the minimum separation distance between the robot and the obstacles within a duration of $[t - \Delta t, t]$, and D denotes the threshold of an uncomfortable distance between the robot and the obstacles. During the robot navigation, the extrinsic reward is obtained directly from the environment.

High-level optimization objective: The optimization objective of the high-level policy is to maximize the expected cumulative discount high-level return from the extrinsic reward of the environment, resulting in an effective high-level policy, i.e.,

$$\underset{\pi^h}{\operatorname{argmax}} \mathbb{E}_{s_t, P, a_t^h \sim \pi^h, a_t \sim \pi^l} \left[\sum_{i=0}^{(K/c)-1} \gamma_t^i \sum_{t=i*c}^{(\bar{t}+1)*c-1} R_E(s_t, a_t) \right], \quad (9)$$

where γ_t is a larger discount factor that can make the policy consider more long-term return, which implicitly gives the policy a longer-term perspective, P denotes the environment transition probability, K represents the total number of timesteps in an episode, c denotes the timestep interval of the high-level policy execution. Herein, the used extrinsic reward is the received reward of the robot from the environment in the navigation task at every timestep. Note that the high-level policy generates the subgoals to guide the learning of the low-level policy that is used to directly interact with the environment (presented in the low-level policy component). The high-level policy can

indirectly affect the received extrinsic reward of the robot. Moreover, through maximizing the high-level return (9), the high-level policy can generate the optimal subgoals for the robot navigation from a longer-term perspective over large temporal scales.

3.4. Low-level policy component

As shown in the low-level policy component of Fig. 3, the low-level policy is a greedy policy about a primitive action-value function Q_θ^l , i.e.,

$$\pi^l(\tau_{(t,T)}, s_t^{sg}) \in \underset{a}{\operatorname{argmax}} Q_\theta^l(\tau_{(t,T)}, s_t^{sg}, a), \quad (10)$$

where $Q_\theta^l(\tau_{(t,T)}, s_t^{sg}, a)$ is parameterized by θ using a low-level policy network.

Low-level policy network structure: The low-level policy network is composed of a low-level ST-GANet and one three-FC-layer network, which takes the history states and subgoal state as input and outputs the primitive action values of the robot. Specifically, the history state $\tau_{(t,T)}$ is used as the input of the low-level ST-GANet (i.e., $ST - GANet^l$), and its output is the influence relation embedding h_t^l of the obstacles on the robot. Next, the embedding is concatenated with the robot's own state s_t^r and subgoal state s_t^{sg} to form a new joint vector s_t^{lv} . This vector is fed into a three-FC-layer network to generate primitive action values of the robot. Herein, the subgoal state $s_t^{sg} = p_t^{sg} - p_t^r$ is the relative position of the robot from its subgoal.

Primitive action: The robot performs the primitive action to directly act on the environment. In this paper, the velocity of the robot is taken as its primitive action, i.e., $a_t = v_t$. The low-level policy outputs the primitive action interacting with the environment, and is implemented anew every timestep.

Low-level compound reward function: In order to make the robot navigation be guided by the short-term subgoals and the extrinsic reward function simultaneously, a low-level compound reward function R_L is defined for the low-level policy learning. The low-level compound reward is the weighted sum of the extrinsic reward R_E and an intrinsic reward R_I , i.e.,

$$R_L(s_t, a_t^h, a_t) = R_E(s_t, a_t) + \alpha R_I(s_t, a_t^h, a_t), \quad (11)$$

where α is an adjustable hyper parameter. The intrinsic reward R_I is the distance between the current position of the robot and its subgoal, defined by,

$$R_I(s_t, a_t^h, a_t) = \operatorname{clip}(-\|p_{t+1}^r - p_t^{sg}\|_2, -2, 0), \quad (12)$$

where $p_t^{sg} = p_t^r + a_t^h$, and the *clip* operation makes this reward in the range of -2 to 0 , which helps to prevent the gradient explosion caused by the larger absolute value of the reward. In this compound reward, the intrinsic reward is specially designed to encourage the robot to approach

Algorithm 1: Training algorithm for SG-HDRL.

- 1: Initialize high-level Q_ϕ^h , target high-level \tilde{Q}^h , low-level Q_θ^l , target low-level \tilde{Q}^l , high-level replay buffer \mathcal{B}_H , and low-level replay buffer \mathcal{B}_L
 - 2: **for** each episode **do**
 - 3: $\tau_{(t,T)} = \operatorname{env.reset}()$
 - 4: **for** each step $t = 1, \dots, K$ in episode **do**
 - 5: **if** $t \bmod c = 0$ **then**
 - 6: **if** $t > 1$ **then**
 - 7: Compute $R_H^{t-c} := \sum_{t-c}^t R_E^t$
 - 8: Store $(\tau_{(t-c,T)}, R_H^{t-c})$ in replay buffer \mathcal{B}_H
 - 9: **end if**
 - 10: Select new a_t^h by ϵ -greedy($Q_\phi^h(\tau_{(t,T)}, a^h)$)
 - 11: Update subgoal $p_t^{sg} = a_t^h + p_t^r$
 - 12: **end if**
 - 13: Compute subgoal state $s_t^{sg} = p_t^{sg} - p_t^r$
 - 14: Get a_t from ϵ -greedy($Q_\theta^l(\tau_{(t,T)}, s_t^{sg}, a)$)
 - 15: $\tau_{(t+1,T)}, R_E^t = \operatorname{env.step}(a_t)$
 - 16: Compute $R_L^t := R_E^t + \alpha R_I^t$
 - 17: Store $(\tau_{(t,T)}, s_t^{sg}, R_L^t)$ in replay buffer \mathcal{B}_L
 - 18: **end for**
 - 19: Update $\tilde{Q}^h = Q_\phi^h$, $\tilde{Q}_\theta^l = Q_\theta^l$ every d episodes
 - 20: Update Q_ϕ^h using \mathcal{L}^H from \mathcal{B}_H , Q_θ^l using \mathcal{L}^L from \mathcal{B}_L
 - 21: **end for**
-

its subgoal, but the extrinsic reward is retained to achieve the ultimate goal and collision avoidance as well.

Low-level optimization objective: The optimization objective of the low-level policy is to maximize the expected discount cumulative low-level return from the defined low-level compound reward, resulting in an effective subgoal-guided low-level policy, i.e.,

$$\underset{\pi^l}{\operatorname{argmax}} \mathbb{E}_{s_t, p_t, a_t^h \sim \pi^h, a_t \sim \pi^l} \left[\sum_{t=1}^K \gamma_t^l R_L(s_t, a_t^h, a_t) \right], \quad (13)$$

where γ_l is a discount factor smaller than γ_h . By maximizing the intrinsic reward in the low-level return, the low-level policy that takes the subgoals generated by the high-level policy as the short-term guidance for the robot navigation can be generated. Moreover, the low-level policy can generate efficient and collision-free primitive actions for the subgoal-guided robot navigation due to the utilization of the low-level compound reward.

3.5. Training algorithm of SG-HDRL

A novel hierarchical training algorithm based on deep Q-learning algorithm [35] as shown in Algorithm 1 is designed to train SG-HDRL. This algorithm can enable hierarchical policies to be performed and trained over different temporal scales. In particular, firstly, two target action-value functions, i.e., \tilde{Q}^h and \tilde{Q}^l , are defined to be the same

as Q_ϕ^h and Q_θ^l respectively, for the calculation of temporal difference (TD) targets. The parameters of Q_ϕ^h and Q_θ^l are initialized, and $\tilde{Q}^h = Q_\phi^h$, $\tilde{Q}^l = Q_\theta^l$. The high-level replay buffer \mathcal{B}_H and low-level replay buffer \mathcal{B}_L are set. Secondly, the high-level and low-level policies are performed over different temporal scales for the robot navigation. The generated samples by executing the high-level and low-level policies are stored in \mathcal{B}_H and \mathcal{B}_L , respectively. Thirdly, using the samples in \mathcal{B}_H and \mathcal{B}_L , the high-level Q_ϕ^h and the low-level Q_θ^l are updated through two losses, respectively, i.e.,

$$\begin{aligned} \mathcal{L}_\phi^H &:= \mathbb{E}_{\pi^h} [(R_H + \gamma_h \tilde{Q}^h - Q_\phi^h)^2], \\ \mathcal{L}_\theta^L &:= \mathbb{E}_{\pi^l} [(R_L + \gamma_l \tilde{Q}^l - Q_\theta^l)^2]. \end{aligned} \quad (14)$$

Repeat the sample collection and the parameter updates of Q_ϕ^h, Q_θ^l until the policies converge.

4. SIMULATION AND EXPERIMENT RESULTS

4.1. Simulation settings

A simulation environment for a robot navigating in n autonomous obstacles based on CrowdNav proposed by [24] is built to evaluate the performance of SG-HDRL. In this simulation, the autonomous obstacles also need to reach their respective goals without collisions. The autonomous motion of the obstacles is realized by optimal reciprocal collision avoidance (ORCA) [12], where the parameters are sampled from a Gaussian distribution to produce behavioral diversity. The robot implements the proposed SG-HDRL or baseline methods to navigate to its goal. It is assumed that the robot and obstacles are omnidirectional; that is they can move in any direction.

Moreover, a circle crossing scenario as shown in Fig. 5 is used for the simulation. In the circle crossing scenario, the initial positions of $n = 5$ obstacles are randomly positioned on a circle with a center as the coordinate origin $(0, 0)$ and a radius of $4m$, remarked as, obstacle 1, ..., obstacle 5. Random disturbance is added to the X and Y coordinates of the initial positions. The goals of the obstacles are symmetrical to the initial positions about the center of the circle, e.g., g_1, g_2, g_3, g_4, g_5 . The initial position and goal of the robot are also symmetrical about the center of the circle, which are $(0, -4)$ and $(0, 4)$ respectively. Meanwhile, the maximum speed and radius of the robot and obstacles are 1 m/s and 0.3 m , respectively, i.e., $v_{pref} = 1 \text{ m/s}$, $r_r = 0.3 \text{ m}$, $r_i = 0.3 \text{ m}$ ($i = 1, \dots, n$). According to the above settings, without considering collision avoidance, the optimal navigation paths of the obstacles and the robot will pass through the center of the circle at the same time. Thus, the circle crossing scenario can create dynamic crowded circumstances near the center of the circle, which provides a dynamic crowded environment for robot navigation. In addition, to fully validate the effectiveness of SG-HDRL, it is assumed that the robot is

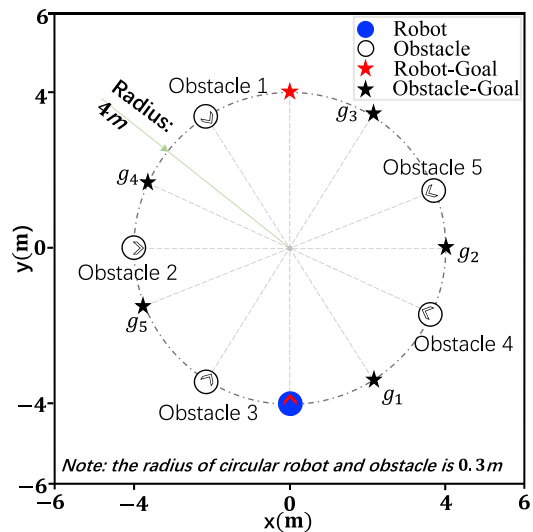


Fig. 5. Circle crossing scenario.

invisible to the obstacles. This requires that the robot need to actively avoid all the obstacles, while the obstacles react only to the obstacles but not to the robot.

To highlight the performance of the proposed method, ORCA [11] and LM-SARL [24] are used as baseline methods. The LM-SARL method mainly adopts attention mechanism to model the interactive relation between the robot and the obstacles. Besides, an ablation method, called SG-HDRL-L, is developed to further verify the effectiveness of the proposed subgoal-guided navigation method. The ablation method only adopts the low-level policy with the low-level ST-GANet, without the subgoals guidance from the high-level policy.

4.2. Implementation details

In the robot navigation, the threshold of uncomfortable distance D is 0.2 m . In each ST-GANet, the size of learnable parameter matrixes W^q, W^v, W^k, W^p are all $(50, 50)$, and the length of historical time is set to 3, i.e., $T = 3$. The high-level policy is implemented every two timesteps, i.e., $c = 2$. Besides, it is considered that the observation radius of the robot is about 2 m (This can be adjusted according to the actual situation), and the goal of the robot is in front of it. Hence, the high-level action space is set to contain 6 discrete actions: $[2 * \cos \theta, 2 * \sin \theta]$, $\theta = 0, \pi/4, \pi/2, 3 * \pi/4, \pi$ and $[0, 0]$. The primitive action space of the low-level policy consists of 81 discrete actions: 5 speeds exponentially spaced between $(0, v_{pref}]$, i.e., $\{\frac{e^{0.2}-1}{e-1}, \frac{e^{0.4}-1}{e-1}, \frac{e^{0.6}-1}{e-1}, \frac{e^{0.8}-1}{e-1}, 1\}$, with 16 headings evenly spaced between $[0, 2\pi)$ and stop action [24]. In each episode, termination has three conditions: reaching the goal, collision and timeout.

For the training phase, the number of training episodes is $100k$. The learning rates of the high-level and low-level

policies are both 0.00001. The high-level discount factor γ_h is 0.98 and the low-level discount factor γ_l is 0.90. They are trained with a batch size of 100 using Adam optimizer. The exploration rates of the policies decrease linearly from 0.5 to 0.1 in the first $20k$ episodes and remain 0.1 for the remaining training episodes. Besides, the number of test episodes is 500.

4.3. Simulation results

Quantitative evaluation: To fully evaluate the effectiveness and efficiency of the proposed method, the proposed method and baseline methods are tested in the same circle crossing scenarios. The test results are shown in Table 1. As expected, the proposed SG-HDRL shows better performance than the baseline methods and the ablation method in the terms of success rate, safety level and cumulative return. This is because the proposed method has subgoals proposed by the high-level policy to guide the robot navigation, which guides the robot cleverly to avoid getting stuck in dangerous or collision situations. On the contrary, the ORCA method fails badly. This is because the invisibility of the robot violates the reciprocal assumption. In addition, compared with the LM-SARL method, the SG-HDRL-L method has a higher success rate and cumulative return. This is because the LM-SARL method only models interaction relation between the robot and the obstacles through attention mechanisms, but does not predict the obstacles’ motions using their trajectories. In con-

trast, SG-HDRL-L using only low-level ST-GANet models the spatial-temporal relation between the robot and the obstacles from the obstacles’ trajectories, which can implicitly predict obstacles’ motions. This allows the robot to better avoid the autonomous obstacles. Meanwhile, SG-HDRL is outperforming SG-HDRL-L in terms of success rate, navigation time and cumulative return. This ablation contrast simulation highlights the superiority of the robot navigation through the subgoal guidance. Although not a large margin, this superiority indicates the benefits of the robot subgoal-guided navigation.

In order to further verify the generalization of the proposed method, the robot’s policy learned by the proposed SG-HDRL is evaluated in new scenarios without any fine-tuning. In particular, firstly, the learned policy of the robot is obtained in a circle crossing training scenario with 5 autonomous obstacles. Then, the learned policy is performed and tested in new circle crossing scenarios where the number of the autonomous obstacles is different from that of the training scenario. Four new scenarios with 2, 4, 6, 8 obstacles are selected, respectively. The test results are shown in Table 2. The results show that the greater the difference in the number of obstacles between the new scenarios and the training scenario, the more obvious the performance degradation is. This is because there are many situations in the new scenarios that have not been encountered in the training scenario. The greater the difference, the more the situations that are not encountered in the

Table 1. Performance of the baseline methods and the proposed method in the circle crossing scenario with $n = 5$ autonomous obstacles. Danger frequency refers to the duration of the robot being too close to obstacles. “Avg. min dis.” refers to the average minimum distance between the robot and the obstacles. Time to goal is computed on successful cases. “Avg. Return” refers to the average cumulative extrinsic return of the robot, i.e., $\mathbb{E}[\sum_{t=1}^K R_E(s_t, a_t)]$.

Methods	% Failures (% collisions/ % timeout) ↓	Time to goal (s) (Avg. / 75th / 90th percentile) ↓	Avg. return ↑ / % Danger frequency ↓ / Avg. min dis. (m) ↓
ORCA [11]	57.0±3.0 (56.6/0.4)	10.93±1.82 / 11.75±1.63 / 13.00±0.94	0.81±0.12 / 30.0±6.0 / 0.08±0.02
LM-SARL [24]	7.0±2.0 (7.0/1.0)	10.98±0.24 / 11.50±0.34 / 12.25±0.36	5.35±0.18 / 14.0±3.0 / 0.16±0.00
RN-ST-GANet	3.0±1.0 (3.0/0.0)	11.26±0.34 / 12.00±0.38 / 12.75±0.38	5.68±0.19 / 3.0±2.0 / 0.14±0.01
SG-HDRL (ours)	1.0±1.0 (1.0/0.0)	10.56±0.25 / 11.00±0.23 / 11.75±0.14	5.98±0.18 / 4.0±1.0 / 0.15±0.01

Table 2. Navigation generalization performance in circle crossing scenarios with different number of autonomous obstacles through the SG-HDRL method (ours).

Number of autonomous obstacles	% Failures (% collisions/ % timeout) ↓	Time to goal (s) (Avg. / 75th / 90th percentile) ↓	Avg. return ↑ / % Danger frequency ↓ / Avg. min dis. (m) ↓
2	4.0±1.0 (1.0/3.0)	9.64±0.23 / 10.00±0.21 / 10.50±0.13	6.07±0.11 / 1.0±1.0 / 0.15±0.02
4	1.0±2.0 (1.0/0.0)	10.19±0.25 / 10.50±0.26 / 11.25±0.19	6.09±0.12 / 4.0±2.0 / 0.15±0.01
6	4.0±1.0 (3.0/1.0)	10.89±0.36 / 11.50±0.27 / 12.25±0.20	5.67±0.22 / 4.0±2.0 / 0.14±0.01
8	9.0±2.0 (6.0/3.0)	11.65±0.23 / 12.50±0.20 / 13.50±0.18	5.07±0.21 / 7.0±3.0 / 0.14±0.01

training. The learned policy of the robot generally has a mediocre performance when it is performed in new situations that are not encountered in the training. Nevertheless, the learned policy still has a low failure rate when the number of the obstacles is 2, 4, 6, 8 in the simulation. Moreover, in these new scenarios, the learned policy can get a high reward and be able to reach the goal fast. Therefore, the results indicate that the proposed method has good generalization and robustness in dynamic crowded environments.

Qualitative evaluation: The illustration of trajectories using different methods is shown in Fig. 6. The blue circle represents the robot, and the orange circles represent the autonomous obstacles. When finding the obstacles, ORCA and LM-SARL demonstrate excessive aggressiveness and conservative behaviors to approach the goal respectively. In contrast, SG-HDRL-L using the low-level ST-GANet and SG-HDRL using ST-GANets can keep a safe distance from the obstacles to approach the goal quickly. This is because the designed ST-GANets can predict the behaviors and intentions of the autonomous obstacles, and avoid them in advance. Besides, the proposed SG-HDRL takes less time to reach the goal compared with SG-HDRL-L. This is due to the guidance of the subgoals. The subgoals can guide the robot to avoid getting stuck in

danger or encountering collisions in dynamic crowded environments from a long-term perspective, which can help the robot to reach the goal quickly while keeping a safe distance from the obstacles.

To fully analyze and verify the effectiveness of the proposed method, two typical scenarios are further investigated, as shown in Fig. 7. In scenario 1 (Fig. 7(a)), the positions of the robot and the obstacles are relatively scattered. The robot controlled by the proposed method is able to bypass the obstacles in advance through the subgoal guidance (green star), instead of moving directly towards the ultimate goal. This navigation scheme enables the robot to avoid getting stuck in dangerous situations, which can promote the efficiency of the navigation. In Fig. 7(c), it can be seen that the highest value action is toward the direction of the subgoal in scenario 1, which means that the proposed method learns the subgoal-guided navigation and it is even explainable. In scenario 2 (Fig. 7(b)), the robot is in a dangerous situation with five obstacles around. Meanwhile, it can be found that the space on the front right side of the robot will become larger in the next step according to the velocities of the obstacles. As expected, the robot also find the space, and it moves to the space by the subgoal guidance (green star). As shown in Fig. 7(d), the most valuable action direction is exactly the result of the previous analysis. Through the qualitative evaluation, it can be concluded that the subgoal-guided policy can take adequate and reasonable actions for safe and efficient navigation.

4.4. Experiment results

Besides the simulations, physical experiments are also implemented to evaluate the performance of the proposed SG-HDRL. As shown in Fig. 8, an experimental platform based on omnidirectional cars is built to deploy the proposed SG-HDRL. The experimental platform consists of omnidirectional cars, a ground control station, a wireless router and a NOKOV motion capture system. Details are given as follows.

- 1) **Omnidirectional cars:** The size of a car is about 234 mm (radius) \times 300 mm (height). Each car is equipped with three Mecanum wheels with 30 mm radius, which make the car move in any direction. As shown in the bottom of Fig. 8, a three-layer control architecture is designed for each car. Specifically, the STM32-based motor drive board forms the bottom control level layer. The open-source flight control board (Pixhawk) and its sensor modules form the middle control level layer. The coprocessor NVIDIA Jetson TX2 and external sensors form the highest local sensing and decision-making level layer. Throughout the architecture, the highest level layer is responsible for the running decision-making algorithms such as robot navigation or collision avoidance algorithms investigated

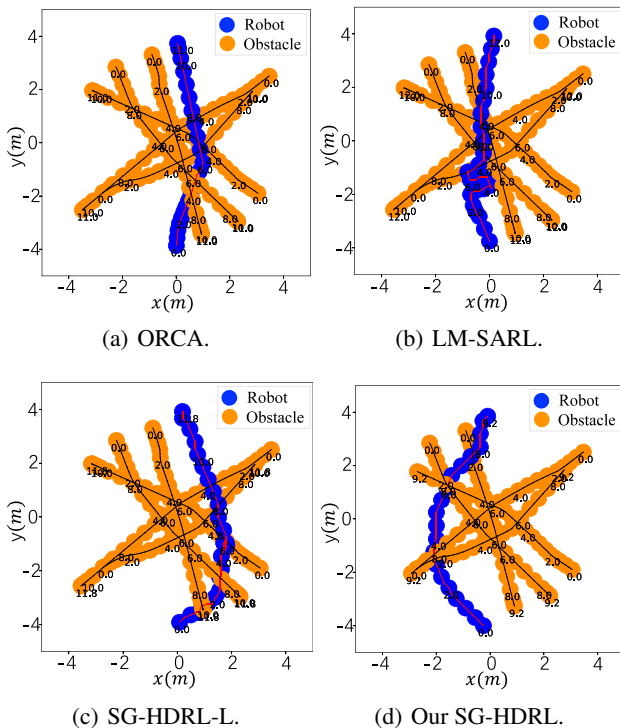


Fig. 6. Trajectories using different methods in the circle crossing scenario with 5 autonomous obstacles. Circles are the position of robot and obstacles at the labeled times.

in this paper. The middle layer is used to receive the desired command, and run on-board sensing and position/attitude control algorithms. The bottom layer is responsible for receiving the desired velocities from the middle layer, and driving the motor according to the calculated PWM.

- 2) **Ground control station:** The ground control station (GCS) is a central point connecting to all the cars. All the cars can be controlled independently or centrally by GCS. Besides, GCS is used to record and display state data and trajectories of the cars. In this article,

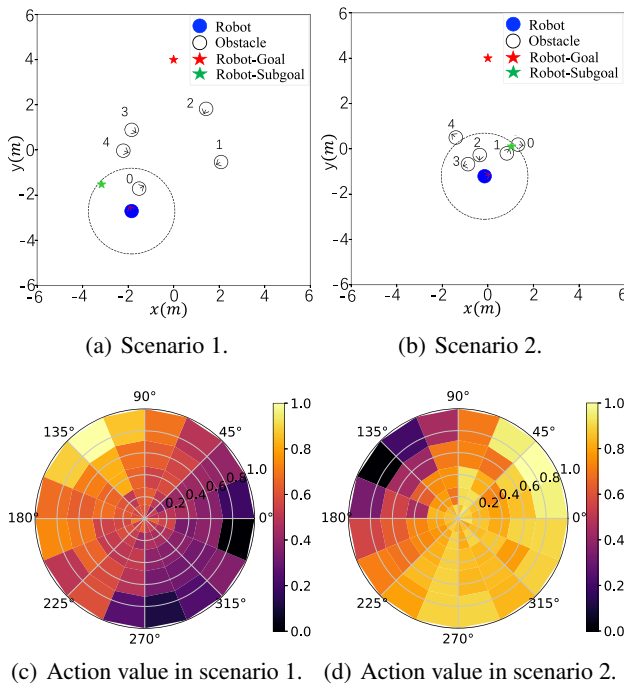


Fig. 7. Value estimates of primitive actions in two scenarios.

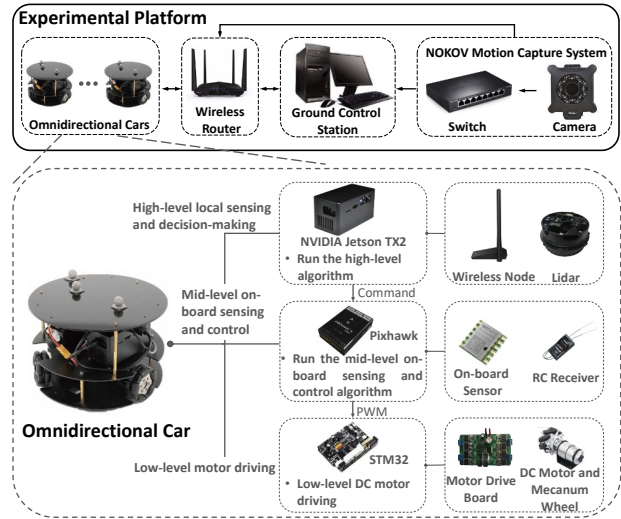


Fig. 8. Experimental platform based on omnidirectional cars.

the control system of the cars is totally decentralized. Each car independently analyzes and processes the environment information, and finally outputs the decision to control itself, without communicating with other cars.

- 3) **NOKOV motion capture system:** The NOKOV motion capture system [36] is used as the indoor positioning system which can provide millimeter-level position measurement. The motion capture system consists of several infrared cameras, one switch, and a set of infrared markers mounted on the top of the cars.
- 4) **Wireless router:** The wireless router is accountable for the transmission of information. For example, the ground control station sends commands to each car through the router, and the NOKOV motion capture system sends the position and velocity of the cars to each car through the router.

Table 3. Main components and features of the experimental platform.

Components	Model number	Features
Omnidirectional car	Three Mecanum wheels	Size: 234 mm (radius) \times 146 mm (height); Max speed: 1.5 m/s; Weight: 1.5 kg
NVIDIA Jetson TX2	/	Computing capability: 1.33 TFLOPs; Memory: 8 GB; GPU: 256-core NVIDIA Pascal
Pixhawk	4 Mini	Main FMU processor: <i>STM32F765</i> ; Barometer: <i>MS5611</i> ; Accel/Gyro: <i>ICM – 20689, BMI055</i> ; Magnetometer: <i>IST8310</i>
NOKOV motion capture	NOKOV MARS 1.3H	Resolution: 1280 \times 1024; Frame rate: 240 Hz; 3D accuracy: ± 0.2 mm; Latency: 4.2 ms
Motor	MG513P30-12V	Rated current: 0.36 A; Rated voltage: 12 V; Rated power: 4 W
Wireless router	GT-AC5300	Max transmission rate: 1732 Mbps; Working frequency band: 5 GHz
Battery	4S-5300mAh-30C	Capacity: 5300 mAh; Discharge rate: 30 C; Voltage: 14.8 V

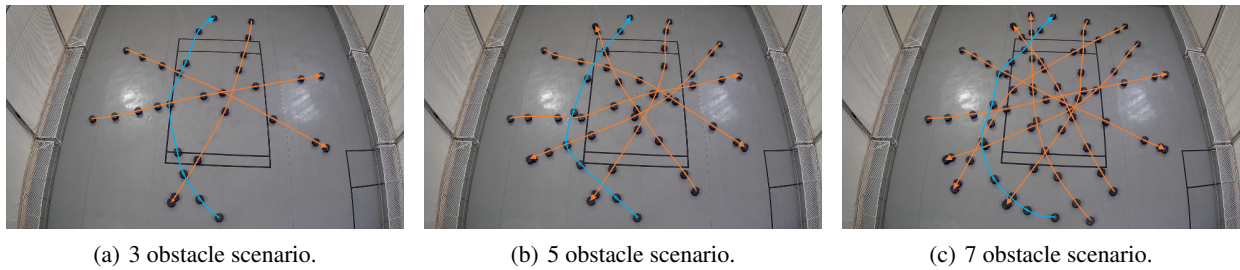


Fig. 9. Navigation trajectories of physical experiments.

In addition, the main components and their features of the experimental platform are given in Table 3.

Based on the abovementioned experimental platform, the performance of the proposed SG-HDRL through physical experiments is evaluated. In the physical experiments, one car is used as the robot and the rest as the autonomous obstacles. The robot adopts the policy learned in the simulation of the circle crossing scenario with 5 autonomous obstacles by using SG-HDRL, and the autonomous obstacles use ORCA algorithm as their policies. Moreover, these policies are implemented on NVIDIA Jetson TX2 of their respective cars in the form of Python codes, respectively. The states of the cars are obtained through NOKOV Motion Capture, and distributed to all the cars. Meanwhile, the history states of all the autonomous obstacles from time $t - 3$ to time t are saved for the robot's policy obtained by the proposed SG-HDRL. Besides, the initial position and goal point of the robot are the same as those of the simulation, which is conducted to check whether the navigation task can be successfully completed.

After that, three physical experiments: 3 obstacle scenario, 5 obstacle scenario and 7 obstacle scenario are selected to evaluate the performance of the proposed SG-HDRL deployed in the real-world settings. The trajectories are collected and shown in Fig. 9. The blue line represents the trajectory of the robot, and the orange lines represents the trajectories of the autonomous obstacles. The arrows indicate the moving directions of the robot or the obstacles. The trajectories show that the robot can safely and efficiently reach the goal while avoiding collision with the obstacles. For these experimental results, it can be concluded that the proposed SG-HDRL is verified in the physical experiments and has the ability of practical application. Moreover, although there may be sim-to-real problems affecting robot navigation performance on the experimental platform, such as positioning accuracy error, control accuracy error and the error between the size of the car and that in the simulation, the navigation task can still be accomplished in the physical experiments. Therefore, these results provide a persuasive evaluation that the proposed method can bridge the gap between simulation and reality.

5. CONCLUSION AND FUTURE WORK

In this paper, SG-HDRL based two-level HDRL using ST-GANets is proposed to address the problem of a robot navigating in a dynamic crowded environment with autonomous obstacles. Specifically, the high-level policy with ST-GANet generates intermediate subgoals from a longer-term perspective. The subgoals provide a favorable and collision-free direction to avoid danger or collisions while approaching the ultimate goal. The low-level policy with ST-GANet takes the subgoals as short-term guidance and intrinsic motivation to generate safe and efficient primitive actions for the robot navigation. In the two-level policies, ST-GANet is designed to model the spatial-temporal relation between the robot and the obstacles for predicating the obstacles' motions. Simulation and physical experiment results show the effectiveness and generalization of SG-HDRL, which outperforms the baseline methods in terms of navigation safety and task accomplishments. In the future, this paper's innovations will be leveraged to investigate how to conduct cooperative navigation in a fully decentralized multi-robot system.

REFERENCES

- [1] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529-533, 2015.
- [2] Z. Sui, Z. Pu, J. Yi, and S. Wu, "Formation control with collision avoidance through deep reinforcement learning using model-guided demonstration," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 32, no. 6, pp. 2358-2372, 2020.
- [3] J.-Y. Jhang, C.-J. Lin, C.-T. Lin, and K.-Y. Young, "Navigation control of mobile robots using an interval type-2 fuzzy controller based on dynamic-group particle swarm optimization," *International Journal of Control, Automation, and Systems*, vol. 16, no. 5, pp. 2446-2457, 2018.
- [4] T. Zhang, Z. Liu, S. Wu, Z. Pu, and J. Yi, "Multi-robot cooperative target encirclement through learning distributed transferable policy," *Proc. of International Joint Conference on Neural Networks (IJCNN)*, IEEE, pp. 1-8, 2020.

- [5] J. Xin, C. Dong, Y. Zhang, Y. Yao, and A. Gong, "Visual servoing of unknown objects for family service robots," *Journal of Intelligent & Robotic Systems*, vol. 104, Article number 10, 2022.
- [6] M. Tampubolon, L. Pamungkas, H.-J. Chiu, Y.-C. Liu, and Y.-C. Hsieh, "Dynamic wireless power transfer for logistic robots," *Energies*, vol. 11, no. 3, p. 527, 2018.
- [7] W. Youn, H. Ko, H. Choi, I. Choi, J.-H. Baek, and H. Myung, "Collision-free autonomous navigation of a small uav using low-cost sensors in GPS-denied environments," *International Journal of Control, Automation, and Systems*, vol. 19, no. 2, pp. 953-968, 2021.
- [8] T. Zhang, T. Qiu, Z. Pu, Z. Liu, and J. Yi, "Robot navigation among external autonomous agents through deep reinforcement learning using graph attention network," *IFAC-PapersOnLine*, vol. 53, no. 2, pp. 9465-9470, 2020.
- [9] R. Tang and H. Yuan, "Cyclic error correction based q-learning for mobile robots navigation," *International Journal of Control, Automation, and Systems*, vol. 15, no. 4, pp. 1790-1798, 2017.
- [10] P. Fiorini and Z. Shiller, "Motion planning in dynamic environments using velocity obstacles," *The International Journal of Robotics Research*, vol. 17, no. 7, pp. 760-772, 1998.
- [11] J. Van den Berg, M. Lin, and D. Manocha, "Reciprocal velocity obstacles for real-time multi-agent navigation," *Proc. of IEEE International Conference on Robotics and Automation*, IEEE, pp. 1928-1935, 2008.
- [12] J. van den Berg, S. J. Guy, M. Lin, and D. Manocha, "Reciprocal n -body collision avoidance," *Robotics Research*, C. Pradalier, R. Siegwart, and G. Hirzinger, Eds., Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 3-19, 2011.
- [13] M. Kuderer, H. Kretzschmar, C. Sprunk, and W. Burgard, "Feature-based prediction of trajectories for socially compliant navigation," *Robotics: Science and Systems*, 2012.
- [14] G. S. Aoude, B. D. Luders, J. M. Joseph, N. Roy, and J. P. How, "Probabilistically safe motion planning to avoid dynamic obstacles with uncertain motion patterns," *Autonomous Robots*, vol. 35, no. 1, pp. 51-76, 2013.
- [15] H.-T. L. Chiang, A. Faust, M. Fiser, and A. Francis, "Learning navigation behaviors end-to-end with actor1," *IEEE Robotics and Automation Letters*, vol. 4, no. 2, pp. 2007-2014, 2019.
- [16] Y. Chen, C. Liu, B. E. Shi, and M. Liu, "Robot navigation in crowds by graph convolutional networks with attention learned from human gaze," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 2754-2761, 2020.
- [17] Y. F. Chen, M. Liu, M. Everett, and J. P. How, "Decentralized non-communicating multiagent collision avoidance with deep reinforcement learning," *Proc. of IEEE International Conference on Robotics and Automation (ICRA)*, pp. 285-292, 2017.
- [18] Y. F. Chen, M. Everett, M. Liu, and J. P. How, "Socially aware motion planning with deep reinforcement learning," *Proc. of IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 1343-1350, 2017.
- [19] P. Long, T. Fanl, X. Liao, W. Liu, H. Zhang, and J. Pan, "Towards optimally decentralized multi-robot collision avoidance via deep reinforcement learning," *Proc. of IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, pp. 6252-6259, 2018.
- [20] T. Fan, X. Cheng, J. Pan, D. Manocha, and R. Yang, "Crowdmov: Autonomous mapless navigation in crowded scenarios," *arXiv preprint arXiv:1807.07870*, 2018.
- [21] T. Fan, P. Long, W. Liu, and J. Pan, "Fully distributed multi-robot collision avoidance via deep reinforcement learning for safe and efficient navigation in complex scenarios," *arXiv preprint arXiv:1808.03841*, 2018.
- [22] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735-1780, 1997.
- [23] M. Everett, Y. F. Chen, and J. P. How, "Motion planning among dynamic, decision-making agents with deep reinforcement learning," *Proc. of IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, pp. 3052-3059, 2018.
- [24] C. Chen, Y. Liu, S. Kreiss, and A. Alahi, "Crowd-robot interaction: Crowd-aware robot navigation with attention-based deep reinforcement learning," *Proc. of International Conference on Robotics and Automation (ICRA)*, pp. 6015-6022, 2019.
- [25] X. Yang, M. Moallem, and R. V. Patel, "A layered goal-oriented fuzzy motion planning strategy for mobile robot navigation," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 35, no. 6, pp. 1214-1224, 2005.
- [26] D. Wang, D. Liu, N. Kwok, and K. Waldron, "A subgoal-guided force field method for robot navigation," *Proc. of IEEE/ASME International Conference on Mechatronic and Embedded Systems and Applications*, IEEE, pp. 488-493, 2008.
- [27] O. Nachum, S. S. Gu, H. Lee, and S. Levine, "Data-efficient hierarchical reinforcement learning," *Proc. of the 32nd International Conference on Neural Information Processing Systems*, pp. 3303-3313, 2018.
- [28] T. D. Kulkarni, K. R. Narasimhan, A. Saeedi, and J. B. Tenenbaum, "Hierarchical deep reinforcement learning: Integrating temporal abstraction and intrinsic motivation," *arXiv preprint arXiv:1604.06057*, 2016.
- [29] T. Haarnoja, K. Hartikainen, P. Abbeel, and S. Levine, "Latent space policies for hierarchical reinforcement learning," *Proc. of International Conference on Machine Learning*, PMLR, pp. 1851-1860, 2018.
- [30] R. Makar, S. Mahadevan, and M. Ghavamzadeh, "Hierarchical multi-agent reinforcement learning," *Proc. of the fifth International Conference on Autonomous Agents*, pp. 246-253, 2001.
- [31] A. S. Vezhnevets, S. Osindero, T. Schaul, N. Heess, M. Jaderberg, D. Silver, and K. Kavukcuoglu, "Feudal networks for hierarchical reinforcement learning," *arXiv preprint arXiv:1703.01161*, 2017.

- [32] P.-L. Bacon, J. Harb, and D. Precup, "The option-critic architecture," *Proc. of the AAAI Conference on Artificial Intelligence*, vol. 31, no. 1, 2017.
- [33] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and S. Y. Philip, "A comprehensive survey on graph neural networks," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 32, no. 1, pp. 4-24, 2021.
- [34] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. u. Kaiser, and I. Polosukhin, "Attention is all you need," *Proc. of 31st Conference on Neural Information Processing Systems (NIPS 2017)*, Long Beach, CA, USA.
- [35] T. Hester, M. Vecerik, O. Pietquin, M. Lanctot, T. Schaul, B. Piot, D. Horgan, J. Quan, A. Sendonaris, I. Osband *et al.*, "Deep q-learning from demonstrations," *Proc. of the AAAI Conference on Artificial Intelligence*, vol. 32, no. 1, pp. 3223-3230, 2018.
- [36] N. Ltd, "Nokov products: Mars series," in *Beijing, China*, 2021. [Online]. Available: <https://www.nokov.com/>



Tianle Zhang received his B.Eng. degree in mechanical engineering and automation from University of Science and Technology Beijing, Beijing, China, in 2018. He is currently pursuing a Ph.D. degree in control theory and control engineering with the Institute of Automation, Chinese Academy of Sciences, Beijing, China. His current research interests include planning and decision-making in mobile robots, multiagent deep reinforcement learning (MDRL), and swarm intelligence.



Zhen Liu received his B.Eng. degree in automation from the Nanjing University of Aeronautics and Astronautics, Nanjing, China, in 2010, and a Ph.D. degree in control theory and control engineering from the Institute of Automation, Chinese Academy of Sciences, Beijing, China, in 2015. He is currently an Associate Professor with the Integrated Information System

Research Center, Institute of Automation, Chinese Academy of Sciences. His research interests include robust adaptive control, fuzzy control, and applications of aerospace systems.



Zhiqiang Pu received his B.Eng. degree in automation from Wuhan University, Wuhan, China, in 2009, and a Ph.D. degree in control theory and control engineering from Institute of Automation, Chinese Academy of Sciences, Beijing, China, in 2014. He is currently a Professor in the Integrated Information System Research Center, Institute of Automation, Chinese

Academy of Sciences. He has been supported by the Talent Program of Youth Innovation Promotion Association CAS since 2017. His research interests include decision intelligence, collective intelligence, and applications of unmanned autonomous systems.



Jianqiang Yi received his B.Eng. degree in mechanical engineering from Beijing Institute of Technology, Beijing, China, in 1985, and his M.Eng. and Ph.D. degrees in automation from the Kyushu Institute of Technology, Kitakyushu, Japan, in 1989 and 1992, respectively. From 1992 to 1994, he worked as a Research Fellow with the Computer Software Development Company, Tokyo, Japan. From 1994 to 2001, He was with MYCOM, Inc., Kyoto, Japan. Since 2001, he has been a Full Professor with the Institute of Automation, Chinese Academy of Sciences, Beijing, China. He has authored or co-authored over 100 international journal papers and 270 international conference papers. His research interests mainly include intelligent control, adaptive control, robot system, and collective intelligence.



Yanyan Liang received his B.S. degree from the Chongqing University of Posts and Telecommunications, Chongqing, China, in 2004, and his M.S. and Ph.D. degrees from the Macau University of Science and Technology (MUST), Taipa, Macau, in 2006 and 2009, respectively. He is currently an Associate Professor with MUST. He has published more than 40 papers related to pattern recognition, image processing, and computer vision in IEEE Transactions and international conferences, including IEEE Transactions on Imageprocessing, IEEE Transactions on Cybernetics, IEEE Transactions on Multimedia, IEEE Transactions on Information Forensics and Security, IJCAI, CVPR, ICPR, and FG. He is also working on smart city applications with computer vision and big data. His research interests include computer vision, image processing, and machine learning.



Du Zhang received his M.S. degree from Nanjing University, Nanjing, China, in 1982, and a Ph.D. degree from the University of Illinois, USA, in 1987. He is currently a Chair Professor with the Faculty of Innovation Engineering, Macau University of Science and Technology, Macau. His research interests cover machine learning (STEP perpetual learning), knowledge-based systems, big data analytics, software engineering, and high-level petri net applications.

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.