Published:
IEEE Transactions on Vehicular Technology (Dec 2020)

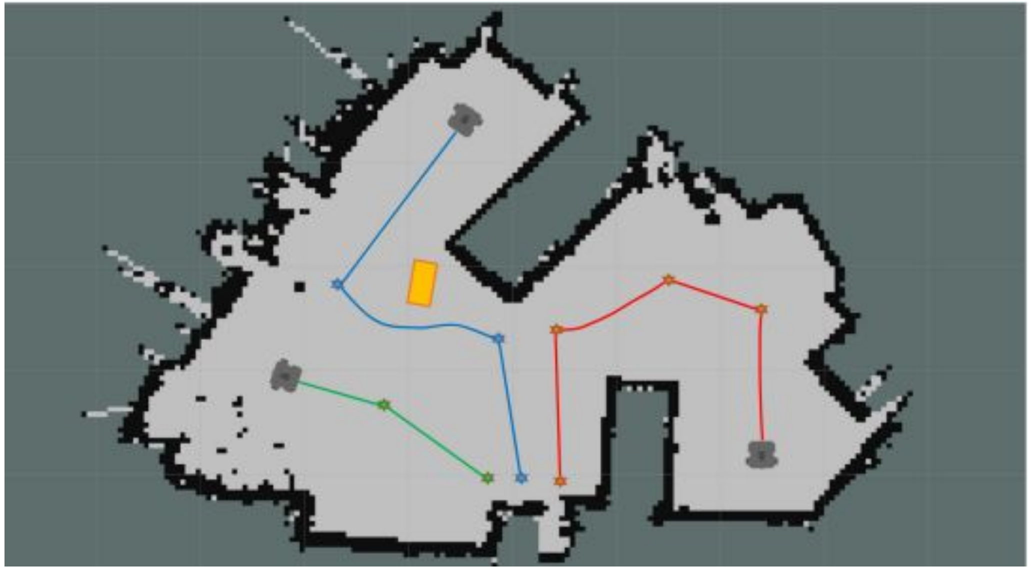
Voronoi-Based Multi-Robot Autonomous Exploration in Unknown Environments via Deep Reinforcement Learning

Authors:
Su Liu
Hanlin Niu
Joaquin Carrasco
Barry Lennox
Farshad Arvin

Presented By:
Ittehad Saleh Chowdhury
Reyadath Ullah Akib

Multi-Robot Autonomous Exploration

Autonomous exploration refers to the process where a team of networked robots collaboratively navigate and investigate an unknown environment without human intervention.



Significance of Autonomous Exploration

Few usefulness of Autonomous Exploration are:

- Search and Rescue Operation
- Monitoring and Surveillance
- Hazardous Material Handling
- Mining and Exploration
- Military Defense
- Space Exploration
- Warehouse Management

Primary Objective

In this paper a novel **cooperative exploration strategy** has been proposed for multiple mobile robots, which reduces the **overall task completion time** and **energy** compared to conventional methods.

This is achieved by focusing mainly on two things:

- **Cooperative exploration approach developed using Voronoi partition for efficient navigation.**
- **Integrated Deep Reinforcement Learning based collision avoidance algorithm**

In the end we discuss the feasibility of the proposed exploration strategy and it’s comparison with traditional methods

Technical Background

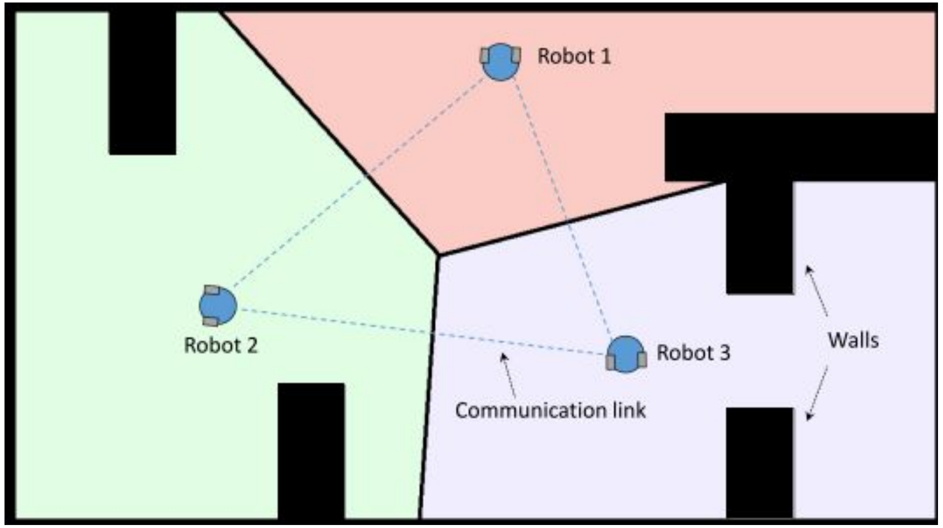
Communication Network

We consider a weighted and un-directed information graph I = (⋎,ε,A) with,

Set of nodes ⋎ = {1,2,..N}

Set of edges Ɛ ⊂ ⋎ x ⋎ and

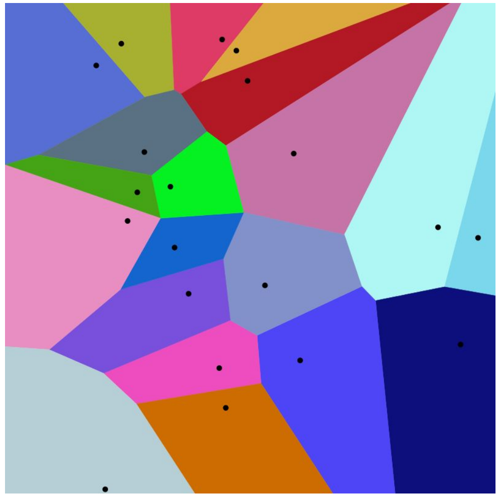
The associated adjacency matrix A = [a_{ij}]



Voronoi Partitions

To coordinate a multi-robot team effectively in exploring an unknown area, each robot is assigned a different target location based on dynamic Voronoi partition to avoid duplicated exploration areas.

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.



Voronoi Partitions (Cont.)

The area to be mapped can be broken up dynamically among the robot team members based on their current locations in the following way:

The Voronoi cell Vor(R_i) is defined by:

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

Q ⊂ $\mathbb{R}^2$

$R_i\ \forall i \in \{1, 2, \ldots, N\}$ are networked robots

P_i represents the position of robot R_i

$\mathcal{N}_{R_i}$ is a robot satisfying that each element in this set is a neighbor of R_i

Collision Avoidance Problem Formulation

The Collision Avoidance problem can be formulated as to find the translation function:

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

where,

l_t = laser range data at time t

P_t = relative position of the target

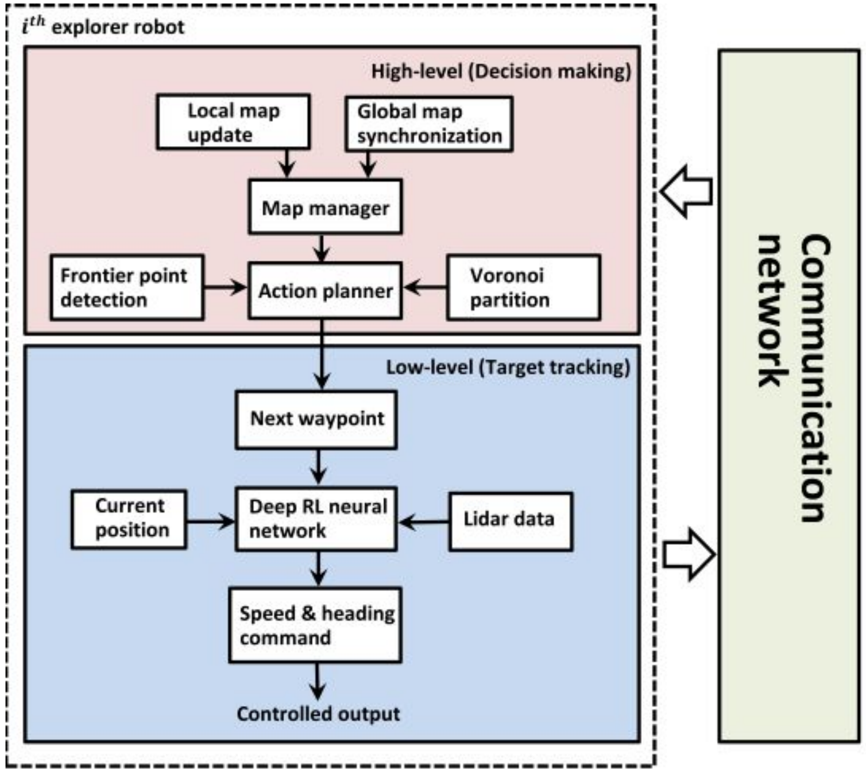
v_{t-1} = velocity command at last time stamp

Cooperative Exploration Strategy

Hierarchical Control Architecture

Two Layer Control Architecture:

1. High-level Decision making layer
2. Low-level Target tracking layer



Hierarchical Control Architecture (Cont.)

The cooperative exploration technique is introduced based on the following set of presumptions:

- **Omnidirectional Sensory System:** Robots are equipped with sensors for perceiving surrounding space and detecting obstacle with **sensing ranges (r_s)**
- **Communication Capabilities:** Robots are capable of broadcasting stored information and receive relative positions from neighboring robots with **communication range (r_c)**
- **(r_c) > (r_s)**

Exploration Strategy

The Exploration Strategy for the Multi-Robot Autonomous Exploration is mainly based on two techniques:

- **Frontier based exploration technique:** An exploration technique that directs robots to focus on the boundaries between known and unknown areas, allowing for systematic and efficient exploration of unfamiliar environments.
- **Information Nodes:** Building of an information network in an unknown environment by deploying information nodes.

This strategy allows for a Decentralized Control Design

Exploration Strategy (Cont.)

The robot explores the unexplored areas by selecting one of the Frontier points based on the minimum value of a utility function:

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

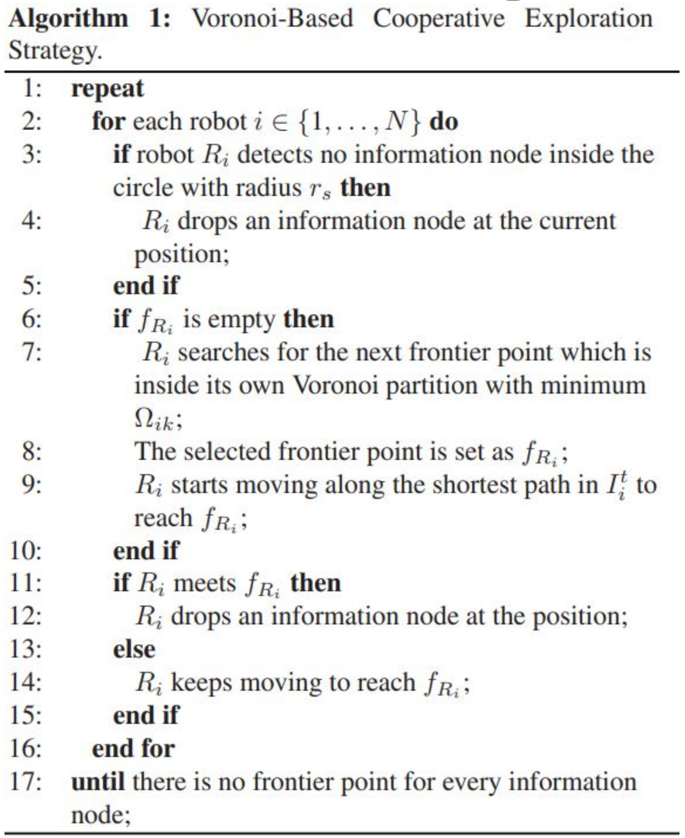
d_{ik} = Distance between R_i and frontier point k

$ø_{ik}$ = Distance between initial position and frontier point k

λ = Constant ($0 \leq \lambda \leq 1$)

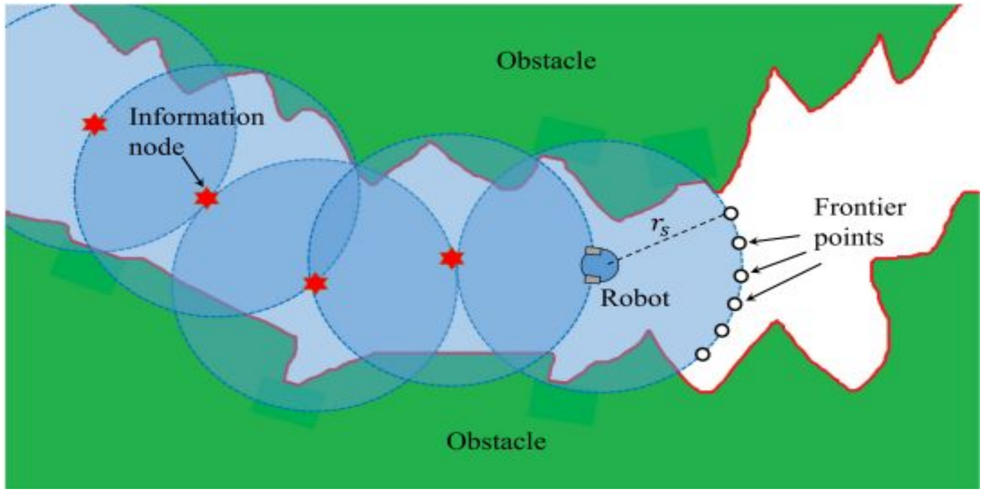
The information nodes work as a network to navigate the map efficiently

Coordination Algorithm for Explorer Robots



Algorithm 1: Voronoi-Based Cooperative Exploration Strategy.

```
1:  repeat
2:    for each robot $i \in \{1, \dots, N\}$ do
3:      if robot $R_i$ detects no information node inside the circle with radius $r_s$ then
4:        $R_i$ drops an information node at the current position;
5:      end if
6:      if $f_{R_i}$ is empty then
7:        $R_i$ searches for the next frontier point which is inside its own Voronoi partition with minimum $\Omega_{ik}$;
8:        The selected frontier point is set as $f_{R_i}$;
9:        $R_i$ starts moving along the shortest path in $I_i^t$ to reach $f_{R_i}$;
10:     end if
11:     if $R_i$ meets $f_{R_i}$ then
12:       $R_i$ drops an information node at the position;
13:     else
14:       $R_i$ keeps moving to reach $f_{R_i}$;
15:     end if
16:   end for
17: until there is no frontier point for every information node;
```



Deep Reinforcement Learning Setup

Reward Space

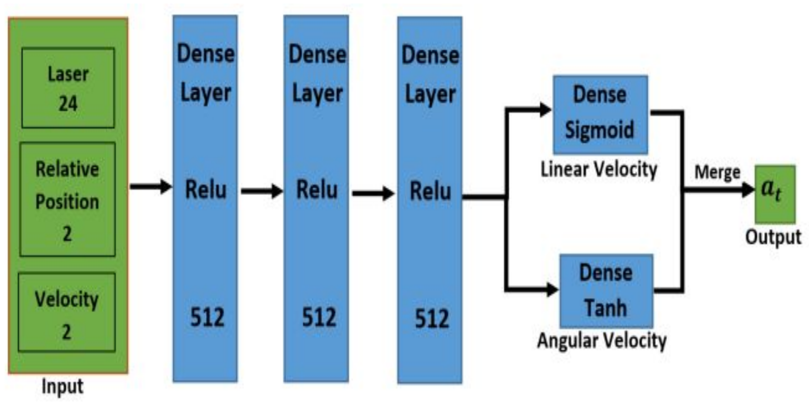
The reward function is defined as

$$r = r_d + r_{cl} + r_{av} + r_{lv}$$

- **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**
- **r_{lv}** is the **linear velocity reward**

Network Structure

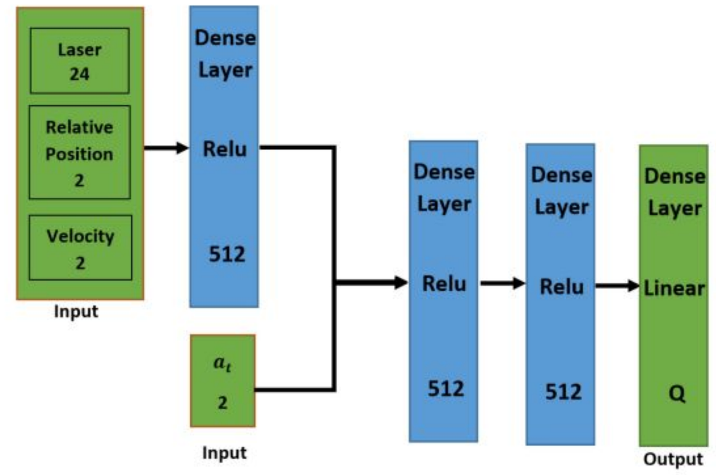
We are using an Actor-Critic network architecture (DDPG) that can act on continuous action spaces.



The input is a 28-dimensional vector connected to three dense layers with each layer having 512 nodes.

It generates the linear and angular velocity command that is merged into an action state $\mathbf{a}_t$

Continued (The Critic Network)

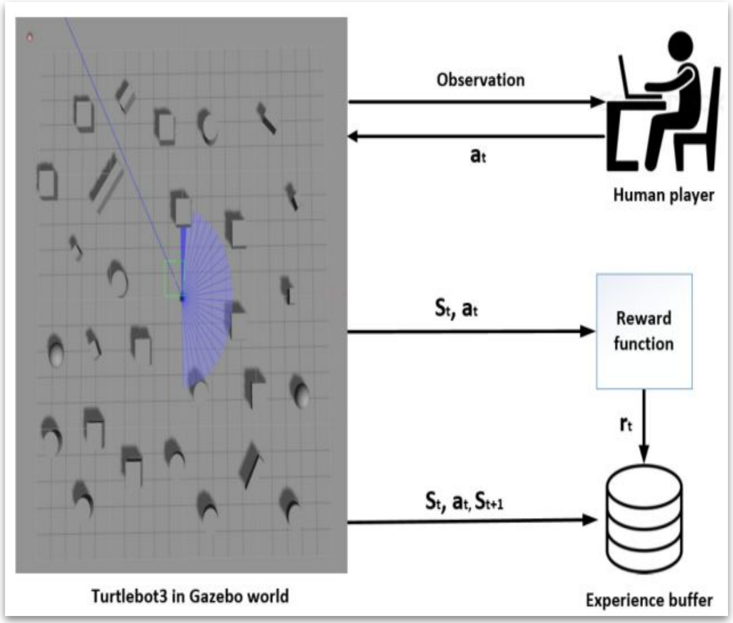


The state data (28D vector) is also used as the input of the critic network which is processed by three dense layers and each layer has 512 nodes.

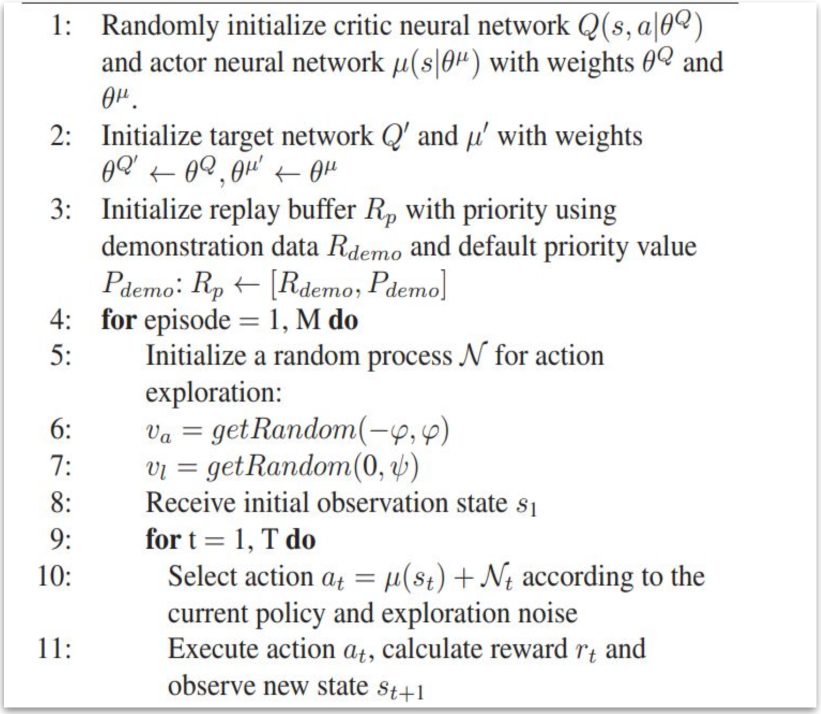
The action input is merged with the second layer, and the Q-value is finally generated by a linear activation function

Integration of DDPG and PER

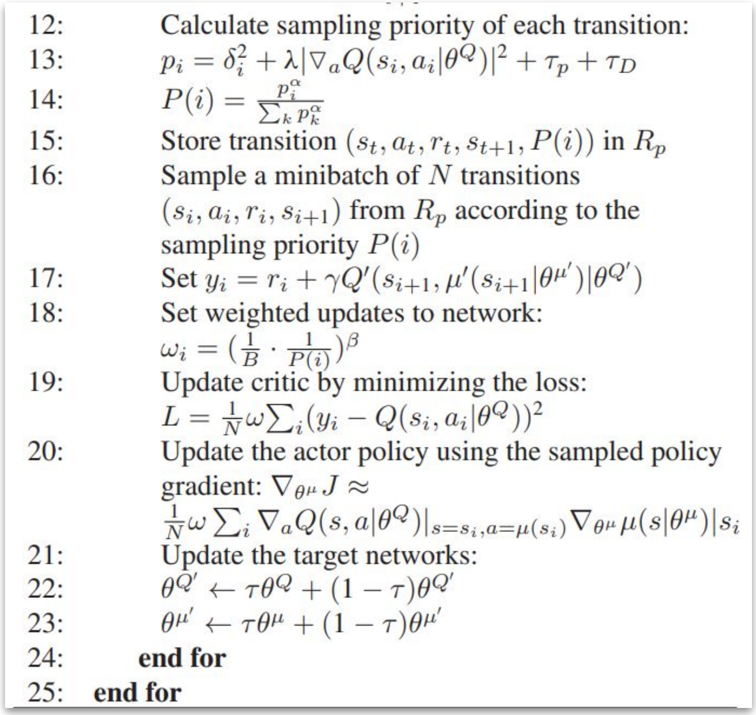
- We integrate DDPG with PER algorithm using human demonstration data.
- Instead of sampling data uniformly, we integrate DDPG and PER algorithm to sample important data more frequently by calculating the priority of each state transition
- In this research, human demonstration data $\boldsymbol{R_{demo}}$ is recorded before training the networks
- The priority for human demonstration data is typically set to be higher.



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 $\theta^Q$ and
    $\theta^\mu$.
2:  Initialize target network $Q'$ and $\mu'$ 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
```

Algorithm 2 (Continued)

- In line 13 we use the following equation to calculate the priority for a given transition

Calculate sampling priority of each transition:
$$p_i = \delta_i^2 + \lambda|\nabla_a Q(s_i, a_i|\theta^Q)|^2 + \tau_p + \tau_D$$

- Here δ means the Temporal Difference (TD) error, the second term denotes the actor loss. ($\lambda|\nabla_a Q(s_i, a_i|\theta^Q)|^2$)
- The sampling probability is calculated using $P(i) = \frac{p_i^\alpha}{\sum_k p_k^\alpha}$
- We then store transition $(s_t, a_t, r_t, s_{t+1}, P(i))$ in the replay buffer $\mathbf{R_p}$

Complexity Analysis

- The complexity of the actor neural network is $O(S_{ain}F_{a1}F_{a2}F_{a3}S_{aout})$
- The complexity of the critic neural network is $O((S_{cin}F_{c1} + S_{aout})F_{c2}F_{c3}S_{cout})$
- Considering that the collision avoidance algorithm can be deployed on each mobile robot independently, the total computational complexity can be described by ***O(NM)***
    - **N =** number of mobile robots
    - **M =** $S_{ain}F_{a1}F_{a2}F_{a3}S_{aout} + (S_{cin}F_{c1} + S_{aout})F_{c2}F_{c3}S_{cout}$

RESULTS AND ANALYSIS

Software and Hardware Used:

› The Turtlebot3 Waffle Pi

› Gazebo simulator

› Robot Operating System (ROS)

› Host computer [GTX 1080 GPU Intel i9 CPU (2.9 GHZ)]

Collision Avoidance Policy Training and Evaluation

- The robot is commanded to go to the target position while avoiding the obstacles
- Once the robot arrives at the target or collides with an obstacle, it will reset to origin
- 3000 steps of demonstration data were sampled for training
- The Red sphere (target) is placed randomly

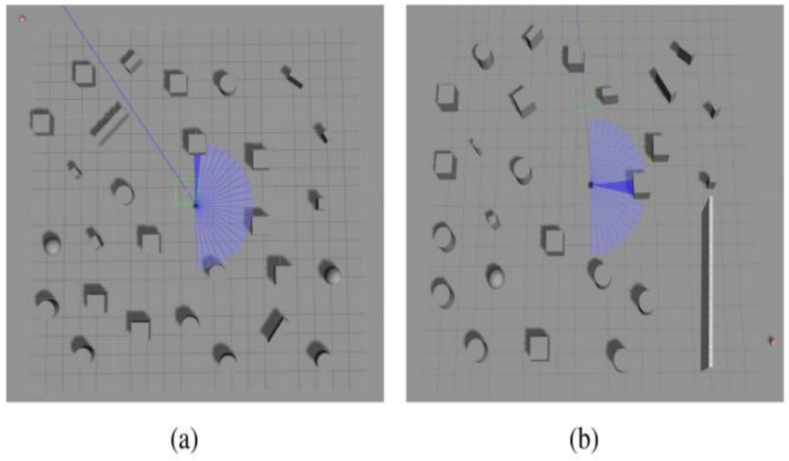
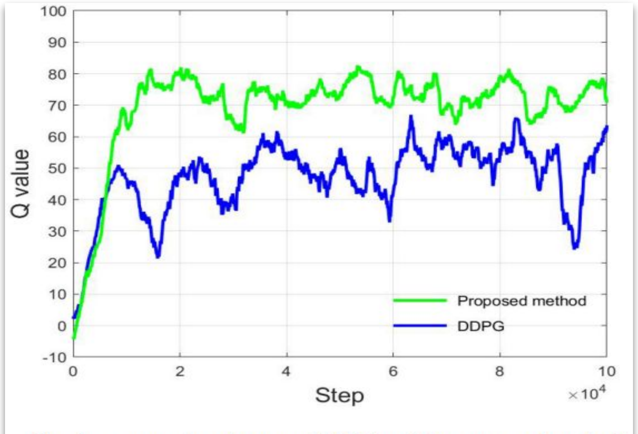


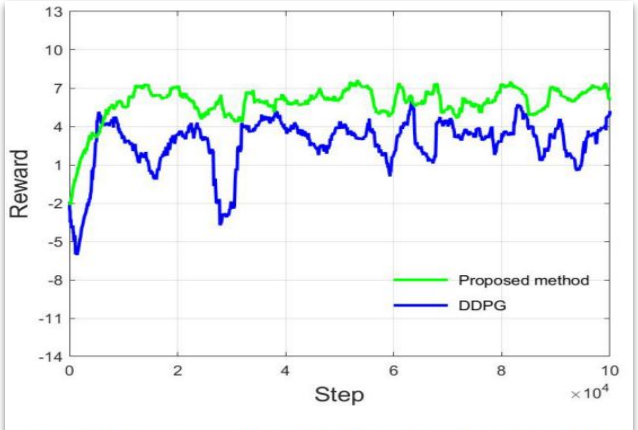
Fig. 7. Gazebo environment for (a) training and (b) evaluation: The red sphere represents the target location; the blue lines stand for the laser beam around the Turtlebot3; the gray cubes, cuboids, cylinders and spheres are used as obstacles.

Q and Reward Value Comparison

- Q-value of the proposed method arrives at 67 using 8900 steps
- Original DDPG needs 63310 steps
- The reward value of the proposed method increases faster than DDPG in the beginning
- We compared the trained model after only 10000 training steps of the proposed method and the trained model after 50000 training steps of DDPG



Q-value comparison between DDPG and the proposed method.

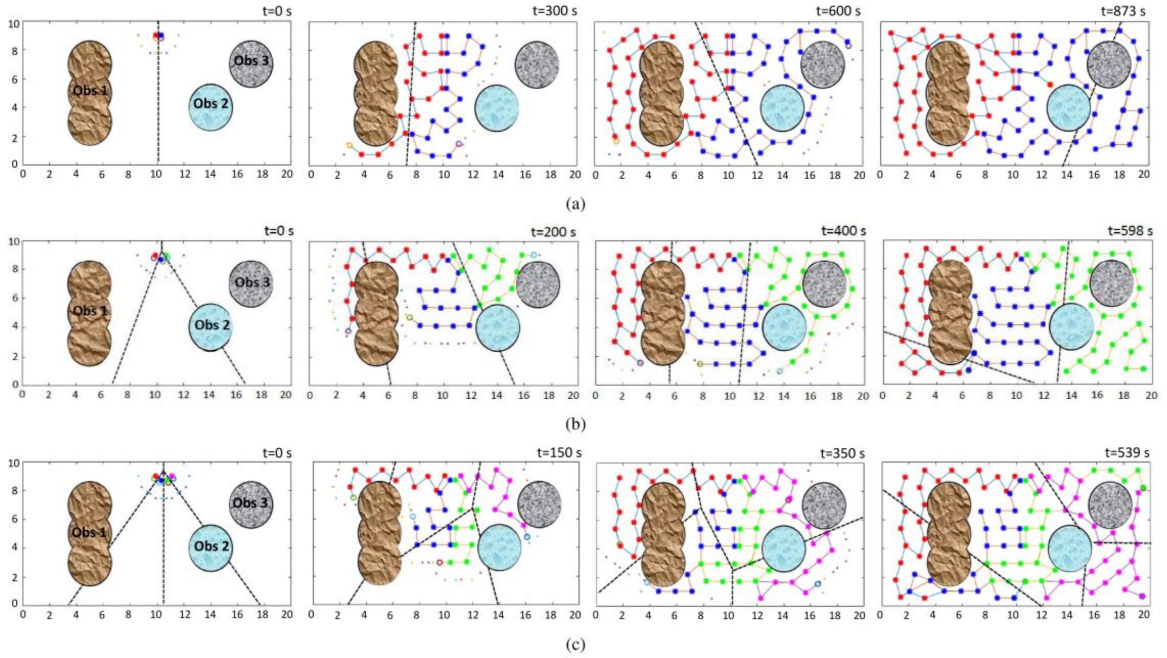


Reward comparison between DDPG and the proposed method.

Completion Rate and Time

- Proposed method enables the robots to complete all 20 missions successfully
- In comparison with the proposed method, the DDPG method only completed 15 of the 20 random missions
- Collecting and training 10000 steps data of the proposed algorithm cost 40 minutes Compared to 2 hours and 40 minutes of DDPG (50000 steps)
- The proposed method enables mobile robot to learn faster, only requiring 14% steps and 25% of the training time required by DDPG

Performance of the Proposed Cooperative Exploration Strategy



Comparing the Proposed Method With the Approach in [16]

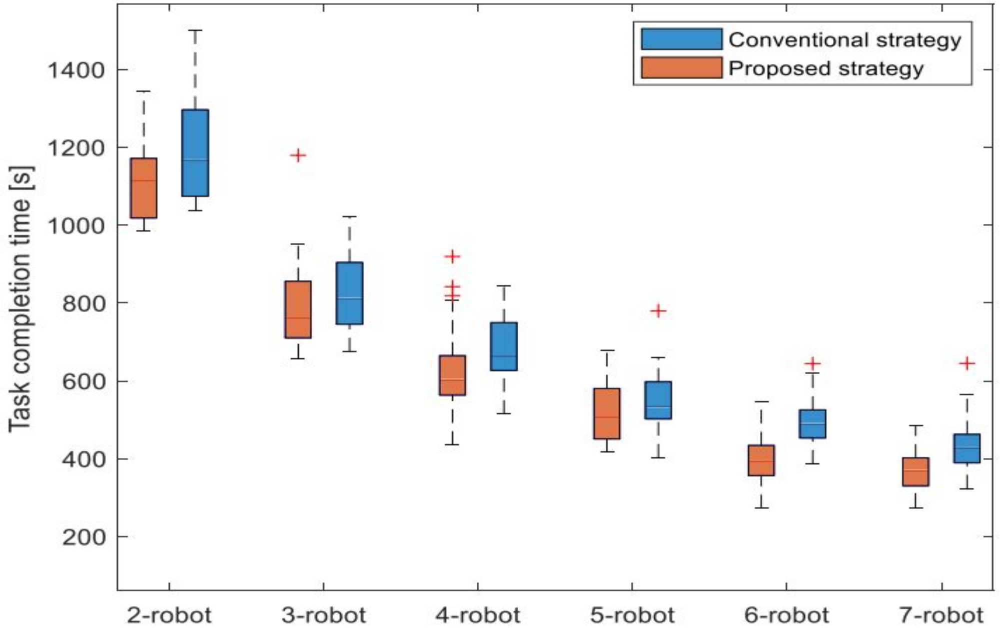
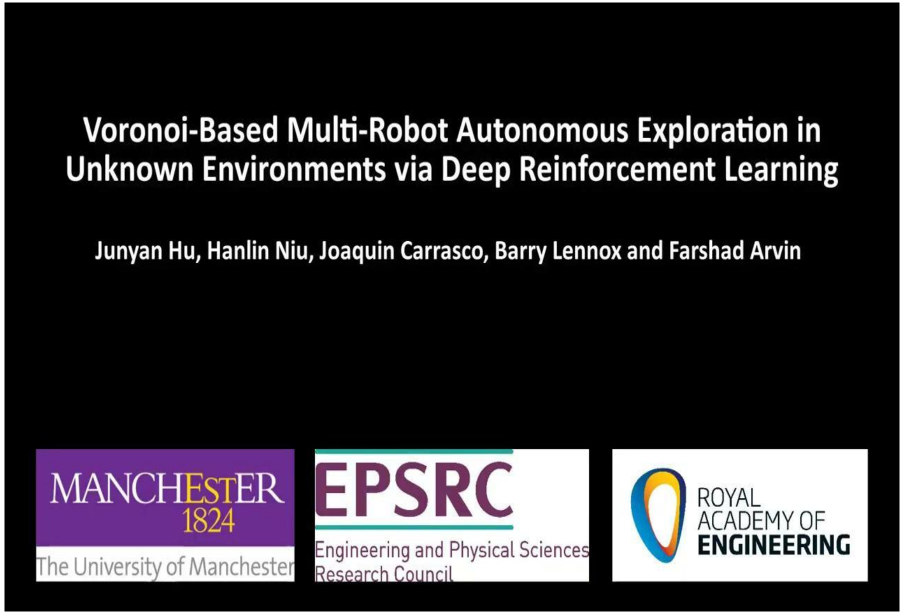


Fig. 11. Exploration accomplishment time of the robots using the proposed strategy and conventional strategy in [16]. The results in each case are collected from 50 trials with different initial positions.

Experimental Validation

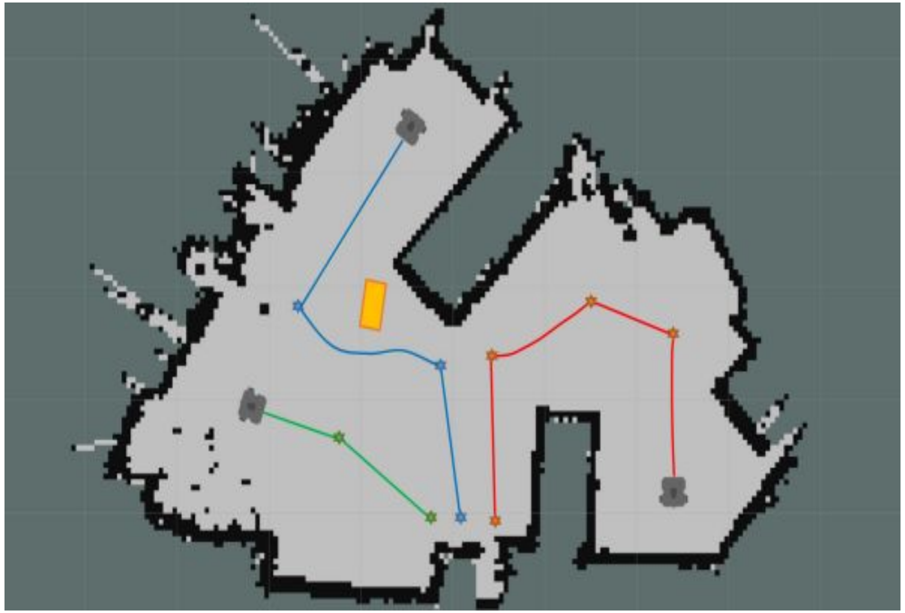

Voronoi-Based Multi-Robot Autonomous Exploration in Unknown Environments via Deep Reinforcement Learning

Junyan Hu, Hanlin Niu, Joaquin Carrasco, Barry Lennox and Farshad Arvin

MANCHESTER 1824 The University of Manchester

EPSRC Engineering and Physical Sciences Research Council

ROYAL ACADEMY OF ENGINEERING



Thank You!