Voronoi-Based Multi-Robot Autonomous Exploration in Unknown Environments via Deep Reinforcement Learning

Junyan Hu, Hanlin Niu, Joaquin Carrasco, Barry Lennox, *Senior Member, IEEE*, and Farshad Arvin

***Abstract*—Autonomous exploration is an important application of multi-vehicle systems, where a team of networked robots are coordinated to explore an unknown environment collaboratively. This technique has earned significant research interest due to its usefulness in search and rescue, fault detection and monitoring, localization and mapping, etc. In this paper, a novel cooperative exploration strategy is proposed for multiple mobile robots, which reduces the overall task completion time and energy costs compared to conventional methods. To efficiently navigate the networked robots during the collaborative tasks, a hierarchical control architecture is designed which contains a high-level decision making layer and a low-level target tracking layer. The proposed cooperative exploration approach is developed using dynamic Voronoi partitions, which minimizes duplicated exploration areas by assigning different target locations to individual robots. To deal with sudden obstacles in the unknown environment, an integrated deep reinforcement learning based collision avoidance algorithm is then proposed, which enables the control policy to learn from human demonstration data and thus improve the learning speed and performance. Finally, simulation and experimental results are provided to demonstrate the effectiveness of the proposed scheme.**

***Index Terms*—Autonomous exploration, path planning, deep reinforcement learning, multi-vehicle systems, collision avoidance.**

I. Introduction

MULTI-ROBOT coordination has already established its worth in both theory and practical applications over the past two decades. By combining the techniques of navigation, localization, communication and control, networked multi-robot teams can be coordinated to accomplish a given task in a cooperative manner. Compared with a single sophisticated robot, a collaborative multi-robot team may offer robustness to hardware faults, adaptability to environmental changes and cost-effectiveness. Therefore, multi-robot systems have many potential applications, such as object transportation [1], security surveillance [2], disaster detection [3], formation coordination [4], [5], vehicle-assisted wireless communication networks [6], autonomous shepherding [7], connected vehicle platoons [8], [9], etc. As an important component in multi-robot coordination, cooperative exploration techniques have been widely used in search and rescue missions and mapping of unknown environments, which motivates the need to develop more efficient and feasible solutions.

In recent years, many advanced control methods and path planning techniques have been investigated for use in single robots to perform exploration tasks. In [10], Frontier Exploration planning (FEP) and receding horizon Next-Best-View Planning (NBVP) were combined, using FEP as a global exploration planner and receding horizon NBVP for local exploration. A two-stage heuristic information gain-based NBVP algorithm was proposed in [11] for autonomous exploration and reconstruction in 3-D unknown environments. The two-stage planner included a frontier-based boundary coverage planner and a fixed start open travelling salesman problem solver, such that it returns an optimal path. In [12], a priori knowledge-based dynamic object search strategy was proposed to improve the search efficiency of mobile robot in home environments. An incremental road-map based path planning strategy was developed in [13], where the feasible global path was further optimized with the proposed trajectory optimization method that considered the motion constraints of the robot. In [14], the traditional approach of frontier-based exploration and deep reinforcement learning were combined to allow a robot to autonomously explore unknown cluttered environments. However, in the aforementioned works, autonomous exploration methods were only designed for single-robot cases, which may be viewed as a limitation in exploring large unknown areas. The use of multiple cooperating robots offers several advantages, e.g., reduced mission completion time, increased fault-tolerance of the whole system, and improved localization efficiency [15], which motivates the need for further research that considers multi-robot cases.

When multiple robots are deployed in an exploration task, the autonomous exploration problem becomes more challenging as the cooperation between the robots must be considered thoroughly to ensure high levels of efficiency and to avoid conflicts

Manuscript received June 30, 2020; revised September 16, 2020 and September 26, 2020; accepted October 19, 2020. Date of publication October 29, 2020; date of current version January 22, 2021. This work was supported by the Engineering and Physical Sciences Research Council (EPSRC) under Grants EP/R026084/1, EP/P01366X/1, and EP/S03286X/1, in part by EU H2020-FET-OPEN Robocoenosis project under Grant 899520 and in part the Royal Academy of Engineering under Grant number CiET1819_13. The review of this article was coordinated by Dr. Hui Zhang. *(Corresponding author: Hanlin Niu.)*

The authors are with the Department of Electrical and Electronic Engineering, The University of Manchester, M13 9PL Manchester, U.K. (e-mail: junyan.hu@manchester.ac.uk; hanlin.niu@manchester.ac.uk; joaquin.carrasco@manchester.ac.uk; barry.lennox@manchester.ac.uk; farshad.arvin@manchester.ac.uk).

This article has supplementary downloadable material available at https://ieeexplore.ieee.org, provided by the authors.

Digital Object Identifier 10.1109/TVT.2020.3034800

such as repeated trajectories. Some pioneering research works have been conducted in recent years. As an example, in [16], a significant contribution was made in developing a collaborative multi-robot system that built an information network in an unknown environment by deploying information nodes. By using the proposed networking strategy, the power consumption was reduced while preserving the global coverage. However, this article did not include hardware experiments to validate the theoretical results. In [17], pioneering research work was completed on a decentralized cooperative exploration strategy for a mobile robotic team equipped with range finders using a sensor-based random graph method. A trade-off between information gain and navigation cost was considered in the local planner. In another study [18], a Gaussian mixture model for global mapping to model complex environment geometries was proposed. A small memory footprint was maintained which enables distributed operation with a low volume of communication. A local occupancy grid for use in planning from the Gaussian mixture model was then generated using Monte Carlo ray tracing. In [19], autonomous exploration and mapping strategy was developed for a heterogeneous multi-robot systems including both quadrotors and wheeled mobile robots. Even though exploring a static environment using a team of robots has been investigated extensively [20], [21], dealing with unexpected obstacles, such as human movements and dynamic environments, remains an open question.

With improvements to computational capability and access to large training databases, deep learning has shown great potential for solving autonomous vehicle navigation problems. In contrast to artificial potential field based collision-free path planning methods [22]–[24], which require comprehensive knowledge about the environment (e.g., the sizes and positions of the obstacles) and the robotic platforms (e.g., the accurate mathematical model of the robots), deep reinforcement learning techniques have the potential to achieve safe navigation with less work area information in advance. Besides, since the neural network is trained end-to-end and then map the vehicle sensor data into action command directly, the analysis complexity to obtain the desired trajectory is hence largely simplified for different scenarios.

Some recent progresses in using deep learning techniques to solve autonomous navigation problems are introduced as follows. In [25], an efficient method to solve the inverse reinforcement learning problem based on the sparse Gaussian process prediction was proposed. The robots were able to mimic the expert behaviour for avoiding collision and the work demonstrated the potential that this could be applied to real-robots. In [26], a robust collision avoidance algorithm was developed for autonomous vehicles, which accelerates the learning of optimal policies and efficiently utilizes the remaining resources. In [27], a multi-stage and multi-scenario training framework for training the collision avoidance policy was proposed. The collision avoidance capability of multiple agents was finally demonstrated in a simulated environment. A reinforcement learning based anti-jamming relay scheme for underwater sensor networks was proposed in [28], which optimized the relay mobility and power allocation without being aware of the underwater channel model and the jamming model. The relay performance of the proposed scheme was also validated by real-world experiments in a water tank. In [29], a successor-feature-based deep reinforcement learning algorithm was developed that improved the training speed and transferred learned experience to the new scenarios. A novel deep reinforcement learning algorithm was designed in [30] to integrate long short-term memory and mocal-map critic, enabling the robot to navigate with a limited field of view, whilst outperforming methods that used a wider field of view. In [31], a long-term path planning algorithm using deep reinforcement learning instead of a near field collision avoidance algorithm [32] was proposed to navigate the robot to visit unexplored environments with a greedy strategy. An end-to-end deep reinforcement learning based collision avoidance algorithm was proposed in [33]. The laser data and the relative target position were used as inputs to a neural network and the policy generated heading and speed commands. An asynchronous version of Deep Deterministic Policy Gradient (DDPG) was proposed to improve the Q-value. The efficiency was validated in Gazebo simulation and the real-world environment. However, the performance of conventional deep learning methods relies heavily on the training data and hence a human's experience is not well utilized during this process. How to improve the output performance and accelerate the training process by exploiting human experience and guidance is still an interesting area which needs further efforts.

Motivated by the consistent progress and technological advancements in applications of cooperative exploration, especially by the increasing need to develop novel coordination techniques for handling multiple robots, in this paper, we aim to design an efficient autonomous exploration strategy for a decentralized collaborative multi-robot team while avoiding suddenly observed obstacles. The contributions of the paper can be summarized as follows:

- A Voronoi-based exploration strategy is developed to coordinate a multi-robot team effectively in exploring an unknown area. Each robot is assigned a different target location based on dynamic Voronoi partitions to avoid duplicated exploration areas.
- A collision avoidance algorithm with deep reinforcement learning is proposed to navigate the robot to the target. The proposed method enables the control policy to learn from human demonstration data and outperforms conventional methods in terms of training speed and final performance.
- The feasibility of the proposed cooperative exploration strategy is validated by real-world experiments using wheeled mobile robots.

The rest of the paper proceeds as follow. Section II covers the technical background. In Section III, a hierarchical control architecture for networked explorers is proposed. A Voronoi-based exploration algorithm and deep reinforcement learning based collision avoidance approach are then provided to coordinate the robots efficiently while avoiding sudden obstacles. Simulation case studies and experimental validation using real mobile robots are given in Section IV to highlight the feasibility

and efficiency of the proposed strategy. Section V concludes the paper.

II. Technical Background

A. Communication Networks

Consider a weighted and undirected information graph $I = (\mathcal{V}, \mathcal{E}, \mathcal{A})$ with a non-empty set of nodes $\mathcal{V} = \{1, 2, \ldots, N\}$, a set of edges $\mathcal{E} \subset \mathcal{V} \times \mathcal{V}$, and the associated adjacency matrix $\mathcal{A} = [a_{ij}] \in \mathbb{R}^{N \times N}$. An edge rooted at the i^{th} node and ended at the j^{th} node is denoted by (i, j), which means information can flow from node i to node j. a_{ij} is the weight of edge (j, i) representing the distance between these two nodes and $a_{ij} \neq 0$ if $(j, i) \in \mathcal{E}$. Node j is called a neighbor of node i if $(j, i) \in \mathcal{E}$. An undirected graph I is connected if there is a path between every pair of distinct nodes. The shortest-path distance between two nodes in a graph is the sum of weights for all edges in a shortest path connecting them.

B. Voronoi Partitions

We partition the area between the available mobile robotic platforms using Voronoi partitions in which the centroid of each Voronoi cell is taken to be the position of a single mobile robot. Thus, a certain region within this area (namely the corresponding Voronoi cell) is allocated to each robot for exploring. This is performed on an iterative basis, so the Voronoi partitions are dynamic in nature.

Using Voronoi partitions, the area to be mapped can be broken up dynamically among the robot team members based on their current locations. Also by construction, Voronoi partitioning can be implemented in a decentralized fashion via inter-robot communications.

Let the convex polygon $Q \subset \mathbb{R}^2$ be an open space without obstacles to be explored. Suppose $R_i \ \forall i \in \{1, 2, \ldots, N\}$ are networked robots and $\mathcal{N}_{R_i}$ is a robot set satisfying that each element in this set is a neighbor robot of R_i, and p_i represents the position of robot R_i. The Voronoi cell $\text{Vor}(R_i)$ is defined by:

$$\text{Vor}(R_i) = \{q \in Q \mid \|q - p_i\| \leq \|q - p_j\|, \ \forall R_j \in \mathcal{N}_{R_i}\}, \quad (1)$$

which means the Voronoi partition of Q, generated by R_i, is the set of all points in Q such that all points in the region $\text{Vor}(R_i)$ are closer to R_i than any other point in Q.

Notice that to define the boundaries of each Voronoi cell, robot i at position p_i only needs to know the boundary of Q, and the position of its neighbor p_j, such that the Voronoi cells of p_i and p_j share a common edge. Using dynamic Voronoi partitions, each robot can compute its partition with only knowledge of its neighbors locations. Thus, using Voronoi partitioning facilitates decentralized control designs. Another advantage of using Voronoi partitions is in the case where there is a robot or sensor failure. Because the Voronoi partitions are made dynamically, the team can adjust their Voronoi partition configuration taking into account their new neighbors excluding the failed robot. This procedure ensures that all regions in the area will be covered by an operational robot.

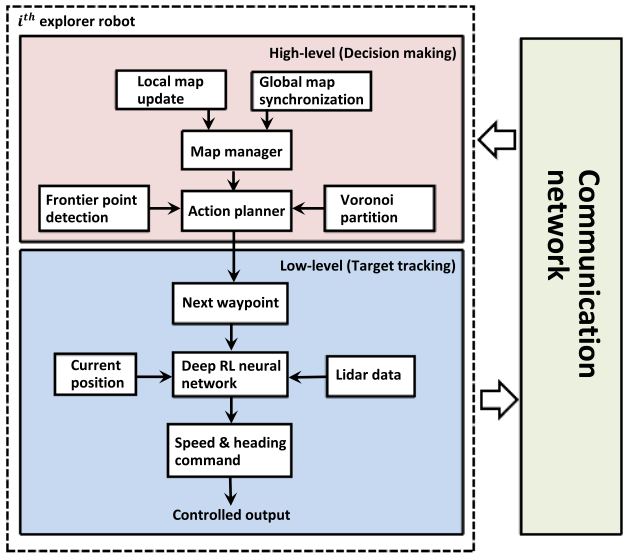


Fig. 1. The hierarchical control architecture includes a high-level decision making layer and a low-level target tracking layer.

C. Collision Avoidance Problem Formulation

The collision avoidance problem for autonomous agents is defined in the context of each autonomous agent moving on the Euclidean plane in the assumption that obstacles might appear. The problem can be formulated as to find the translation function:

$$v_t = f\left(l_t, p_t, v_{t-1}\right) \quad (2)$$

where l_t is the laser range data at time step t, p_t stands for the relative position of the target, v_{t-1} denotes the velocity command at the last time step. The model calculates the next time velocity command v_t. It is assumed that each agent only has partial observations in the sensing range of the rangefinder r_s. This assumption makes our method more practical and robust in real world environments.

III. Cooperative Exploration Strategy

A. Hierarchical Control Architecture

In this subsection, a two-layer control architecture for networked robots is presented, which includes a high-level decision making layer and a low-level target tracking layer as shown in Fig. 1. In the first layer, a desired next frontier point is selected based on the Voronoi partition and the synchronized map. This information is sent to the second layer for tracking and a deep reinforcement learning neural network is used to train the robot to reach the desired position while avoiding potential obstacles.

The cooperative exploration method is proposed under the following assumptions:

1) The robots are equipped with an omnidirectional sensory system that provides a description of the free space surrounding the robot and detects the boundary of an obstacle within the maximum sensing range r_s.

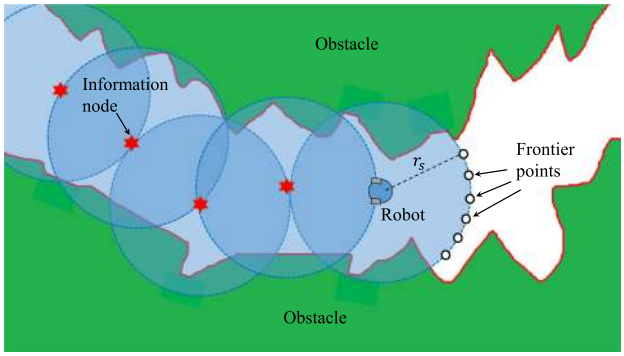


Fig. 2. Exploration strategy using a single robot in a cluttered environment. The information nodes are marked by the red stars and the sensing range is represented by the blue dotted circle. Some new generated frontier points are marked by small black circles which indicate the next possible way point.

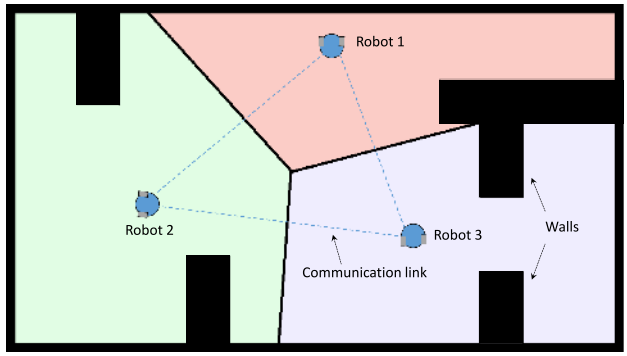


Fig. 3. An example of Voronoi partition using three robots. The working areas of all the robots are represented by different colors. Note that the Voronoi partition is dynamically changing based on different locations of the robots at different time instants, which is a more robust method for dealing with unexpected disturbances in the environments.

2) Each robot can broadcast the information stored in its memory within a communication range r_c at any time. The robot is always open for receiving the relative position from a neighbor robot located within the communication range r_c.
3) The communication range r_c is larger than the sensing range r_s.

B. Coordination Algorithm for Explorer Robots

Frontier-based techniques have been widely used in the multi-robot exploration missions, motivated by [16], [17], [34], the proposed cooperative exploration strategy in the first layer is introduced in this section. In order to mark the explored area, every robot in the multi-vehicle system deploys information nodes while exploring the environment. Those deployed information nodes form an information network, which allows the robots to share information in a decentralized manner. Note that the information nodes can be real sensor devices or virtual targets in the synchronized map of each robot. Once an information node is deployed in a certain position, it will generate some new frontiers points on the border between a sensed area (e.g., blue area in Fig. 2) and an open area covered by no sensor (e.g., white area in Fig. 2). In the case where no frontier point can be found, it can be concluded that the whole environment is fully explored.

At the beginning, the robot R_i deploys a new information node and generates some new frontier points. Now, we will define the utility function for frontier points k assigned to the robot R_i as

$$\Omega_{ik} = \lambda d_{ik} + (1-\lambda)\phi_{ik}, \tag{3}$$

where λ is a scalar that satisfies $0 \leq \lambda \leq 1$, where d_{ik} represents the distance between R_i and frontier point k and ϕ_{ik} denotes the distance between the frontier point k and the initial position of R_i (i.e., the position of the first information node). The robot R_i iteratively searches for the frontier node with the minimum Ω_{ik}. Note that when $\lambda = 0$, the robots perform a breadth-first exploration, that is, they explore all of the neighbor frontier points at the present depth, prior to moving on to the frontier points at the next depth level. On the contrary, if $\lambda = 1$, the robots will adopt depth-first exploration, where the robots explore the frontier points as far as possible before being forced to backtrack and expand other frontier points. Based on different environments and tasks, the value of λ should be fixed by the engineers to achieve the best exploration and mapping performance.

Whenever robot R_i deploys a new information node, the information graph I is updated and each robot broadcasts the updated I so that every robot sharing the network can update I. In order to reach the selected frontier point f_{R_i}, a graph search is performed. R_i locates the information node that generates the selected frontier point first, then R_i selects the short-path distance in the graph to reach the desired information node. We define I_i^t as the subgraph of I at time t generated by the set of nodes which can be observed by R_i. According to the definition of I_i^t, this path is a collision-free path for R_i. Besides, the length of each line segment along this path is bounded by r_s. Once R_i reaches f_{R_i}, it deploys another new information node and iterates this procedure until there is no frontier point in the map. An example is shown in Fig. 2 for a single-robot exploration task using the proposed strategy.

When there exist multiple explorers in the task, it can be observed that, by using the aforementioned strategy, the robots may move to the same direction and explore duplicated areas, thus causing low efficiency and longer mission accomplishment time. In order to avoid such scenarios, a Voronoi-based method is introduced. For any robot i, based on the position of the neighbor robot, R_j, a Voronoi partition is generated and only those frontier points in its own Voronoi partition will be considered for the next movement. This will effectively remove undesirable frontier points to be selected by the explorers. An example of the dynamic Voronoi partition of three robots in the autonomous exploration task is illustrated in Fig. 3.

Towards this end, we consolidate the aforementioned multi-robot cooperative exploration techniques into an algorithm (Algorithm 1) which provides a systematic set of guidelines for the robotic practitioners to implement.

C. Deep Reinforcement Learning Setup

A mapless collision avoidance algorithm is proposed for navigating the mobile robots to go through the waypoints generated

Algorithm 1: Voronoi-Based Cooperative Exploration Strategy.

repeat
  for each robot $i \in \{1, \ldots, N\}$ **do**
    if robot R_i detects no information node inside the circle with radius r_s **then**
      R_i drops an information node at the current position;
    end if
    if f_{R_i} is empty **then**
      R_i searches for the next frontier point which is inside its own Voronoi partition with minimum Ω_{ik};
      The selected frontier point is set as f_{R_i};
      R_i starts moving along the shortest path in I_i^t to reach f_{R_i};
    end if
    if R_i meets f_{R_i} **then**
      R_i drops an information node at the position;
    else
      R_i keeps moving to reach f_{R_i};
    end if
  end for
until there is no frontier point for every information node;

by the cooperative exploration algorithm. The mapless navigation algorithm allows the mobile robots to navigate through the waypoints without having accurate information of static obstacles in advance, which improves the safety of the mobile robots significantly. As it is expected that some obstacles will appear while the mobile robot is navigating through the waypoints, a collision avoidance algorithm is required to generate reactive maneuvering.

1) Observation and Action Space: The laser rangefinder state l_t, previous velocity state v_{t-1} and relative target position state p_t at the time t concentrate the observation state vector s_t. The relative target state p_t is represented in the polar coordinate, i.e., the relative distance and angle between the robot and the target. The continuous action generated by the collision avoidance policy includes linear velocity v_l and angular velocity v_a.

2) Reward Space: Each robot is commanded to navigate through the waypoints generated by the path planning algorithm, while avoiding the potential collisions. The reward function is defined as

$$r = r_d + r_{cl} + r_{av} + r_{lv} \tag{4}$$

where r represents the sum reward, r_d represents the distance reward, r_{cl} describes the safety clearance reward, r_{av} denotes the angular velocity reward, and r_{lv} is the linear velocity reward. r_d can be calculated by

$$r_d = \begin{cases} r_a & \text{if } d_p < d_{p\min} \\ \Delta_{d_p} & \text{otherwise} \end{cases} \tag{5}$$

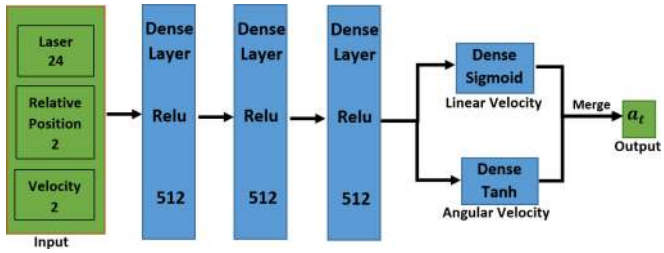


Fig. 4. Actor network: the input layer is a concatenation vector of rangefinder data (24-dimensional vector) and relative target position (2-dimensional vector) and velocity (2-dimensional vector), followed by three dense layers and each Relu layer has 512 nodes.

where r_a is the arrival reward when the distance between robot and target d_p is less than threshold $d_{p\min}$. Otherwise, r_d is the distance Δ_{d_p} the robot moves towards the target at the last time step. Safety clearance reward r_{cl} can be calculated by

$$r_{cl} = \begin{cases} r_{cp} & \text{if } d_{o\max} \le d_p < 2d_{o\max} \\ r_{cpo} & \text{if } d_p < d_{o\max} \\ 0 & \text{otherwise} \end{cases} \tag{6}$$

where r_{cp} denotes the negative reward/punishment when distance between target and robot d_p is between clearance distance threshold $d_{o\max}$ and $2d_{o\max}$. When d_p is less than threshold $d_{o\max}$, negative reward r_{cpo} will be applied. Otherwise, the robot will not be punished. The angular velocity reward r_{av} and linear velocity r_{lv} reward are given by

$$r_{av} = \begin{cases} r_{ap} & \text{if } |v_a| > v_{a\max} \\ 0 & \text{otherwise} \end{cases} \tag{7}$$

$$r_{lv} = \begin{cases} r_{lp} & \text{if } v_l < v_{l\min} \\ 0 & \text{otherwise} \end{cases} \tag{8}$$

where $v_{a\max}$ denotes the angular velocity threshold and $v_{l\min}$ represents the linear velocity threshold. If angular velocity v_a is bigger than $v_{a\max}$ or linear velocity v_l is smaller than $v_{l\min}$, the robot will receive punishment value r_{ap} or r_{lp}, respectively.

3) Network Structure: As shown in Fig. 4, the input to the neural network is the concatenation vector of rangefinder data (24-dimensional vector), relative target position (2-dimensional vector) and velocity data (2-dimensional vector). The input layer is connected with three dense layers with each layer having 512 nodes. The actor network finally generates the linear velocity command through a sigmoid function and produces the angular velocity using a hyperbolic tangent function. These two velocity commands are finally concatenated into the action state, which is used as the input of the critic network shown in Fig. 5. The state data of the robot (28-dimensional vector) is also used as the input of the critic network and is processed by three dense layers and each layer has 512 nodes. The action input is merged with the second layer, and the Q-value is finally generated by a linear activation function.

4) Integration of DDPG and PER: Deep Deterministic Policy Gradient (DDPG) algorithm is an actor-critic and model-free algorithm that can operate over continuous action spaces. The

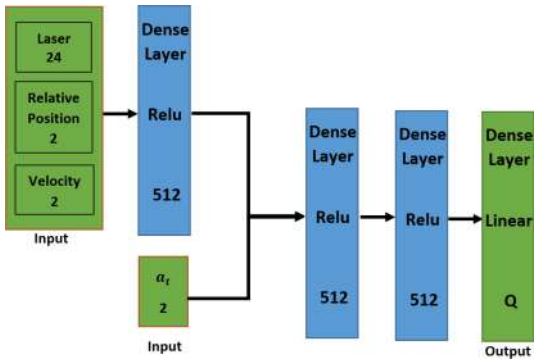


Fig. 5. Critic network: the input layer includes range finder data (24-dimensional vector), relative target position (2-dimensional vector), velocity (2-dimensional vector) and action data (2-dimensional vector). Each dense layer has 512 nodes.

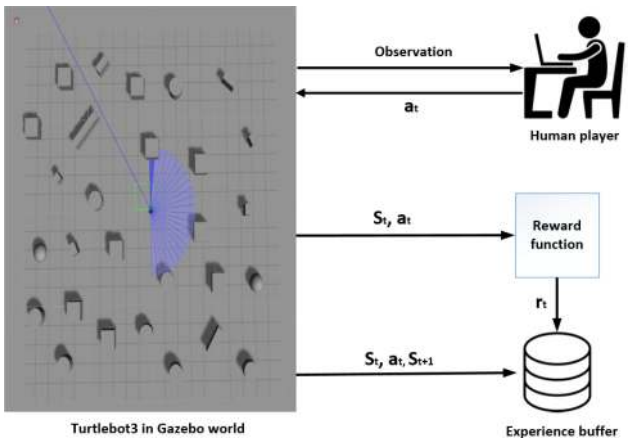


Fig. 6. Human demonstration data collection.

performance of DDPG is demonstrated in 20 simulated physics tasks [35]. However, randomly action exploration makes the learning process inefficient in terms of mobile robot collision avoidance application. In order to improve the performance and learning speed of DDPG, we integrate DDPG with Prioritised Experience Replay (PER) algorithm using human demonstration data. In this research, human demonstration data R_{demo} is recorded before training the networks, as shown in Fig. 6. Through observing the robot behaviour and Gazebo environment, we manually control the robot to avoid obstacles and move towards the goal. The reward value r_t is calculated by using reward function to evaluate the state data s_t and action data a_t. Instead of sampling data uniformly, we integrate DDPG and PER algorithm to sample important data more frequently by calculating the priority of each state transition. The integration of DDPG and PEG is shown in Algorithm 2.

In this research, the implementation of PER is given as follows: In Line 6 and 7 (in Algorithm 2), the random process for angular velocity v_a and linear velocity v_l actions are set as choosing random values from ranges $(-\varphi, \varphi)$ and $(0, \psi)$ respectively, where φ and ψ are positive constants depending on the hardware constraints of the robotic platforms. In Line 13, p_i describes the priority of a transition, δ means the Temporal Difference (TD) error, the second term $\lambda|\nabla_a Q(s_i, a_i|\theta^Q)|^2$ denotes the actor loss, τ_p represents a small positive sampling probability for each transition, ensuring each transition data can still be sampled and τ_D is used for increasing the sampling frequency of demonstration data. Note that τ_p and τ_D are user-defined constant values based on empirical experience. In Line 14, $P(i)$ defines the sampling probability of the i^{th} transition. In Line 18, the sampling weight ω_i of each transition for updating the network is calculated, indicating the importance of each transition data. B stands for the total batch size, and β is a constant value for adjusting the sampling weight. Correspondingly, the critic network and actor network are updated using the equations shown in Line 19 and Line 20.

Algorithm 2: Integration of DDPG and PER.

1: Randomly initialize critic neural network $Q(s, a|\theta^Q)$ and actor neural network $\mu(s|\theta^\mu)$ with weights θ^Q and θ^μ.
2: Initialize target network Q' and μ' with weights $\theta^{Q'} \leftarrow \theta^Q, \theta^{\mu'} \leftarrow \theta^\mu$
3: Initialize replay buffer R_p with priority using demonstration data R_{demo} and default priority value P_{demo}: $R_p \leftarrow [R_{demo}, P_{demo}]$
4: **for** episode = 1, M **do**
5: Initialize a random process $\mathcal{N}$ for action exploration:
6: $v_a = getRandom(-\varphi, \varphi)$
7: $v_l = getRandom(0, \psi)$
8: Receive initial observation state s_1
9: **for** t = 1, T **do**
10: Select action $a_t = \mu(s_t) + \mathcal{N}_t$ according to the current policy and exploration noise
11: Execute action a_t, calculate reward r_t and observe new state s_{t+1}
12: Calculate sampling priority of each transition:
13: $p_i = \delta_i^2 + \lambda|\nabla_a Q(s_i, a_i|\theta^Q)|^2 + \tau_p + \tau_D$
14: $P(i) = \frac{p_i^\alpha}{\sum_k p_k^\alpha}$
15: Store transition $(s_t, a_t, r_t, s_{t+1}, P(i))$ in R_p
16: Sample a minibatch of N transitions (s_i, a_i, r_i, s_{i+1}) from R_p according to the sampling priority $P(i)$
17: Set $y_i = r_i + \gamma Q'(s_{i+1}, \mu'(s_{i+1}|\theta^{\mu'})|\theta^{Q'})$
18: Set weighted updates to network: $\omega_i = (\frac{1}{B} \cdot \frac{1}{P(i)})^\beta$
19: Update critic by minimizing the loss: $L = \frac{1}{N}\omega\sum_i (y_i - Q(s_i, a_i|\theta^Q))^2$
20: Update the actor policy using the sampled policy gradient: $\nabla_{\theta^\mu} J \approx \frac{1}{N}\omega \sum_i \nabla_a Q(s, a|\theta^Q)|_{s=s_i, a=\mu(s_i)} \nabla_{\theta^\mu}\mu(s|\theta^\mu)|s_i$
21: Update the target networks:
22: $\theta^{Q'} \leftarrow \tau\theta^Q + (1-\tau)\theta^{Q'}$
23: $\theta^{\mu'} \leftarrow \tau\theta^\mu + (1-\tau)\theta^{\mu'}$
24: **end for**
25: **end for**

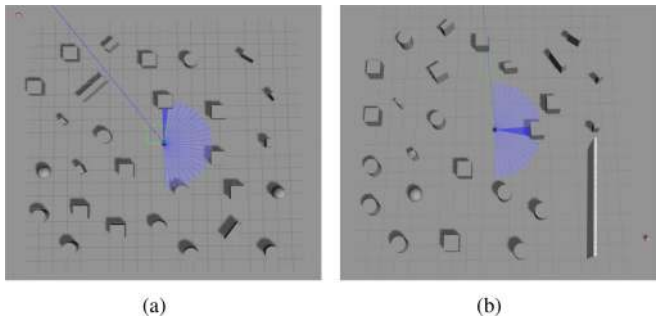
In terms of the computational complexity of the proposed collision avoidance algorithm, the policy is trained end-to-end and it maps discrete lidar information and relative goal positions into action command directly instead of using the whole environmental map. As shown in Fig. 4, the computational complexity of actor neural networks can be denoted as $O(S_{ain}F_{a1}F_{a2}F_{a3}S_{aout})$, where S_{ain}, S_{aout} denote the dimension of the input layer and output layer for actor neural network, respectively. F_{a1}, F_{a2} and F_{a3} represent the dimension of three fully connected layers for actor network, respectively. As shown in Fig. 5, the computational complexity of critic neural networks can be represented by $O((S_{cin}F_{c1} + S_{aout})F_{c2}F_{c3}S_{cout})$, where S_{cin} and S_{cout} represent the dimension of the input layer and output layer for critic network, respectively. F_{c1}, F_{c2} and F_{c3} denote the dimension of three fully connected layers for critic network, respectively. Considering that the collision avoidance algorithm can be deployed on each mobile robot independently, the total computational complexity can be described by $O(NM)$, where N is the number of mobile robots and M represents the function of the dimension of input states, output states and each fully connected layers: $M = S_{ain}F_{a1}F_{a2}F_{a3}S_{aout} + (S_{cin}F_{c1} + S_{aout})F_{c2}F_{c3}S_{cout}$. Since the dimension of input layer, output layer and each one-dimensional fully connected layer are pre-defined and fixed, M can be considered as a constant. Therefore, for the multi-robot scheme, the proposed collision avoidance algorithm will ensure the computational time increase linearly with the number of mobile robots.

IV. Results and Analysis

A. Simulation Results

In this subsection, the performance of the proposed Voronoi-based cooperative exploration strategy is tested with different numbers of robots. The goal is to use mobile robots to explore and map an unknown environment containing several obstacles. The simulation results demonstrate the effectiveness and feasibility of the proposed scheme and also show its efficiency when coordinating multiple robots. Firstly, we used Gazebo [36] as our simulation software, which is an open source robotic platform, to train the collision avoidance policy. The comparison of the proposed algorithm and DDPG is evaluated in terms of training speed and final performance. Then, the simulation results of three different scenarios (with two, three and four robots, respectively) are provided to verify the feasibility of the proposed Voronoi-based cooperative exploration techniques. Finally, a comparison of the performance with a conventional exploration approach based on multiple trials is provided to show the efficiency of the proposed algorithm.

1) Collision Avoidance Policy Training and Evaluation: The Turtlebot3 Waffle Pi model is applied in both Gazebo simulation and the real-world experiment. In the Gazebo environment, as shown in Fig. 7(a), cubes, cylinders, spheres and cuboids are used as obstacles and the red sphere represents the target location of the robot. The robot is commanded to go to the target position while avoiding the obstacles. Once the robot arrives at the target or collides with a simulated obstacle, the robot will be reset to be at the origin position and the red sphere is placed randomly in the outer area of the obstacles. 3000 steps of demonstration data were sampled for training. In terms of the reward function, the threshold values $d_{p\min}$ (0.2 m), $d_{o\max}$ (0.25 m), $v_{a\max}$ (0.8 rad/s), $v_{l\min}$ (0.052 m/s) are set according to the geometry and capacity of Turtlebot3 and can also be reconfigured for other robotic platforms. We set $\tau_p = 0.1$ and $\tau_D = 0.4$ in Algorithm 2 based on empirical experience. The Q-value and reward value comparison between DDPG and the proposed algorithm for the training process are shown in Fig. 8 and Fig. 9 respectively.



(a) (b)

Fig. 7. Gazebo environment for (a) training and (b) evaluation: The red sphere represents the target location; the blue lines stand for the laser beam around the Turtlebot3; the gray cubes, cuboids, cylinders and spheres are used as obstacles.

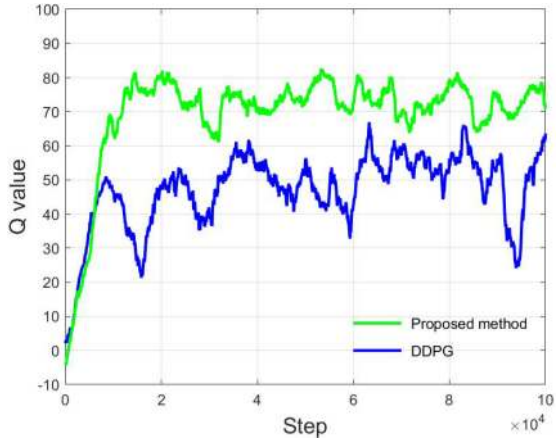


Fig. 8. Q-value comparison between DDPG and the proposed method.

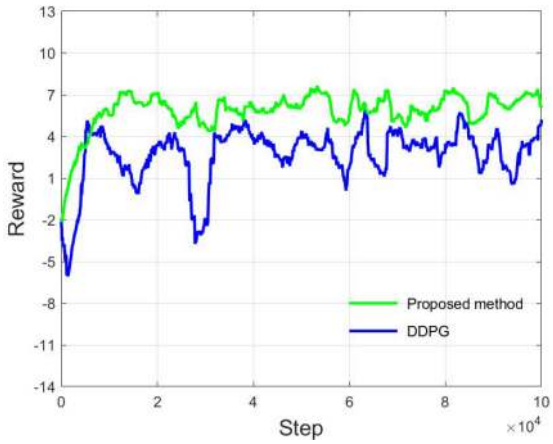


Fig. 9. Reward comparison between DDPG and the proposed method.

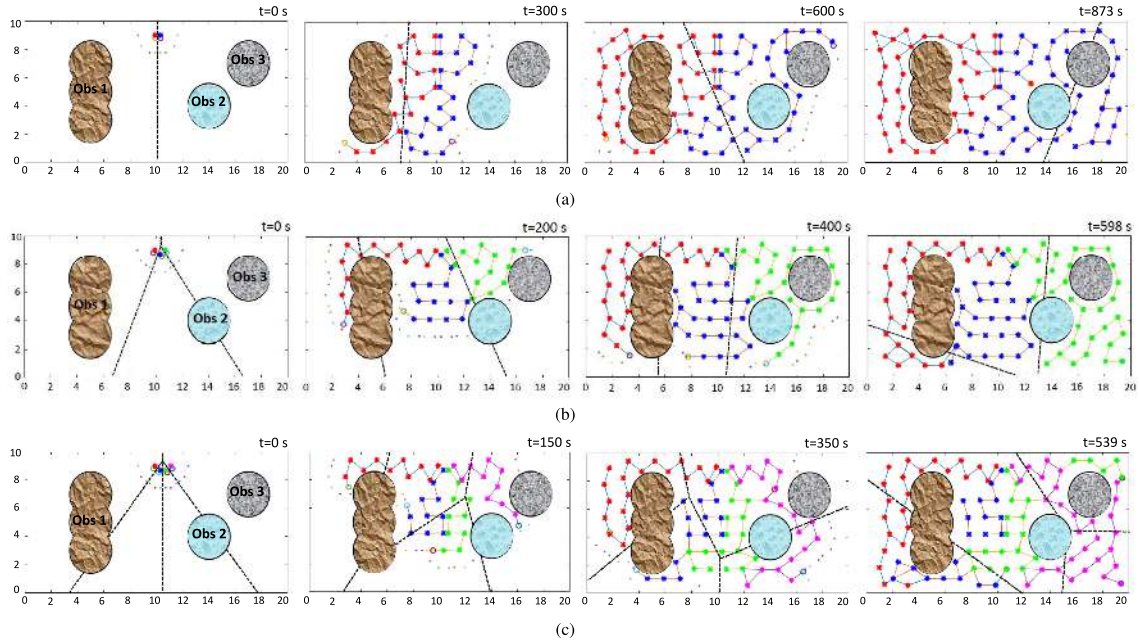


Fig. 10. Simulation results of the proposed cooperative exploration approach at different time instants are shown for different cases: (a) using two robots for exploration; (b) using three robots for exploration; (c) using four robots for exploration. There are three obstacles in the environment represented by brown, blue and gray areas respectively. The deployed information nodes from different robots are marked by solid circles with different colors. The existing frontier points in the environment are marked by small points. The boundary of the Voronoi partition at the current time instant is shown by the dotted line.

In Fig. 8, the Q-value of the proposed method arrives at 67 using 8900 steps while the original DDPG needs 63310 steps, showing that the proposed algorithms only needs 14% steps of DDPG. As shown in Fig. 9, the reward value of the proposed method increases faster than DDPG in the beginning and the average reward of the proposed method between 20000 steps and 100000 steps is 6.0782 and that of DDPG is just 2.9883. The fluctuation range of the proposed method is also smaller than DDPG, meaning that the proposed method is more robust.

To evaluate the performance of the trained models in an unknown environment, a new Gazebo environment was applied, as shown in Fig. 7(b). In the new environment, the locations of obstacles are rearranged and a long wall is added, making the environment more challenging than the training environment. We used 20 random missions to evaluate the performance of DDPG and the proposed algorithm. We compared the trained model after only 10000 training steps of the proposed method and the trained model after 50000 training steps of DDPG. It was found that the proposed method enables the robots to complete all 20 missions successfully and even find the targets behind the wall three times. In comparison with the proposed method, the DDPG method only completed 15 of the 20 random missions. In these 15 successful missions, the mobile robot still had unnecessary turns during 3 missions.

In terms of training time, collecting and training 10000 steps data of the proposed algorithm only cost 40 minutes. However, it cost 2 hours and 40 minutes to collect and train 50000 steps data using DDPG. In terms of trajectory quality, the DDPG method makes the robot travel longer distances than the proposed method and also generates unnecessary turns. Therefore, the proposed method enables mobile robot to learn faster, only requiring 14% steps and 25% of the training time required by DDPG, and still achieves better performance than DDPG in terms of safety and trajectory quality.

2) Performance of the Proposed Cooperative Exploration Strategy: As shown in Fig. 10, the size of the arena is set to 20 m $\times$ 10 m. There are three obstacles in this environment, which are indicated by the brown, blue, and gray areas. To validate the effectiveness of the proposed strategy, three different scenarios (using 2, 3 and 4 robots) are considered for comparison. The sensing range of each robot is set as $r_s = 1.3$ m and the communication range is set as $r_c = 5$ m. The tuning parameter λ for each robot is set as 0.8 in all the three cases. Note that different selections of λ may only affect total exploration time due to different structures of the environment, but the exploration mission will always be completed within finite time regardless of the value of λ.

In the first case, two explorers are used for this autonomous exploration task. Fig. 10(a) depicts the trajectories of the robots and positions of the obstacles. At the beginning of the task (i.e., $t = 0$ s), both robots are placed at the top of the arena. Following the proposed algorithm, they start deploying the first information node at their initial positions and thus generate several frontier points at the edge of the sensing area. The dotted line represents the generated Voronoi partition based on their current locations. As it can be seen from the second and third sub-figures in Fig. 10(a), due to the effect of Voronoi partition, one robot explores mainly the left part of the arena while the other moves toward the right side. The information nodes deployed by the two robots are marked by red and blue stars, respectively. Finally, the cooperative exploration task is accomplished within 873 s and no frontier point can be observed in the environment.

In the second case, three robots are implemented to perform the mission. Similar to the first case, they are placed at the top of the arena in the beginning as shown in the first sub-figure in Fig. 10(b). Due to the communication between the robots, the arena is divided into three partitions and each robot will only explore its corresponding area. Since the robots are moving, the Voronoi partition of the whole environment is also time-varying, which guarantees all the collaborative robots are working efficiently. The information nodes deployed by each robot are represented by red, blue, and green stars respectively. From the last sub-figure of Fig. 10(b), the mission accomplishment time is 598 s, which is significantly less than when using two robots and shows the high efficiency in increasing the number of working robots.

In the third case, we increase the number of robots to four for the same autonomous exploration and mapping mission. The simulation results are illustrated in Fig. 10(c) and the information nodes deployed by each robot are represented by red, blue, green, and pink stars, respectively. The mission is completed within 539 s, which is slightly shorter than when using three robots.

3) Comparing the Proposed Cooperative Exploration Method With the Approach in [16]: In this subsection, we compare the performance of the proposed exploration strategy with the conventional approach presented in [16]. Note that the exploration algorithm in [16] can only avoid the same frontier point that is selected by different robots as their next goal, which cannot guarantee the efficiency when coordinating multiple robots in a large unknown environment. Alternatively, the proposed Voronoi-based strategy ensures that each robot explores a certain area without conflicting with others, thus maximally decreasing the task completion time and saves more resources and energy. Besides, the sudden obstacles can also be handled in the proposed scheme via deep reinforcement learning techniques, which provides more robustness to the multi-robot systems when performing the real-world mission.

In order to verify the performance of both strategies, 50 trials were tested for each case shown in the previous subsection corresponding to 50 different sets of initial positions of the explorers. Fig. 11 illustrates the task completion time of the robots subjected to different scenarios. It can be seen that the proposed control scheme leads to a faster exploration speed in all cases, which can be viewed as a significant improvement over conventional methods. Note that due to the limited size of the arena and the density of the robots, the reduction in mission completion time becomes insignificant when the number of robots increases above five. Hence, a trade-off between the exploration time and the available hardware resources should be determined based on the specific environments and requirements of the tasks to achieve better performance.

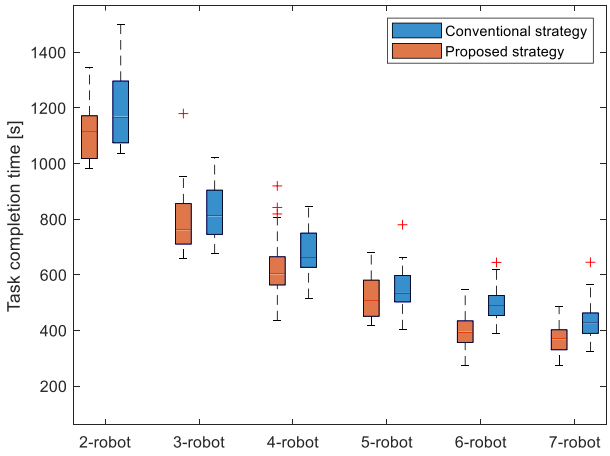


Fig. 11. Exploration accomplishment time of the robots using the proposed strategy and conventional strategy in [16]. The results in each case are collected from 50 trials with different initial positions.

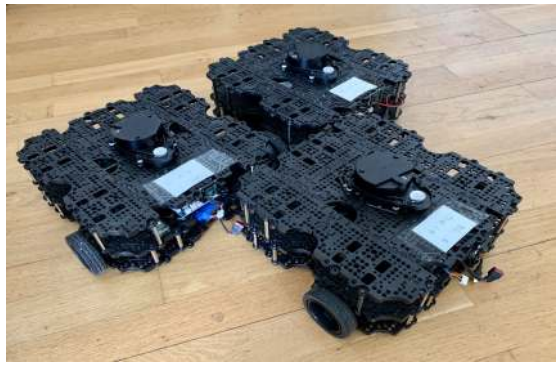


Fig. 12. Three TurtleBot3 Waffle Pi mobile robots equipped with Lidar were used in the experiments.

B. Experimental Validation

In the real-robot experiment, we used three Turtlebot3 Waffle Pi mobile robots (as shown in Fig. 12) to test the feasibility of the proposed autonomous exploration strategy and learning based collision avoidance algorithm in the real world. Each robot was equipped with a 360° Lidar for detecting the frontier points in the unknown environment. The communication between neighboring robots was achieved via Robot Operating System (ROS). The Raspberry Pi 3 board was installed on each Turtlebot3 for sending sensor information to and receiving control command from the host computer with Nvidia GTX 1080 GPU and Intel Core i9 CPU (2.9 GHZ). The lidar sensing range was set as 1 m in the experiment. In the experiments, we used virtual information nodes to validate the proposed exploration strategy. Those virtual information nodes and associated frontier points were generated in the synchronized map on the robots' storage devices. In the future, more advanced robotic platforms can be developed to carry and place real information nodes via manipulator arms.

In this case study, the objective of the mobile robot team was to explore an unknown room. The initial position of each robot is shown in Fig. 13. All the robots were placed at the middle of the room. Each robot deployed an information node at the initial position and then started moving in different directions due to the usage of Voronoi-based algorithms. In order to test the robustness of the multi-robot system against dynamic environments, an obstacle was suddenly placed during the mission. In Fig. 14, a cuboid obstacle was placed in front of the second mobile robot while it was approaching the next frontier point.

Fig. 13. Real-world experiment: the map is generated using gmapping for illustration. Real-time third person view and Turtlebots camera views are shown on the left side.

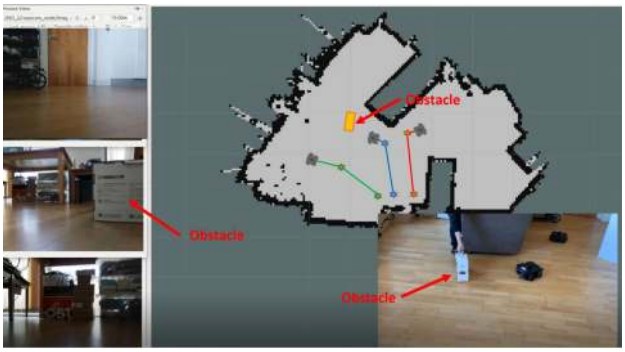


Fig. 14. Obstacle interruption: one obstacle is placed randomly during the Turtlebots mission.

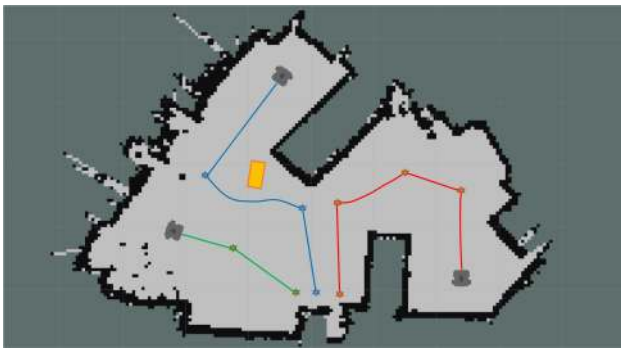


Fig. 15. Final trajectories of three Turtlebot3.

Based on the deep reinforcement learning training, the robot quickly adjusts its path to avoid the unexpected disturbance. The final trajectories of the three robots are illustrated in Fig. 15. No frontier point can be observed in the environment and the cooperative exploration task is accomplished. Besides, it can also be seen that the Voronoi-based strategy minimizes the control effort with respect to the mission completion time and the total travel distances. The video of the experimental results is shown in the Supplementary Material.

V. Conclusion

In this paper, a novel cooperative exploration strategy and deep reinforcement learning based mapless collision avoidance algorithm are proposed for multiple mobile robots in unknown environments. In order to navigate and coordinate the networked robots efficiently during the collaborative tasks, dynamic Voronoi partitions are generated to minimize duplicated exploration areas when using multiple robots. A utility function that takes into account both the path cost and the target distance is designed to determine a desired next frontier point such that depth-first and breadth-first modes can be chosen based on different scenarios. To deal with sudden appearance of obstacles in the unknown environment, a deep reinforcement learning based collision avoidance algorithm was proposed which integrated DDPG and PER algorithm to enable the control policy to learn from human demonstration data and improve the learning speed and performance of DDPG. The proposed algorithm was able to perform collision free maneuvering during 20 random missions in the new environment, requiring only 40 minutes of training time. By comparison, the proposed method only needs 14% training steps and 25% of the training time required by DDPG and generates safer and smoother trajectories than DDPG. Simulation results and hardware experiments using real robots are provided to demonstrate the effectiveness of the proposed multi-robot cooperative exploration scheme and learning based collision avoidance algorithm. In the future, finite-time consensus algorithm [37] will be exploited to coordinate heterogeneous robotic systems including both aerial and ground vehicles.

References

[1] H. Lee, H. Kim, and H. J. Kim, "Planning and control for collision-free cooperative aerial transportation," *IEEE Trans. Automat. Sci. Eng.*, vol. 15, no. 1, pp. 189–201, Jan. 2018.

[2] J. Hu, P. Bhowmick, and A. Lanzon, "Distributed adaptive time-varying group formation tracking for multiagent systems with multiple leaders on directed graphs," *IEEE Trans. Control Netw. Syst.*, vol. 7, no. 1, pp. 140–150, Mar. 2020.

[3] X. Zhou, W. Wang, T. Wang, Y. Lei, and F. Zhong, "Bayesian reinforcement learning for multi-robot decentralized patrolling in uncertain environments," *IEEE Trans. Veh. Technol.*, vol. 68, no. 12, pp. 11 691–11 703, Dec. 2019.

[4] J. Zhang, J. Yan, and P. Zhang, "Multi-UAV formation control based on a novel back-stepping approach," *IEEE Trans. Veh. Technol.*, vol. 69, no. 3, pp. 2437–2448, Mar. 2020.

[5] J. Hu, P. Bhowmick, and A. Lanzon, "Two-layer distributed formation-containment control strategy for linear swarm systems: Algorithm and experiments," *Int. J. Robust Nonlinear Control*, vol. 30, no. 16, pp. 6433–6453, 2020.

[6] X. Liu, Y. Liu, Y. Chen, and L. Hanzo, "Trajectory design and power control for multi-UAV assisted wireless networks: A machine learning approach," *IEEE Trans. Veh. Technol.*, vol. 68, no. 8, pp. 7957–7969, Aug. 2019.

[7] J. Hu, A. E. Turgut, T. Krajník, B. Lennox, and F. Arvin, "Occlusion-based coordination protocol design for autonomous robotic shepherding tasks," *IEEE Trans. Cogn. Develop. Syst.*, to be published, doi: 10.1109/TCDS.2020.3018549.

[8] Y. Feng, D. He, and Y. Guan, "Composite platoon trajectory planning strategy for intersection throughput maximization," *IEEE Trans. Veh. Technol.*, vol. 68, no. 7, pp. 6305–6319, Jul. 2019.

[9] J. Hu, P. Bhowmick, F. Arvin, A. Lanzon, and B. Lennox, "Cooperative control of heterogeneous connected vehicle platoons: An adaptive leader-following approach," *IEEE Robot. Automat. Lett.*, vol. 5, no. 2, pp. 977–984, Apr. 2020.

[10] M. Selin, M. Tiger, D. Duberg, F. Heintz, and P. Jensfelt, "Efficient autonomous exploration planning of large-scale 3D environments," *IEEE Robot. Automat. Lett.*, vol. 4, no. 2, pp. 1699–1706, Apr. 2019.

[11] Z. Meng *et al.*, "A two-stage optimized next-view planning framework for 3D unknown environment exploration, and structural reconstruction," *IEEE Robot. Automat. Lett.*, vol. 2, no. 3, pp. 1680–1687, Jul. 2017.

[12] Y. Zhang, G. Tian, J. Lu, M. Zhang, and S. Zhang, "Efficient dynamic object search in home environment by mobile robot: A priori knowledge-based approach," *IEEE Trans. Veh. Technol.*, vol. 68, no. 10, pp. 9466–9477, Oct. 2019.
[13] C. Wang, W. Chi, Y. Sun, and M. Q.-H. Meng, "Autonomous robotic exploration by incremental road map construction," *IEEE Trans. Automat. Sci. Eng.*, vol. 16, no. 4, pp. 1720–1731, Oct. 2019.
[14] F. Niroui, K. Zhang, Z. Kashino, and G. Nejat, "Deep reinforcement learning robot for search and rescue applications: Exploration in unknown cluttered environments," *IEEE Robot. Automat. Lett.*, vol. 4, no. 2, pp. 610–617, Apr. 2019.
[15] D. Fox, W. Burgard, H. Kruppa, and S. Thrun, "A probabilistic approach to collaborative multi-robot localization," *Auton. Robot.*, vol. 8, no. 3, pp. 325–344, 2000.
[16] J. Kim, "Cooperative exploration and networking while preserving collision avoidance," *IEEE Trans. Cybern.*, vol. 47, no. 12, pp. 4038–4048, Dec. 2016.
[17] A. Franchi, L. Freda, G. Oriolo, and M. Vendittelli, "The sensor-based random graph method for cooperative robot exploration," *IEEE/ASME Trans. Mechatronics*, vol. 14, no. 2, pp. 163–175, Apr. 2009.
[18] M. Corah, C. OMeadhra, K. Goel, and N. Michael, "Communication-efficient planning and mapping for multi-robot exploration in large environments," *IEEE Robot. Automat. Lett.*, vol. 4, no. 2, pp. 1715–1721, Apr. 2019.
[19] H. Qin *et al.*, "Autonomous exploration and mapping system using heterogeneous UAVs and UGVs in GPS-denied environments," *IEEE Trans. Veh. Technol.*, vol. 68, no. 2, pp. 1339–1350, Feb. 2019.
[20] H. Niu, Y. Lu, A. Savvaris, and A. Tsourdos, "An energy-efficient path planning algorithm for unmanned surface vehicles," *Ocean Eng.*, vol. 161, pp. 308–321, 2018.
[21] H. Niu, A. Savvaris, A. Tsourdos, and Z. Ji, "Voronoi-visibility roadmap-based path planning algorithm for unmanned surface vehicles," *J. Navigation*, vol. 72, no. 4, pp. 850–874, 2019.
[22] Y. Huang *et al.*, "A motion planning and tracking framework for autonomous vehicles based on artificial potential field elaborated resistance network approach," *IEEE Trans. Ind. Electron.*, vol. 67, no. 2, pp. 1376–1386, Feb. 2020.
[23] J. Sun, J. Tang, and S. Lao, "Collision avoidance for cooperative UAVs with optimized artificial potential field algorithm," *IEEE Access*, vol. 5, pp. 18 382–18 390, 2017.
[24] U. Orozco-Rosas, O. Montiel, and R. Sepúlveda, "Mobile robot path planning using membrane evolutionary artificial potential field," *Appl. Soft Comput.*, vol. 77, pp. 236–251, 2019.
[25] J. Lim, S. Ha, and J. Choi, "Prediction of reward functions for deep reinforcement learning via Gaussian process regression," *IEEE/ASME Trans. Mechatronics*, vol. 25, no. 4, pp. 1739–1746, Aug. 2020.
[26] Y. Yuan, R. Tasik, S. S. Adhatarao, Y. Yuan, Z. Liu, and X. Fu, "Race: Reinforced cooperative autonomous vehicle collision avoidance," *IEEE Trans. Veh. Technol.*, vol. 69, no. 9, pp. 9279–9291, Sep. 2020.
[27] P. Long, T. Fanl, X. Liao, W. Liu, H. Zhang, and J. Pan, "Towards optimally decentralized multi-robot collision avoidance via deep reinforcement learning," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2018, pp. 6252–6259.
[28] L. Xiao, D. Jiang, Y. Chen, W. Su, and Y. Tang, "Reinforcement-learning-based relay mobility and power allocation for underwater sensor networks against jamming," *IEEE J. Ocean. Eng.*, vol. 45, no. 3, pp. 1148–1156, Jul. 2020.
[29] J. Zhang, J. T. Springenberg, J. Boedecker, and W. Burgard, "Deep reinforcement learning with successor features for navigation across similar environments," in *Proc. IEEE/RSJ Int. Conf. Intell. Robot. Syst.*, 2017, pp. 2371–2378.
[30] J. Choi, K. Park, M. Kim, and S. Seok, "Deep reinforcement learning of navigation in a complex and crowded environment with a limited field of view," in *Proc. Int. Conf. Robot. Automat.*, 2019, pp. 5993–6000.
[31] D. Zhu, T. Li, D. Ho, C. Wang, and M. Q.-H. Meng, "Deep reinforcement learning supervised autonomous exploration in office environments," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2018, pp. 7548–7555.
[32] A. Savvaris, H. Niu, H. Oh, and A. Tsourdos, "Development of collision avoidance algorithms for the C-enduro USV," in *Proc. 19th IFAC World Congr.*, 2014, vol. 47, pp. 12 174–12 181.
[33] L. Tai, G. Paolo, and M. Liu, "Virtual-to-real deep reinforcement learning: Continuous control of mobile robots for mapless navigation," in *Proc. IEEE/RSJ Int. Conf. Intell. Robot. Syst.*, 2017, pp. 31–36.
[34] W. Burgard, M. Moors, C. Stachniss, and F. E. Schneider, "Coordinated multi-robot exploration," *IEEE Trans. Robot.*, vol. 21, no. 3, pp. 376–386, Jun. 2005.
[35] T. P. Lillicrap *et al.*, "Continuous control with deep reinforcement learning," 2015, *arXiv:1509.02971*.
[36] N. Koenig and A. Howard, "Design and use paradigms for gazebo, an open-source multi-robot simulator," in *Proc. IEEE/RSJ Int. Conf. Intell. Robot. Syst.*, 2004, vol. 3, pp. 2149–2154.
[37] J. Hu and A. Lanzon, "Distributed finite-time consensus control for heterogeneous battery energy storage systems in droop-controlled microgrids," *IEEE Trans. Smart Grid*, vol. 10, no. 5, pp. 4751–4761, Sep. 2019.

Junyan Hu received the B.Eng. degree in automation from the Hefei University of Technology, in 2015 and the Ph.D. degree in electrical and electronic engineering from the University of Manchester, in 2020.

Dr. Hu is currently a Postdoctoral Research Associate in Robotics with the University of Manchester. His research interests include multi-robot coordination, cooperative control, autonomous vehicles, and swarm intelligence.

Hanlin Niu received the B.Eng. degree in mechanics from Tianjin University, in 2012 and the Ph.D. degree in aeronautical engineering from Cranfield University, in 2018. From 2017 to 2019, he was a Research Associate with Robotics, Cardiff University. Since 2019, he has been working as a Research Associate with Robotics, the University of Manchester. His research interests include path planning, path following, collision avoidance, deep reinforcement learning of autonomous vehicles, and tele-operation of robotic arm.

Joaquin Carrasco was born in Abarn, Spain, in 1978. He received the B.Sc. degree in physics and the Ph.D. degree in control engineering from the University of Murcia, Murcia, Spain, in 2004 and 2009, respectively. He is a Senior Lecturer with the Control Systems Centre, Department of Electrical and Electronic Engineering, University of Manchester, U.K. From 2009 to 2010, he was with the Institute of Measurement and Automatic Control, Leibniz Universitt Hannover, Hannover, Germany. From 2010 to 2011, he was a Research Associate with the Control Systems Centre, School of Electrical and Electronic Engineering, University of Manchester, U.K. His current research interests include absolute stability, multiplier theory, and robotics applications.

Barry Lennox (Senior Member, IEEE) is a Professor of applied control and nuclear engineering decommissioning and holds a Royal Academy of Engineering Chair in Emerging Technologies. He is a Director of the Robotics and Artificial Intelligence for Nuclear (RAIN) Robotics Hub and Research Director of the Dalton Cumbrian Facility. He is a Fellow of the Royal Academy of Engineering, Fellow of the IET and InstMC, and a Chartered Engineer. He is an expert in applied control and its use in robotics and process operations and has considerable experience in transferring leading edge technology into industry.

Farshad Arvin received the B.Sc. degree in computer engineering, the M.Sc. degree in computer systems engineering, and the Ph.D. degree in computer science, in 2004, 2010, and 2015, respectively. Since 2018, he has been an Assistant Professor in Robotics with the University of Manchester, U.K. He visited several leading institutes including Artificial Life Laboratory with the University of Graz, Institute of Microelectronics, Tsinghua University, Beijing, and Italian Institute of Technology (IIT) in Genoa as a Senior Visiting Research Scholar. His research interests include swarm robotics and autonomous systems. He is the Founding Director of the Swarm & Computation Intelligence Laboratory (SwaCIL) formed in 2018.