IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received August 9, 2020, accepted September 15, 2020, date of publication September 21, 2020, date of current version October 20, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.3025287

Decentralized Control of Multi-Robot System in Cooperative Object Transportation Using Deep Reinforcement Learning

LIN ZHANG, YUFENG SUN, ANDREW BARTH, (Member, IEEE), AND OU MA
Department of Aerospace Engineering and Engineering Mechanics, University of Cincinnati, Cincinnati, OH 45040, USA
Corresponding author: Lin Zhang (lin.zhang@uc.edu)

ABSTRACT Object transportation could be a challenging problem for a single robot due to the oversize and/or overweight issues. A multi-robot system can take the advantage of increased driving power and more flexible configuration to solve such a problem. However, an increased number of individuals also changed the dynamics of the system which makes control of a multi-robot system more complicated. Even worse, if the whole system is sitting on a centralized decision making unit, the data flow could be easily overloaded due to the upscaling of the system. In this research, we propose a decentralized control scheme on a multi-robot system with each individual equipped with a deep Q-network (DQN) controller to perform an oversized object transportation task. DQN is a deep reinforcement learning algorithm, thus does not require the knowledge of system dynamics, instead, it enables the robots to learn appropriate control strategies through trial-and-error style interactions within the task environment. Since analogous controllers are distributed on the individuals, the computational bottleneck is avoided systematically. We demonstrate such a system in a scenario of carrying an oversized rod through a doorway by a two-robot team. The presented multi-robot system learns abstract features of the task and cooperative behaviors are observed. The decentralized DQN-style controller is showing strong robustness against uncertainties. In addition, We propose a universal metric to assess the cooperation quantitatively.

INDEX TERMS Cooperative object transportation, decentralized control, deep Q-network, multi-robot system.

I. INTRODUCTION

In the world of humans, complex tasks require multiple persons to cooperate mentally and physically. For example, in the Space Shuttle STS-49 mission [1], NASA originally planned to have only one astronaut to capture the slow-rotating satellite IntelSat but failed to accomplish the task. The task failed again the second day with two astronauts working together. The task was finally accomplished on the fifth day by three astronauts and one robot (the Canadarm) worked together. The mission set several records including maximum number of astronauts in a space walk and longest hours in a single spacewalk [1]. The main reason such a task requires multiple astronauts is that the size of the IntelSat is overwhelming for a single astronaut. We can easily imagine that a single robot is facing the same challenge when handling oversized objects. A multi-robot system (MRS) can achieve the goal but deploying such a system requires more advanced coordination and control strategies.

The associate editor coordinating the review of this manuscript and approving it for publication was Nikhil Padhi.

Oversized object transportation is a typical task that is usually composed with smaller subtasks which can be assigned to multiple robots simultaneously [2]. We assume that the robots are physically attached to the object, and transport is achieved by either pushing or pulling (or both) the object. Decentralized architecture is the natural way to control a MRS in such a task as it is more flexible and more scalable [3]. To solve such a problem, on the one hand, an individual robot has to guarantee accomplishment of its own assignment. On the other hand, the robots have to cooperate with each other to achieve the shared high-level goal. However, the variation of the system (e.g. number of the robots, moving obstacles, unpredictable perturbations, etc.) can bring challenges toward management of the MRS. A leader/follower architecture is very popular in the past for it can plan and control on top of the leader robot's well-studied dynamics model [4], [5]. Besides that, researchers proved that an MRS with identical controllers equally distributed on individuals performs the task well [6]. However, treating all the members equally can result in more complicated dynamics of the integrated system.

Designing a good controller for a robot could be extremely challenging due to the complicated dynamics of the task. However, the performance of a controller can be easily evaluated through the more obvious success conditions. Instead of modeling dynamics of the system, reinforcement learning (RL) algorithms model the reward mechanisms which directly tells the goodness of a state that is the consequence of the controller's outputs. Therefore, we can optimize the controller with the guide of the reward function to achieve the goal without knowing the dynamics model at all [7]. The rapidly developing deep learning technologies further boost the RL into deep reinforcement learning (DRL) which enables handling more generally representative data (e.g. continuous state) and thus has been applied to many complex control problems [8]. DRL is also welcomed by the researchers aiming at solving cooperative object transportation tasks with MRS [9]–[11].

In this paper, we present an MRS controlled by decentralized DRL controllers performing a cooperative object transportation task. We expand the original DQN algorithm [8] to train distributed controllers in an MRS. Specifically, we instantiate the MRS with two homogeneous mobile robots with each robot driven by a differential driving mechanism. The task is transporting a long rod from inside of a room to the outside of it. The most challenging part is the robots need to transport the rod through a narrow doorway. We employ DQN controllers to regulate the behavior of the robots by estimating the total reward of taking a possible action at any given state then take the action with the highest value. Due to the DRL's nature of working on a value system, we propose to use absolute error of estimated state-action values between two robots to quantitatively measure how well they cooperate with each other. To our best knowledge, this is the first implementation of DRL algorithm in a cooperative transportation task by taking in high-level sensing data then output low-level control signals directly. The highlights of this research are:

- An intuitive cooperation metric is proposed to measure how different the individuals in an MRS value their situations at the same moment.
- The Controllers are equally distributed on individual robots. No command center is involved, no leader/follower pattern was specified;
- No dynamics modeling, no path planning. Controllers make decisions directly from the sensing data (end-to-end);

In the next section (section II), we will review some related researches. We will introduce our research methodology including configurations, algorithms and metrics definitions in section III. The experimental results and in-depth analyses will be described in section IV. We will summarize and look into the future of this research in section V

II. RELATED WORK

More researchers began to be interested in solving the problem of cooperative object transportation using MRS from the mid 90's [4], [12]–[15]. Although the settings could be varied a lot from each others, there are three major strategies to configure such a problem: 1) pushing-only strategy; 2) grasping strategy; 3) caging strategy [2]. Our research adopts the second strategy such that the robots coordinate with each others using different actions(pushing or pulling), while the robots are focusing on the transportation without considering the way they interact with the object. While centralized control schemes are rarely reported, there exists research using such an organization [16]. Nevertheless, a majority of the researchers adopt the distributed architectures, thus the following literature utilize decentralized control schemes.

When solving an MRS based cooperative object transportation problem with grasping strategy, the leader/follower configuration is very popular [4], [5], [17]–[19]. In general, a leader robot is responsible for initiating and directing the transportation, while the follower robots coordinate their actions with respect to the leader's guidance. It is true that the leader/follower architecture saves the cost of computation and communication, but this architecture sacrifices some flexibility of the follower robots. Under some complicated situations, involving a leader robot with superior capabilities does not make the problem easier.

Researchers investigated MRS with every individual playing the same role, such that the MRS can be more flexible in their tasks. A swarm of simulated mobile robots were introduced in a task of cooperatively transporting a payload around obstacles [20], however, kinematics modeling and path planning were required. A two-stage motion planning strategy was proposed in [21], [22] to help the MRS safely transport an object around dynamic obstacles. A research scenario that was closely analogous to ours was proposed in [23]. The researchers were able to control two omnidirectional mobile robots to transport an object through an narrow opening with decentralized sliding mode controllers. In this research, predefined trajectories and a dynamics model of the MRS needs to be established beforehand. An MRS controlled with decentralized sliding mode controllers was reported in [6], which was able to transport arbitrary shaped objects without predefined trajectories. However, a dynamics model of the individual robot is still needed. Research of a decentralized adaptive control strategy was reported in [24], which enabled cooperative object transportation with two robotic arms. These researchers also investigated model predictive control (MPC) on this task [25]. Although adaptive control compensated for uncertainties and MPC dealt with unsolvable optimalities, the dynamics model and path planning routine cannot be saved. Our colleagues proposed a solution with a fuzzy logic system which saves the complication of dynamics modelling and path planning [26]. However, the control strategy is only validated with point kinematics and point-mass dynamics.

Due to the complicated dynamics in the cooperative object transportation task, researchers began to seek help from DRL. The most related research was introduced in [10] that had two robots transporting an object through a narrow opening managed by a DQN style algorithm [27]. However, this research

relied on a path planning algorithm and the function of the DQN algorithm was only to adjust the robots whenever a pre-planned trajectory is not accessible. In [28], a multi-agent reinforcement learning algorithm was proposed to deal with a hose transportation problem. The proposed algorithm was based on the original Q-learning, thus was not capable of continuous state inputs. A showcase of multiple robots carrying a long rod while preventing it from falling was demonstrated in [9]. This research was highlighted with controlling the MRS by taking continuous input data to resolve continuous output signals. Although the algorithm was more generalized, the goal and constraints were largely different from our study.

III. METHODOLOGY

A. SYSTEM CONFIGURATION

The problem originates from a scenario of multiple persons manipulating a large object. A representative case is two persons moving a long sofa out of the room through a narrow door. We simplify and model the case as two mobile robots linked by a solid rod moving out of a walled cell with an opening. We define the cell in a squared shape with dimensions of $10\,m \times 10\,m$. A global coordinate system $\{X, Y, Z\}$ is fixed to the center of the room, where X axis is pointing to the east, Y axis is pointing to the north and Z axis is determined by the right-hand rule. The opening is located on the south wall which is a doorway with width, $w = 2m$ and depth, $d = 1m$. The rod has length $l = w$. After attached to the robots with radius $r = 0.25m$, the rod cannot be transported out when parallel to the room opening. We restrict the whole system with two linear degrees of freedom (DOF) and one rotational DOF all in the plane of XY. Three body reference frames: $\{^{(1)}x, ^{(1)}y, ^{(1)}z\}$, $\{^{(2)}x, ^{(2)}y, ^{(2)}z\}$ and $\{^{(s)}x, ^{(s)}y, ^{(s)}z\}$ are attached to *robot 1*, *robot 2* and the rod, respectively. The $^{(1)}x$ and $^{(2)}x$ are set to point toward the head of the robots, while $^{(s)}x$ is set along the line between body frames of the robots pointing towards *robot 1*'s origin. $^{(i)}z$ axes are perpendicular to the XY plane with the same direction as the Z axis. Hence, $^{(i)}y$ axes can be determined by $^{(i)}x$, $^{(i)}z$ and the right hand rule. Coordinates of the body frames origin under the global frame define the positions of the robots ($\vec{r}_1$, $\vec{r}_2$) and the rod ($\vec{r}_s$). To better focus on the object transportation problem, we assume the rod is connected to the robots with cylindrical revolute joints, so there is no way for the MRS to drop the rod during transportation. Our settings can be illustrated by Fig. 1 (to save more space, we do not draw east and west wall in full length).

B. DEEP REINFORCEMENT LEARNING CONTEXT

The rod transportation task can be viewed as a series of events. The task starts at time step $t = 0$ and ends at time step $t = T$. Everything happens in between this period consists of an episode. At any time step t, we can describe a robot in the state of $^{(i)}\mathbf{s}_t$, where i is the index of the robot. Assume every robot in this system can perfectly sense the pose and velocity of the rod and all the robots (including itself and teammates), we can define the state of *robot i* as: $^{(i)}\mathbf{s}_t = [^{(i)}\vec{r}_t, ^{(i)}\dot{\vec{r}}_t, ^{(s)}\vec{r}_t, ^{(s)}\dot{\vec{r}}_t, ^{(j)}\vec{r}_t, ^{(j)}\dot{\vec{r}}_t]$, where $\dot{\vec{r}}$'s are velocity vectors of the robots and the rod. Each robot can take an action $^{(i)}\mathbf{a}_t \in \{forwardleft, forwardright, backwardleft, backwardright\}$ at any time step t. Then, the system will transfer to the next step $t+1$, and the state of *robot i* can be obtained as: $^{(i)}\mathbf{s}_{t+1} = [^{(i)}\vec{r}_{t+1}, ^{(i)}\dot{\vec{r}}_{t+1}, ^{(s)}\vec{r}_{t+1}, ^{(s)}\dot{\vec{r}}_{t+1}, ^{(j)}\vec{r}_{t+1}, ^{(j)}\dot{\vec{r}}_{t+1}]$. Together with the new state, the *robot i* receives a reward $^{(i)}\mathbf{r_{t+1}}$ as defined in Eq. 1 which is the key to guarantee the robots can cooperate in this task. An individual will not receive any reward until both of them successfully escaped the room.

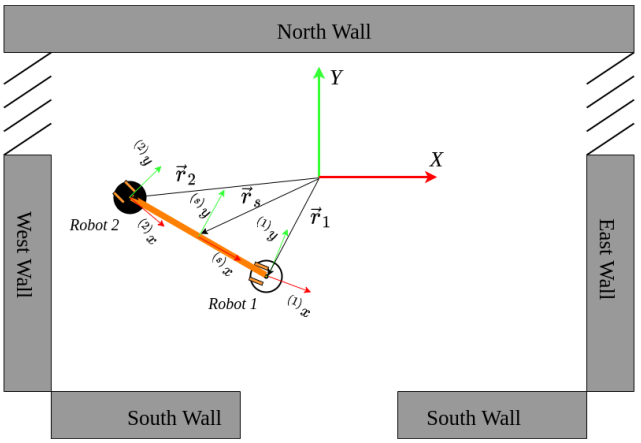


FIGURE 1. Task environment description.

$$^{(i)}\mathbf{r}(^{(i)}\mathbf{s}, ^{(i)}\mathbf{a}) = \begin{cases} 1 & \text{if entire body of the rod is out of the room} \\ 0 & \text{otherwise} \end{cases} \tag{1}$$

However, rewards generated by such a function will be too sparse and the learning process could be largely slowed down due to this effect. In practice we extend Eq. 1 to the form as seen in Eq. 2 to guarantee a non-zero reward can be received at every time step.

$$^{(i)}\mathbf{r}(^{(i)}\mathbf{s}, ^{(i)}\mathbf{a}) = \begin{cases} 400 & \text{if entire body of the rod is out of the room} \\ -100 & \text{if any robot hit wall} \\ -0.1 & \text{otherwise} \end{cases} \tag{2}$$

The dense reward function in Eq. 2 is inspired by *LunarLander* environment from OpenAI Gym [29]. Instead of giving a negative reward according to the scale of control signals, we punish the robots from the perspective of time. Since we limit the horizon of an episode to be 1000 time steps, it is reasonable to give a large negative reward for the event of hitting the wall 1000 times larger than the routine time cost. In our case, if punishment of hitting a wall is larger than -100 (e.g. -1), the robots could become stuck at a local optimal point which will lead them to hit the wall directly. Because they are more likely to receive a smaller negative total reward for hitting the wall compare to struggling too long in the room but accumulating a larger negative reward in the end.

In an episode, the total reward a robot receives by taking an action $\mathbf{a}_t$ at state $\mathbf{s}_t$ can be defined as

$$\mathbf{R}_t = \sum_{k=0}^{T-t} \gamma^k r_{t+k+1}(\mathbf{s}_t, \mathbf{a}_t) \tag{3}$$

where $\gamma \in [0, 1]$ is the discount rate which weighs future rewards less and less because of the nature of uncertainties. We can further describe the value of an action $\mathbf{a}_t$ taken at state $\mathbf{s}_t$ to be the expected total reward: $\mathbf{Q}(\mathbf{s_t}, \mathbf{a_t}) = \mathbb{E}[\mathbf{R}_t]$. We can call this value of state-action pair as **Q value**. Assume the behavior of a robot is determined by a control policy $\pi(\mathbf{a}|\mathbf{s})$, then the Q value of current step can be represented by the Q value of next time step according to the Bellman Expectation Equation as shown in Eq. 4 [7].

$$\mathbf{Q}^{\pi}(\mathbf{s}_t, \mathbf{a}_t) = \mathbf{r}_{t+1} + \gamma \sum\nolimits_{\mathbf{s}_{t+1} \in \mathcal{S}} \mathcal{P}(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t) \times \sum\nolimits_{\mathbf{a}_{t+1} \in \mathcal{A}} \pi(\mathbf{a}_{t+1}|\mathbf{s}_{t+1}) \mathbf{Q}^{\pi}(\mathbf{s}_{t+1}, \mathbf{a}_{t+1}) \tag{4}$$

where $\mathcal{P}$ is the dynamics model (here we use a probability model) which governs the transition between two consecutive time steps. Typically, we need to model the dynamics, $\mathcal{P}$, such that we can make plans for the robots then control it to stick to the plan. However, obtaining the dynamics model becomes more challenging as the task becomes more complex. Reinforcement learning (RL) methods seek to solve the problem bypassing the dynamics model to only focus on optimizing the control policy through trial-and-error style interactions. The interactions have one objective that is maximizing the expected total reward at any given state, thus the dynamics model can be safely ignored. In the MRS cooperative object transportation problem, the dynamics of the system is hard to model. Therefore, we propose to use RL methods to solve such a problem.

C. DQN CONTROLLER FOR MULTI-ROBOT SYSTEM

To succeed in this task, a robot has to employ an optimal control policy, π^* that maximizes the expected total reward at a given state, $\mathbf{s}_t$ by taking an optimal action, $\mathbf{a}_t$ governed by π^*. Then Eq. 4 becomes Eq. 5 according to Bellman Optimality Equation.

$$\mathbf{Q}^{\pi^*}(\mathbf{s}_t, \mathbf{a}_t) = \mathbf{r}_{t+1} + \gamma \sum\nolimits_{\mathbf{s}_{t+1} \in \mathcal{S}} \mathcal{P}(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t) \times \max_{\mathbf{a}_{t+1}} \mathbf{Q}^{\pi^*}(\mathbf{s}_{t+1}, \mathbf{a}_{t+1}) \tag{5}$$

Q-learning provides a straightforward way to iteratively optimize tabularized Q values without considering the dynamics model, $\mathcal{P}$ [27]. The Q values can be updated through Eq. 6 that

$$\mathbf{Q}(\mathbf{s}_t, \mathbf{a}_t) = \mathbf{Q}(\mathbf{s}_t, \mathbf{a}_t) + \alpha(\mathbf{r}_{t+1} + \gamma \max_{\mathbf{a}_{t+1}} \mathbf{Q}(\mathbf{s}_{t+1}, \mathbf{a}_{t+1}) - \mathbf{Q}(\mathbf{s}_t, \mathbf{a}_t)) \tag{6}$$

where α is the learning rate. However, the original Q-learning suffered from the limitation of a discrete state and action space. From the previous section, our robot's state space and action space has 18 and 4 dimensions, respectively. Discretizing such state and action space may result in a giant table that to optimize it could involve a huge amount of computing resources or even be impossible. Hence, we can introduce a function approximator to take continuous states into account and serve the same role as the Q table in Q-learning algorithm. A popular type of function approximator is neural networks (NN), and this is how the Deep Q-network succeeded in the control tasks with image inputs [8]. We adopt NN with trainable weights, $\boldsymbol{\theta}$ to approximate the Q function as the Q-net: $\mathbf{Q}(\mathbf{s}, \mathbf{a}; \boldsymbol{\theta})$. The Q-net needs to approximate the expected total reward as can be computed in Eq. 3. So, we need to define a loss function which can tell the difference between the current Q values and the targeted total rewards as Eq. 7 shows.

$$\mathcal{L}(\boldsymbol{\theta}) = [\mathbf{r}_{t+1} + \gamma \max_{a_{t+1}} \mathbf{Q}(\mathbf{s}_{t+1}, \mathbf{a}_{t+1}; \boldsymbol{\theta}) - \mathbf{Q}(\mathbf{s}_t, \mathbf{a}_t; \boldsymbol{\theta})]^2 \tag{7}$$

Differentiating Eq. 7 with respect to $\boldsymbol{\theta}$, we obtain the gradient $\nabla_{\boldsymbol{\theta}} \mathcal{L}$. Hence we can use stochastic gradient decent to update $\boldsymbol{\theta}$ and optimize the loss function.

The robots in this task are set to be homogeneous, thus only one DQN controller needs to be trained with integration of all the robots' experience. The individuals in the MRS can actually accelerate the training by collecting more data in every time step. A little modification on the original DQN algorithm leads to our multi-robot DQN algorithm for homogeneous MRS as shown in Alg. 1. The robots collect experience based on their own observation, but they share the experience using the same replay buffer. As a result, a single DQN controller needs to be trained with the shared experience data. Once the controller is trained, it can be deployed onto every individual in the MRS. Although the number of robots in this research is limited to two, this algorithm can be scalable to more robots as needed. As opposed to the original DQN algorithm which set the target Q value to be: $y_j = \mathbf{r}_{t+1} + \gamma \arg\max_{\mathbf{a}} \mathbf{Q}(\mathbf{s}_{j+1}, \mathbf{a}; \bar{\boldsymbol{\theta}})$, we implement a Double DQN (DDQN) trick to compute target Q value as: $y_j = \mathbf{r}_{t+1} + \gamma \mathbf{Q}(\mathbf{s}_{j+1}, \arg\max_{\mathbf{a}} \mathbf{Q}(\mathbf{s}_{j+1}, \mathbf{a}; \boldsymbol{\theta}); \bar{\boldsymbol{\theta}})$. So the algorithm can suppress the over-optimistic estimations toward the target Q values [30].

The Alg. 1 describes a distributedly deployed controller, but the training process is still centralized. In fact, by allocating each individual a replay buffer and an optimizer, separately, the training process can be distributed onto individuals in an MRS. Hence, we extend Alg. 1 to a heterogeneous version (Alg. 2) as shown in Appendix A so that an individual in an MRS will be trained by self collected data only. As a homogeneous MRS is just a special case of a heterogeneous MRS, Alg. 2 is adaptive to the homogeneous MRS without any problem.

D. COOPERATION METRICS

As researchers lack tools to quantitatively assess the cooperation among the individuals in an MRS, we propose to introduce the mean absolute error (MAE) of Q-values between any

Algorithm 1 Homogeneous MRS DQN

Require: Initialize replay memory $\mathcal{D}$
Require: Initialize active Q-net with random weights: $\boldsymbol{\theta}$
Require: Initialize target Q-net with identical weights as in the active Q-net: $\bar{\boldsymbol{\theta}} = \boldsymbol{\theta}$
for *episode* $= 1$ **to** M **do**
  Initialize pose of the MRS randomly
  Perform ϵ decay
  for *step* $= 1$ **to** T **do**
    for *robot* $= 1$ **to** N **do**
      Select a random action ${}^{i}\mathbf{a}_t$ with probability ϵ; otherwise, ${}^{(i)}\mathbf{a}_t = \arg\max \mathbf{Q}({}^{(i)}\mathbf{s}_t, {}^{(i)}\mathbf{a}_t; \boldsymbol{\theta})$
    end for
    Execute all ${}^{(i)}\mathbf{a}_t$'s, then observe next states: ${}^{(i)}\mathbf{s}_{t+1}$, receive rewards: ${}^{(i)}\mathbf{r}_{t+1}$.
    Store all transitions $({}^{(i)}\mathbf{s}_t, {}^{(i)}\mathbf{a}_t, {}^{(i)}\mathbf{s}_{t+1}, {}^{(i)}\mathbf{r}_{t+1})$ into $\mathcal{D}$.
    Sample random batch of transitions $(\mathbf{s}_j, \mathbf{a}_j, \mathbf{s}_{j+1}, \mathbf{r}_{j+1})$ from $\mathcal{D}$.
    Set target $y_j = \mathbf{r}_{j+1}$ if episode terminates at $j+1$ otherwise, $y_j = \mathbf{r}_{t+1} + \gamma \mathbf{Q}(\mathbf{s}_{j+1}, \arg\max_{\mathbf{a}} \mathbf{Q}(\mathbf{s}_{j+1}, \mathbf{a}; \boldsymbol{\theta}); \bar{\boldsymbol{\theta}})$
    Perform gradient decent on $\boldsymbol{\theta}$
    Every C steps, reset $\bar{\boldsymbol{\theta}} = \boldsymbol{\theta}$
  end for
end for

two robots to be the standard. The nature of the Q-value is the expected total reward as illustrated by Eq. 4. At time step t, the expected total reward of robot i can be represented by $\max_{\mathbf{a}}({}^{(i)}\mathbf{Q}({}^{(i)}\mathbf{s}_t, {}^{(i)}\mathbf{a}_t))$. The absolute error of Q-values at the same moment t between robot i and robot j, $\mathbf{e}_t = |{}^{(i)}\mathbf{Q}_t - {}^{(j)}\mathbf{Q}_t|$, represents how differently these two robots evaluate their situation at the same moment. Using the MAE of Q-values, we introduce the novel metric ΔQ to assess cooperation between any pair of robots in an episode (Eq. 8).

$$ {}^{(ij)}\Delta Q = \frac{1}{T}\sum_{t=0}^{T} |{}^{(i)}\mathbf{Q}_t - {}^{(j)}\mathbf{Q}_t| \tag{8} $$

We consider the cooperation between two robots having higher quality when ΔQ is smaller. In fact, ΔQ not only allows us to quantify the cooperation in realtime, but it also offers a tool to measure how good the cooperation will be if they were working under pre-planned trajectories. More details can be found in Fig. 3.

E. EXPERIMENT CONFIGURATIONS

Design of the two-robot system is as Fig. 4 illustrates in Appendix B. For each individual robot, five rigid parts with basic geometries are included (chassis, left wheel, right wheel, caster wheel, hat). The rod is attached to the robots through their hats which can be freely rotated with respect to their chassis.

The multibody dynamics is taken care of by the Open Dynamics Engine (ODE) [31], which is integrated in the Gazebo simulation software [32]. To enable interactions between the MRS and the simulated task environment, we first design robot models and an environment model, then add a Python API for the MRS to interact with the environment. Lastly, we can implement out RL algorithms through Tensorflow (an end-to-end open source platform for machine learning) [33]. The architecture of the software interface can be seen in Fig. 5. The details of system modeling, software API construction and training scripts are all open-sourced and can be found at https://github.com/IRASatUC/two_loggers.

The DQN's in this research were all constructed with two fully connected hidden layers with 256 weights in each layer. Hyper-parameters that we applied in the training are listed in Table 4 in Appendix C. The training was performed on a desktop computer with an AMD Threadripper 1900X CPU and an Nvidia GeForce GTX 1080Ti GPU (A GPU is not required for this task).

IV. RESULTS AND ANALYSES

A. TRAINING PERFORMANCE IMPROVEMENT

We employ an averaged total reward as the metric to assess performance improvement during training. Assume the MRS receives total reward $\mathbf{R}_m$ at episode m, then the averaged total reward at this episode can be expressed as

$$ \bar{\mathbf{R}}_m = \frac{1}{m}\sum_{i=1}^{m} \mathbf{R}_i \tag{9} $$

The training procedure of using both homogeneous algorithm and heterogeneous counterparts was recorded and compared in Fig. 2. In addition to the proposed algorithms, we trained a centralized controller using the original DQN algorithm [8] to serve as the baseline which is also presented in Fig. 2. The centralized DQN took the concatenated individuals' states as input and output combinations of actions. The training process was centralized with one optimizer, and eventually deployed a single controller on top of the whole MRS. All these three types of training lasted 30000 episodes. We can notice that the homogeneous DQN training performed slightly better than the heterogeneous one. The decentralized DQN training apparently outperforms the centralized DQN training for they started to succeed earlier and also achieved much higher averaged total reward at the 30000 episode milestone. Considering the centralized DQN controller needs to handle 16 actions (compared to 4 actions for the decentralized setting), it is reasonable that the learning speed slowed down and the training process was less stable. The total amount of interactions(time steps) that happened during the homogeneous DQN training was about 5.1×10^6, the training of heterogeneous DQN took more than 5.8×10^6 interactions, while training of the centralized DQN took more than 8×10^6 interactions to finish 30000 episodes. This result suggests that the decentralized architecture is more efficient than the centralized counterpart. Employing homogeneous robots in an MRS is the most efficient configuration when training decentralized DQN controllers.

B. DQN CONTROLLER PERFORMANCE ANALYSIS

We evaluated the performance of the controllers by running 1000 trials with the MRS randomly initiated in the cell.

Algorithm 2 Heterogeneous MRS DQN Training

for *robot* $= 1$ **to** N **do**
  Initialize replay memory ${}^{(i)}\mathcal{D}$.
  Initialize active Q-net with random weights: ${}^{(i)}\boldsymbol{\theta}$.
  Initialize target Q-net with identical weights as in the active Q-net: ${}^{(i)}\bar{\boldsymbol{\theta}} = {}^{(i)}\boldsymbol{\theta}$.
end for
for *episode* $= 1$ **to** M **do**
  Initialize pose of the MRS randomly.
  Perform ϵ decay.
  for *step* $= 1$ **to** T **do**
    for *robot* $= 1$ **to** N **do**
      Select a random action ${}^{i}\mathbf{a}_t$ with probability ϵ;
      otherwise, ${}^{(i)}\mathbf{a}_t = \arg\max \mathbf{Q}({}^{(i)}\mathbf{s}_t, {}^{(i)}\mathbf{a}_t; \boldsymbol{\theta})$.
    end for
    Execute all ${}^{(i)}\mathbf{a}_t$'s, then observe next states: ${}^{(i)}\mathbf{s}_{t+1}$, receive rewards: ${}^{(i)}\mathbf{r}_{t+1}$.
    for *robot* $= 1$ **to** N **do**
      Store transition $({}^{(i)}\mathbf{s}_t, {}^{(i)}\mathbf{a}_t, {}^{(i)}\mathbf{s}_{t+1}, {}^{(i)}\mathbf{r}_{t+1})$ into ${}^{(i)}\mathcal{D}$.
      Sample random batch of transitions $({}^{(i)}\mathbf{s}_j, {}^{(i)}\mathbf{a}_j, {}^{(i)}\mathbf{s}_{j+1}, {}^{(i)}\mathbf{r}_{j+1})$ from ${}^{(i)}\mathcal{D}$.
      Set target ${}^{(i)}y_j = {}^{(i)}\mathbf{r}_{j+1}$ if episode terminates at $j+1$.
      otherwise,
      ${}^{(i)}y_j = {}^{(i)}\mathbf{r}_{t+1} + \gamma\, {}^{(i)}\mathbf{Q}({}^{(i)}\mathbf{s}_{j+1}, \arg\max_{\mathbf{a}}^{(i)} \mathbf{Q}({}^{(i)}\mathbf{s}_{j+1}, {}^{(i)}\mathbf{a}; {}^{(i)}\boldsymbol{\theta}); {}^{(i)}\bar{\boldsymbol{\theta}})$.
      Perform gradient decent on ${}^{(i)}\boldsymbol{\theta}$.
      Every C steps, reset ${}^{(i)}\bar{\boldsymbol{\theta}} = {}^{(i)}\boldsymbol{\theta}$.
    end for
  end for
end for

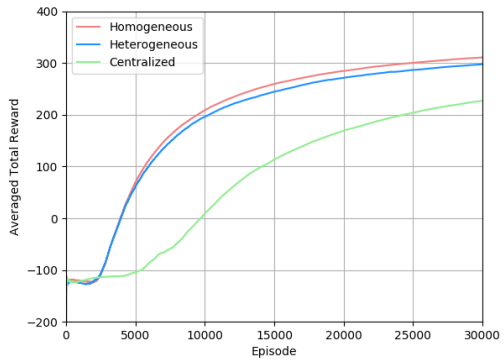


FIGURE 2. Averaged total reward growth along training episodes.

The homogeneous training algorithm resulted in a MRS with 0.966 success rate, the heterogeneous training algorithm produced a MRS with 0.955 success rate, and the centralized DQN gave out a 0.941 success rate. We also tested the performance of the DQN controllers against uncertainties. By increasingly adding Gaussian noise to the states (inputs) of the controllers, we found the success rate of both MRSs degraded gradually. Table 1 shows the performance drop of the MRS against the increased noise level. Considering the success rate of an untrained controller (analogous to taking random actions all the time) barely reached 0.001, the DQN controllers demonstrated reasonable robustness against input uncertainties. Especially when the noise level was controlled under $\mathcal{N}(0, 0.1)$,

TABLE 1. Effect of state noise.

Noise Level	Homogeneous	Heterogeneous	Centralized
w/o noise	0.966	0.955	0.941
$\mathcal{N}(0, 0.1)$	0.974	0.969	0.933
$\mathcal{N}(0, 0.2)$	0.926	0.952	0.896
$\mathcal{N}(0, 0.3)$	0.819	0.867	0.794
$\mathcal{N}(0, 0.4)$	0.696	0.725	0.670
$\mathcal{N}(0, 0.5)$	0.546	0.553	0.482

TABLE 2. Effect of action randomness.

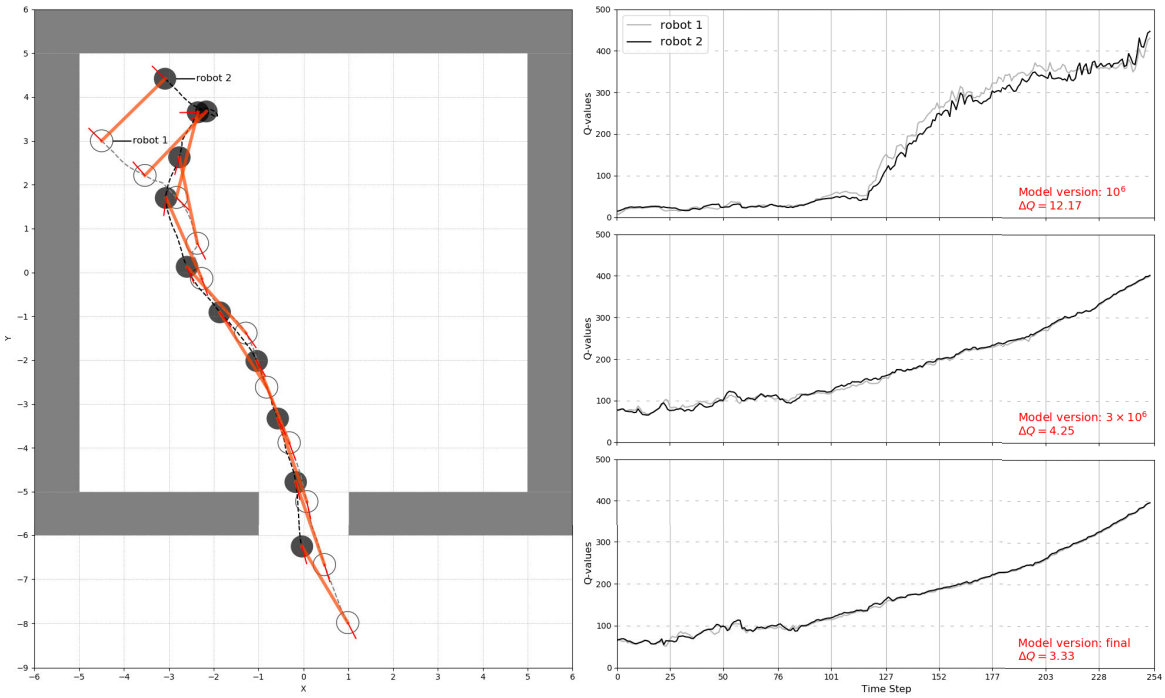
Random Level	Homogeneous	Heterogeneous	Centralized
w/o randomness	0.966	0.955	0.941
10%	0.961	0.969	0.938
20%	0.970	0.967	0.924
30%	0.952	0.940	0.925
40%	0.938	0.919	0.889
50%	0.882	0.884	0.832

TABLE 3. Metric of cooperation.

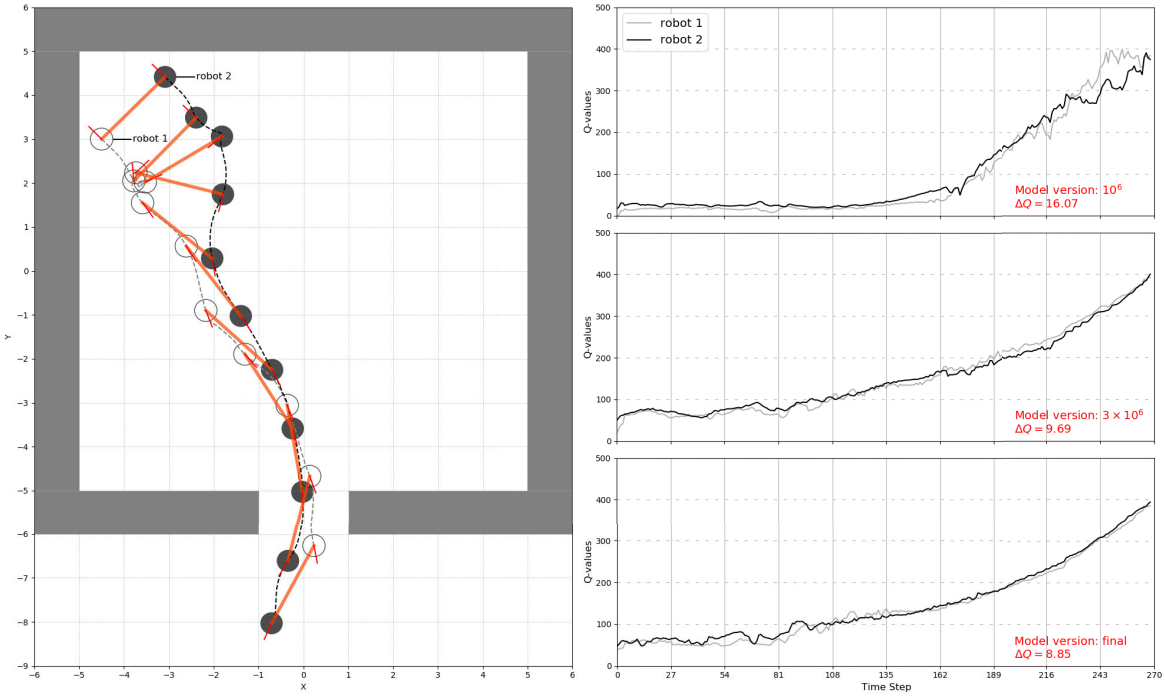
Model version	Homogeneous	Heterogeneous
10^6	11.72 ± 4.55	19.39 ± 9.70
final	$\mathbf{3.77 \pm 2.20}$	7.83 ± 3.06

the performance of decentralized DQN were not affected at all.

The uncertainties can happen on the output end of the controllers, thus we also tested the controllers' tolerance on the randomness of control signals. The results can be seen in Table 2 that the performance degradation was slowly building up along the gradually increased output randomness. Even with a 50% chance that a controller will take a



(a) Homogeneous DQN algorithm trained case



(b) Heterogeneous DQN algorithm trained case

FIGURE 3. **Sampled trajectories and cooperation assessment.**

random action, all the controllers still had success rate over 0.8. When output randomness was under 10%, there was no performance drop observed in all three categories. When output randomness increased to 20%, the decentralized DQN controllers can still maintain their performance, whereas the performance of the centralized controller began to degrade.

C. QUANTITATIVE COOPERATION ASSESSMENT

As a highlight of this research, we proposed a metric, ΔQ (described in III-D), to assess the cooperation between two individuals in an MRS. Table. 3 shows the result from 1000 test trials with MRS randomly initiated in the cell. Under the assessment of ΔQ, controllers trained by the homogeneous DQN algorithm demonstrated the best cooperation

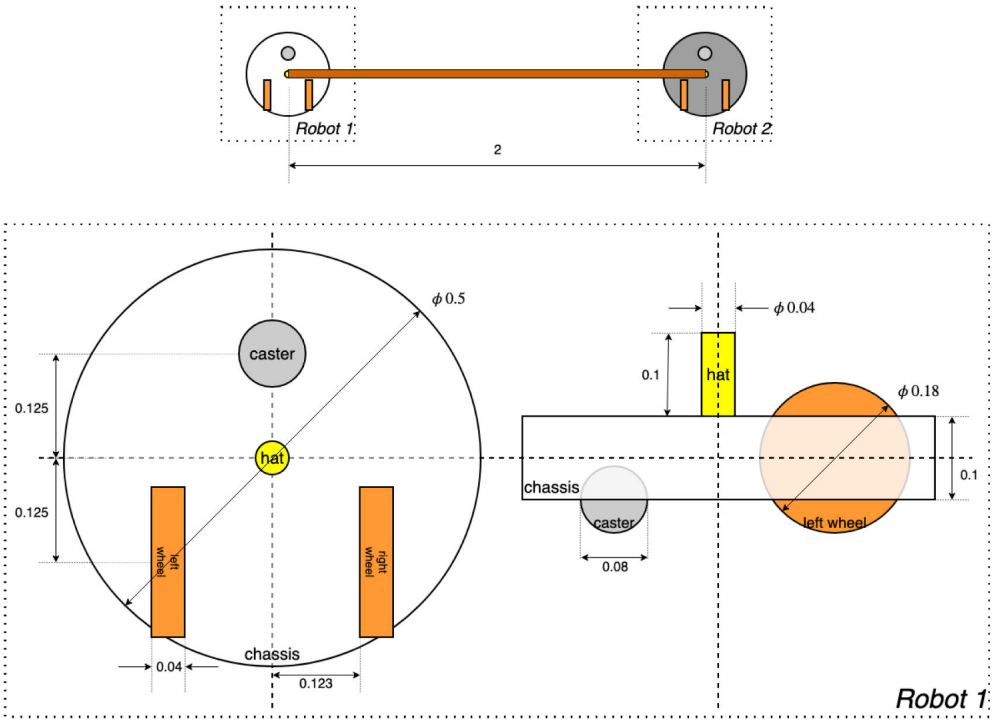


FIGURE 4. Dimensions of the two-robot system.

with minimum difference when estimating the total reward at the same moment. The two controllers had larger disagreement if trained by the heterogeneous DQN algorithm. During training, we saved DQN models every 10^6 interactions, so that the changed behaviors of the robots can always be tracked by loading previously saved models. We also compared metric ΔQ at different stages of training. Both algorithms exhibited larger ΔQs when trained with 10^6 interactions data. Both algorithms showed a trend of decreasing ΔQ with increasing training episodes, which indicates both algorithms can shape the value systems in-between the controllers to a more unified form.

A case study is given here for a better understanding of the cooperation metric, ΔQ. We sampled two trajectories of transportation using the final models trained by the homogeneous DQN algorithm and the heterogeneous one, respectively. We can see these two trajectories on the left hand side of Fig. 3a and 3b. The homogeneous DQN algorithm trained controllers took more time (219 time steps) to solve the problem. While the heterogeneous DQN algorithm trained controllers took fewer time (205 time steps). After the trajectories were sampled, we can load different versions of the DQN models to evaluate the same trajectory. On the right hand side of Fig. 3, we evaluate the cooperation between these two robots using DQN models from different stages of training. The graphs on the top, indicate cooperation evaluation using controllers trained by one million interactions. Similarly, the graphs in the middle employed controllers trained by 3 million interactions and the bottom ones used the final states of the controllers. Regardless of whether the controllers were trained with the homogeneous algorithm or

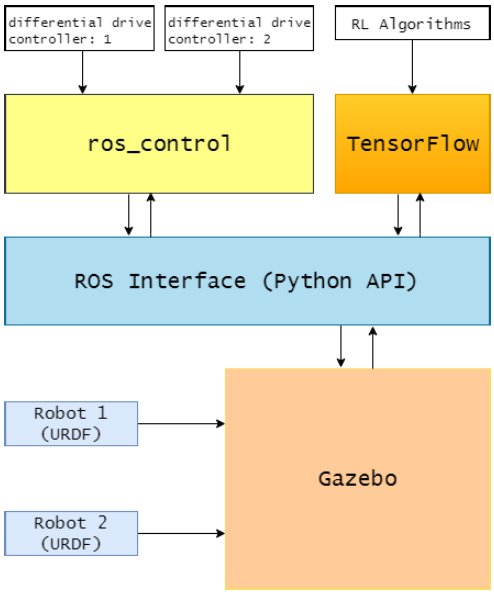


FIGURE 5. Simulation layout.

the heterogeneous one, the disagreement between the robots is hardly noticeable. In general, more obvious Q-value deviations can be observed on the heterogeneous DQN algorithm side. For both homogeneous and heterogeneous DQN algorithms, models trained with 3×10^6 interactions resulted in similar Q-values as the final models did when evaluating the same trajectories. The cooperation metric ΔQs were relatively small when using these two versions of the DQN model. Models trained with 10^6 interactions, however, tended to underrate the values of first 70% trajectories.

TABLE 4. DQN hyper-parameters.

Hyper-parameter	Value	Description
batch size	8192	Number of experiences sampled for one step of gradient decent optimization.
replay buffer size	10000000	Maximum experiences can be stored in replay buffer.
update frequency	8000	Target network will be updated to comply to the active Q-network at this frequency.
discount rate	0.99	Used on future rewards to calculate the expected total reward.
learning rate	0.0001	Used to control update step of the trainable weights in Q-networks.
initial exploration	1	Initial value of action randomness.
final exploration	0.1	Action randomness after ε stopped decaying.
ε decay period	2000	Number of episodes needed for ε decaying from initial exploration to the final exploration
warm-up episodes	500	Number of episodes run with completely random actions before Q-networks were optimized.

TABLE 5. Case-wise metrics.

Case Index	Time Consumed			Distance Traveled			ΔQ	
	homo.	hete.	cent.	homo.	hete.	cent.	homo.	hete.
1	**253**	269	381	**25.62**	25.71	32.13	**3.33**	8.85
2	233	285	**230**	**24.42**	27.28	24.97	**2.54**	6.82
3	265	**252**	**252**	**25.38**	25.62	26.35	**3.01**	9.04
4	**174**	233	196	**18.93**	22.61	20.38	**3.59**	6.41
5	178	186	**177**	18.92	**18.91**	19.12	**2.32**	9.86
6	161	155	**151**	15.24	**15.16**	15.22	**3.34**	8.68
7	147	**141**	147	14.19	**12.94**	14.89	**5.96**	21.90
8	**147**	199	237	**15.30**	15.76	18.35	**3.08**	9.79
9	**180**	N/A	N/A	**10.88**	N/A	N/A	**10.11**	15.85

It is reasonable since the value was slowly propagated from the exit to further locations. More than that, larger ΔQ can be observed in the results of the version using 10^6 models.

D. OTHER FINDINGS AND DISCUSSIONS

We sampled the MRS's performance from 9 different initial conditions. Metrics of *time consumed*, *travel distance* and ΔQ were extracted from the case-wise experiments which can be found in Table 5. We compared performance of controllers trained by homogeneous, heterogeneous and centralized DQN algorithms. While homogeneous DQN algorithm trained controllers having significantly smaller gaps when evaluating states at the same moment, they were slightly outperformed heterogeneous DQN and centralized DQN trained controllers in the manners of *time consumed* and *distance traveled*. Trajectories of these samples were recorded as Fig. 6 illustrated in Appendix D. An obvious that pattern can be observed is that the robot initially closer to the exit is more likely to be the first to exit, which indicates that the DRL trained MRS is flexible enough to adjust the individuals according to specific situations.

Researchers have proposed many advanced DRL algorithms since the birth of DQN. A major draw back of DQN is it cannot deal with continuous action space. Deep deterministic policy gradient (DDPG) is one of the successors of DQN, which introduced a policy network to produce continuous actions [34]. Proximal policy optimization (PPO) is the one from the family of on-policy RL algorithms, which can optimize continuous policy through guaranteed improvement without stepping too far to avoid collapsed performance [35]. We have tested both algorithms in the presented task, but neither of them induced positive results so far. DDPG's convergence was extremely brittle during training, while both DDPG and PPO had hard time balancing between exploration and exploitation.

V. CONCLUSION

In this research, we introduced decentralized deep Q-network (DQN) controllers in a multi-robot system (MRS) to solve an object transportation task cooperatively. The controllers learned how to behave and cooperate from scratch. Given the state of the MRS, the DQN controllers can output discretized control signals directly without knowing the dynamics nor planning a path. Two training algorithms were proposed to adapt homogeneous and heterogeneous MRS, respectively. Both algorithms were able to train well-performed DQN controllers on homogeneous robots to solve the task with a high success rate. The decentralized architecture was proven to be more efficient than the centralized counterpart. The DQN controllers were proven to be robust against small to medium level uncertainties. More importantly, a novel and universal metric was proposed in this research that can quantitatively assess cooperation between robots in an MRS. With the encouraging results, we are now more confident in the potential of deep reinforcement learning (DRL) type controllers in MRSs.

In the future, we would like to continue exploring the potential of MRS's with more advanced DRL algorithms and settings. Although DQN as an off-policy DRL algorithm is theoretically more data-efficient compared to the on-policy algorithms, the training process in this research still took more than 5×10^6 interactions over 10 days. We look forward to improving the efficiency of training, as this is essential to bring a DRL-type controller to a robot in the real world. The solution proposed is only feasible to the introduced scenario since the input states only contained the pose of the MRS. Due to the ignorance of other environmental features

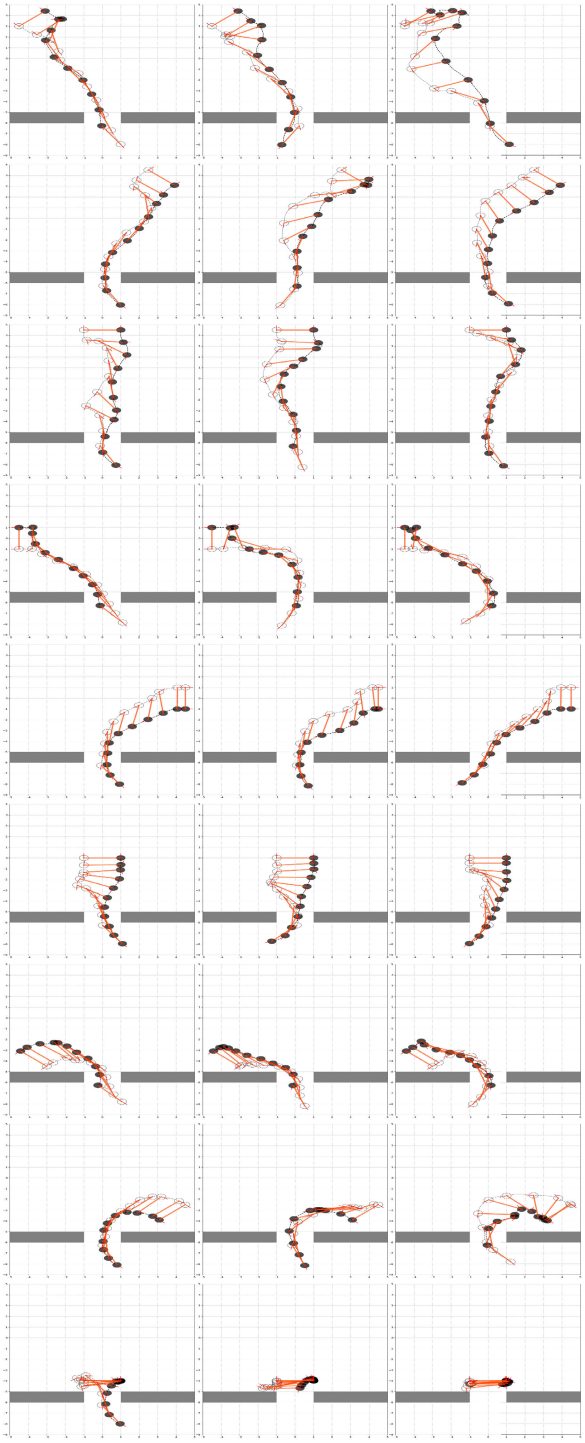


FIGURE 6. Cooperative transportation cases (left: homogeneous; middle: heterogeneous; right: centralized).

(e.g. distance to wall), the controller cannot be generalized to a largely varied environment. Hence, in the future, we look forward to involving low-level input signals (e.g. laser scans, images, etc.) to engage the MRS in a more generalized world. Despite the fact that we introduced a heterogeneous DQN algorithm, the training actually occurred on homogeneous instances. We are planning new experiments to test how exactly this algorithm would perform on a heterogeneous MRS. The sensing data in the current research is pulled out directly from the simulation software, which is not likely to be accessible in real applications. More than that, current sensing data cannot deal with constantly changing obstacles (e.g. the doorway is randomly placed). Hence, we are preparing to upgrade the current robots to be equipped with cameras and Lidars to adapt to upgraded task environment with more dynamic objects.

APPENDIX A HETEROGENEOUS MRS TRAINING ALGORITHM

Unlike the homogeneous counterpart, the training of heterogeneous MRS happens separately on each individual. Instead of sharing the data collected by all the members in the team, each robotic controller will be trained with data collected by itself.

APPENDIX B ROBOT DIMENSIONS

The mechanical design and robots dimensions are shown in Fig. 4

Software layout and interfaces are shown in Fig. 5

APPENDIX C DQN HYPER-PARAMETERS

Hyper-parameters used in DQN training are shown in Table 4

APPENDIX D CASE STUDY

We sampled 9 cases of transportations with specified initial conditions. The trajectories of the transportations using different controllers can be seen in Fig. 6. Key metrics including total travel distance, time consumed and cooperation metrics: ΔQ were recorded. Since heterogeneous and centralized DQN trained controllers failed in the last case, the assessment on these two were not applicable.

REFERENCES

[1] Wikipedia Contributors. (2020). *STS-49—Wikipedia, The Free Encyclopedia*. Accessed: Oct. 6, 2020. [Online]. Available: https://en.wikipedia.org/w/index.php?title=STS-49&oldid=982151904

[2] E. Tuci, M. H. M. Alkilabi, and O. Akanyeti, "Cooperative object transport in multi-robot systems: A review of the State-of-the-Art," *Frontiers Robot. AI*, vol. 5, p. 59, May 2018.

[3] Z. H. Ismail and N. Sariff, "A survey and analysis of cooperative multi-agent robot systems: Challenges and directions," in *Applications of Mobile Robots*. Rijeka, Croatia: IntechOpen, 2018.

[4] K. Kosuge and T. Oosumi, "Decentralized control of multiple robots handling an object," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Nov. 1996, pp. 318–323.

[5] Z. Wang and M. Schwager, "Kinematic multi-robot manipulation with no communication using force feedback," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2016, pp. 427–432.

[6] H. Farivarnejad, S. Wilson, and S. Berman, "Decentralized sliding mode control for autonomous collective transport by multi-robot systems," in *Proc. IEEE 55th Conf. Decis. Control (CDC)*, Dec. 2016, pp. 1826–1833.

[7] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. Cambridge, MA, USA: MIT Press, 2018.

[8] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, Feb. 2015.

[9] J. K. Gupta, M. Egorov, and M. Kochenderfer, "Cooperative multi-agent control using deep reinforcement learning," in *Proc. Int. Conf. Auton. Agents Multiagent Syst.* Cham, Switzerland: Springer, 2017, pp. 66–83.

[10] S. V. Manko, S. A. K. Diane, A. E. Krivoshatskiy, I. D. Margolin, and E. A. Slepynina, "Adaptive control of a multi-robot system for transportation of large-sized objects based on reinforcement learning," in *Proc. IEEE Conf. Russian Young Researchers Electr. Electron. Eng. (EIConRus)*, Jan. 2018, pp. 923–927.

[11] P. Hernandez-Leal, B. Kartal, and M. E. Taylor, "Agent modeling as auxiliary task for deep reinforcement learning," in *Proc. AAAI Conference Artif. Intell. Interact. Digit. Entertainment*, 2019, vol. 15, no. 1, pp. 31–37.

[12] C. R. Kube and H. Zhang, "Collective robotics: From social insects to robots," *Adapt. Behav.*, vol. 2, no. 2, pp. 189–218, Sep. 1993.

[13] Z.-D. Wang, E. Nakano, and T. Matsukawa, "Cooperating multiple behavior-based robots for object manipulation," in *Distributed Autonomous Robotic Systems*. Tokyo, Japan: Springer, 1994, pp. 371–382.

[14] S. Sen, M. Sekaran, and J. Hale, "Learning to coordinate without sharing information," in *Proc. AAAI*, vol. 94, 1994, pp. 426–431.

[15] R. G. Brown and J. S. Jennings, "A pusher/steerer model for strongly cooperative mobile robot manipulation," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. Human Robot Interact. Cooperat. Robots*, Aug. 1995, pp. 562–568.

[16] B. Hichri, L. Adouane, J.-C. Fauroux, Y. Mezouar, and I. Doroftei, "Cooperative mobile robot control architecture for lifting and transportation of any shape payload," in *Distributed Autonomous Robotic Systems*. Tokyo, Japan: Springer, 2016, pp. 177–191.

[17] T. Machado, T. Malheiro, S. Monteiro, W. Erlhagen, and E. Bicho, "Multi-constrained joint transportation tasks by teams of autonomous mobile robots using a dynamical systems approach," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2016, pp. 3111–3117.

[18] C. P. Bechlioulis and K. J. Kyriakopoulos, "Collaborative multi-robot transportation in obstacle-cluttered environments via implicit communication," *Frontiers Robot. AI*, vol. 5, p. 90, Aug. 2018.

[19] C.-H. Lin, S.-H. Wang, and C.-J. Lin, "Interval Type-2 neural fuzzy controller-based navigation of cooperative load-carrying mobile robots in unknown environments," *Sensors*, vol. 18, no. 12, p. 4181, Nov. 2018.

[20] O. Medina, S. Hacohen, and N. Shvalb, "Robotic swarm motion planning for load carrying and manipulating," *IEEE Access*, vol. 8, pp. 53141–53150, 2020.

[21] J. Alonso-Mora, S. Baker, and D. Rus, "Multi-robot formation control and object transport in dynamic environments via constrained optimization," *Int. J. Robot. Res.*, vol. 36, no. 9, pp. 1000–1021, Aug. 2017.

[22] J. Alonso-Mora, E. Montijano, T. Nägeli, O. Hilliges, M. Schwager, and D. Rus, "Distributed multi-robot formation control in dynamic environments," *Auto. Robots*, vol. 43, no. 5, pp. 1079–1100, Jun. 2019.

[23] A.-N. Ponce-Hinestroza, J.-A. Castro-Castro, H.-I. Guerrero-Reyes, V. Parra-Vega, and E. Olguin-Diaz, "Cooperative redundant omnidirectional mobile manipulators: Model-free decentralized integral sliding modes and passive velocity fields," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2016, pp. 2375–2380.

[24] C. K. Verginis, M. Mastellaro, and D. V. Dimarogonas, "Robust cooperative manipulation without Force/Torque measurements: Control design and experiments," *IEEE Trans. Control Syst. Technol.*, vol. 28, no. 3, pp. 713–729, May 2020.

[25] A. Nikou, C. Verginis, S. Heshmati-alamdari, and D. V. Dimarogonas, "A nonlinear model predictive control scheme for cooperative manipulation with singularity and collision avoidance," in *Proc. 25th Medit. Conf. Control Autom. (MED)*, Jul. 2017, pp. 707–712.

[26] Y. Sun, A. Barth, and O. Ma, "An intelligent approach for a two-robot team to perform a cooperative task," in *Proc. AIAA Scitech Forum*, Jan. 2020, p. 1116.

[27] C. J. C. H. Watkins and P. Dayan, "Q-learning," *Mach. Learn.*, vol. 8, nos. 3–4, pp. 279–292, 1992.

[28] B. Fernandez-Gauna, I. Etxeberria-Agiriano, and M. Graña, "Learning multirobot hose transportation and deployment by distributed round-robin Q-Learning," *PLoS ONE*, vol. 10, no. 7, Jul. 2015, Art. no. e0127129.

[29] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba, "OpenAI gym," 2016, *arXiv:1606.01540*. [Online]. Available: http://arxiv.org/abs/1606.01540

[30] H. Van Hasselt, A. Guez, and D. Silver, "Deep reinforcement learning with double q-learning," in *Proc. 13th AAAI Conf. Artif. Intell.*, 2016, pp. 2094–2100.

[31] E. Drumwright, J. Hsu, N. Koenig, and D. Shell, "Extending open dynamics engine for robotics simulation," in *Proc. Int. Conf. Simulation, Modeling, Program. Auton. Robots*. Berlin, Germany: Springer, 2010, pp. 38–50.

[32] N. Koenig and A. Howard, "Design and use paradigms for gazebo, an open-source multi-robot simulator," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, vol. 3, Sep./Oct. 2004, pp. 2149–2154.

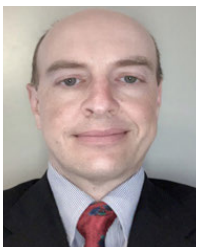
[33] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, and M. Isard, "Tensorflow: A system for large-scale machine learning," in *Proc. 12th USENIX Symp. Operating Syst. Design Implement. (OSDI)*, 2016, pp. 265–283.

[34] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," 2015, *arXiv:1509.02971*. [Online]. Available: http://arxiv.org/abs/1509.02971

[35] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," 2017, *arXiv:1707.06347*. [Online]. Available: http://arxiv.org/abs/1707.06347

LIN ZHANG received the B.S. degree in automation from the Harbin Institute of Technology, Heilongjiang, China, in 2007, and the Ph.D. degree in engineering from New Mexico State University, Las Cruces, NM, USA, in 2016. He is currently a Senior Research Associate with the Intelligent Robotics and Autonomous System Laboratory, University of Cincinnati. His major research interests include deep reinforcement learning, robotics, and intelligent systems.

YUFENG SUN received the M.S. degree in mechanical engineering from the Huazhong University of Science and Technology, Hubei, China, in 2007. He is currently pursuing the Ph.D. degree with the University of Cincinnati. From 2007 to 2018, he was a Software Engineer with Autodesk Inc., working on CAD software design and development. He is also a Research Assistant with the University of Cincinnati. His research interests include robotics, computer vision, and intelligent control.



ANDREW BARTH (Member, IEEE) received the B.S. degree in aerospace engineering from the University of Cincinnati, in 1999, where he is currently pursuing the Ph.D. degree. He is also a Research Assistant with the University of Cincinnati. He had 15 years of experience in the aerospace industry providing engineering support to NASA on the International Space Station, Hubble Robotic Vehicle, and Orion programs. He was named Certified Principle Engineer for the re-entry guidance performance for the Orion Exploration Flight Test 1 spacecraft. His research interests include guidance, navigation, control design, and intelligent systems.

OU MA received the B.S. degree in mechanical engineering from Zhejiang University, in 1982, and the Ph.D. degree from McGill University, in 1991. From 1991 to 2002, he was a Senior Project Engineer and the Research and Development Lead with MDA Space Systems (MD Robotics). From 2002 to 2017, he was an Associate Professor, a Professor, and a John Nakayama and Tome Nakayama Professor with New Mexico State University. Since 2017, he has been the Alan B Shepard Chair Professor with the Department of Aerospace Engineering and Engineering Mechanics, University of Cincinnati. His research interests include multibody dynamics and control, impact-contact dynamics, intelligent control of robotic and autonomous systems, human–robot interaction and collaboration, and smart manufacturing.