Active collaboration in relative observation for multi-agent visual simultaneous localization and mapping based on Deep Q Network

Zhaoyi Pei, Songhao Piao, Meixiang Quan, Muhammad Zuhair Qadir and Guo Li

Abstract
This article proposes a unique active relative localization mechanism for multi-agent simultaneous localization and mapping, in which an agent to be observed is considered as a task, and the others who want to assist that agent will perform that task by relative observation. A task allocation algorithm based on deep reinforcement learning is proposed for this mechanism. Each agent can choose whether to localize other agents or to continue independent simultaneous localization and mapping on its own initiative. By this way, the process of each agent simultaneous localization and mapping will be interacted by the collaboration. Firstly, a unique observation function which models the whole multi-agent system is obtained based on ORBSLAM. Secondly, a novel type of Deep Q Network called multi-agent system Deep Q Network (MAS-DQN) is deployed to learn correspondence between Q value and state-action pair, abstract representation of agents in multi-agent system is learned in the process of collaboration among agents. Finally, each agent must act with a certain degree of freedom according to MAS-DQN. The simulation results of comparative experiments prove that this mechanism improves the efficiency of cooperation in the process of multi-agent simultaneous localization and mapping.

Keywords
MAS, DQN, SLAM, relative localization

Date received: 20 May 2019; accepted: 15 February 2020

Introduction
A critical task of a mobile robot is to determine its pose (position and orientation) under the given environment map, which is also the basis of other type of tasks. However, the environment map does not exist at the beginning. When the mobile robots enter an unknown part of environment, they need to construct the map (3-D point cloud map, topological map, 2-D map) to satisfy their tasks through its own sensors, and at the same time determine their own localization in the map, this is called simultaneous localization and mapping (SLAM) problem. Mapping and localization are mutually complementary. In order to build accurate maps, it is necessary for the robot to have an accurate estimation of its pose, vice versa, localization also requires established high-quality maps, it is the difficulty of SLAM. Visual SLAM refers to the SLAM that only the
camera acts as the only visual sensor. Compared with the traditional laser, the visual sensors can obtain more abundant information and can be deployed on mobile robots at very low cost.

Multi-agent system (MAS) refers to that agents in one system collaborate and negotiate with others to accomplish goals that cannot be accomplished by a single agent or to increase the efficiency of task execution. Each agent’s ability, attributes, and structure are different, which provides a great space for collaboration. MAS has attracted many researchers to investigate the applicability of it in many pertinent areas closely related to human livelihood and industry such as security system, exploration and rescue, surveillance, humanitarian assistance, environment protection, and health care. Traditional SLAM has been considered as an independent behavior. With the continuous application of MAS, multi-agent SLAM has become the focus of scholars’ research.

Multi-agent SLAM can be divided into two types: offline mode and online mode. Off-line mode means that each agent’s SLAM process is independent and the results are fused together after all agents stop. Online mode refers to that agents can cooperate with each other in SLAM process, such as cooperative loop closure detection and cooperative localization, cooperative map fusion, and so on. Relative orientations in MAS are important on account of that it is a basis for combining maps built with data from two agents. The difficulty of online multi-agent SLAM is how to cooperate in the best way under the condition that each agent has different attributes with different states. This issue is particularly prominent in the relative localization between robots. For instance, the localization ability of wheeled robot is stronger than that of unmanned aerial vehicle (UAV) because of the odometer, but the pose of UAV is more abundant, which leads to its strong mapping ability. If there are these two kinds of robots in the system at the same time, an appropriate collaboration model will make full use of the advantages of both. This type of relative localization is controlled manually. If MAS wants to optimize the efficiency of automatic cooperative positioning between agents, each agent needs to perceive all the states and their attributes related to SLAM to determine whether it has the optimal condition to observe the agent in need of assistance or not. But in most cases, the task-related attributes and states of robots are not quantifiable which brings difficulties to collaboration.

In order to solve the above problem of SLAM of agents in MAS, this article proposes a cooperative mechanism applied to multi-agent SLAM, which can learn the main attributes related to each agent and extracts the state feature vectors of all agents in MAS based on ORBSLAM. Each agent will determine its specific behavior outputted by MAS-DQN. In the “Experiment” section, we use model of isomorphic robots with different attributes to simulate various robots, and the validity of our algorithm is proved.

The article is organized as follows: in the second section, related work about coalition formation for MAS and multi-agent SLAM is discussed. In the third section, the DQN and “Rainbow” which we applied to drive MAS-DQN are introduced. In the fourth section, the structure of MAS-DQN including the novel observation function and reward function is described. In the fifth section, we introduce relative observation between agents based on the proposed mechanism. In the sixth section, the effectiveness of the algorithm is discussed. In the seventh section, the complete algorithm of the approach is shown.

Related work

The relative localization among agents in MAS refers to the location of an unknown agent whose pose is unknown using others limited knowledge and ability. This process is mainly composed of two steps: first, select the agents that can provide location-based services. These agents can be determined by a general control center or freely negotiated by the agents according to their own demands, interests, and attributes. This step mainly involves multi-agent task allocation (MRTA). In the second step, the selected agent uses its own pose and different types of map such as graph-based map, landmarks with covariance matrix, and grid-based map to measure or observe the pose of the agent to be localized. Next, we will introduce the related work of these two points separately.

MRTA has always been a hot topic in MAS research. Task allocation in MAS (MRTA) can be formulated as an optimal assignment problem where the goal is to optimally assign a set of agents to a set of tasks in such a way that optimizes the overall system performance under a condition of satisfying a set of constraints. According to team organizational paradigm, MRTA can be divided into two categories: centralized approaches and distributed approaches.

There is a central agent that monitors and assigns tasks to other agents in a centralized MAS. Agents who are assigned tasks need to send all the information they obtain to that central agent. They need to keep communication with each other at all times. The center-based algorithm is very general for solving the MRTA problem as a result that monitors can directly control each agent. Colin and Veloso proposed centralized approaches for mobile robot task allocation in hybrid wireless sensor networks. In the literature, a centralized multi-robot task allocation for industrial plant inspection by using A* and genetic algorithm is introduced. Higuera and Dudek proposed a fair division-based MRTA approach as another centralized algorithm to handle the problem of allocating a single global task among heterogeneous robots. In order to solve the problem of curse of dimension caused by multi-agent reinforcement learning, Wadhwania et al. got inspiration from distillation and value-matching and proposed a new actor-
In configuration of distributed approaches, agents do not need to report to a central agent, and they do not need to be online at all times. They can freely communicate with other agents and assign tasks through negotiation with different demands. This type of algorithm is also widely used to solve MRTA problem. Farinelli et al. proposed a distributed online dynamic task assignment for multi-robot patrolling. Luo et al. proposed distributed algorithm for constrained multi-robot task assignment for grouped tasks. Various ambient assisted living technologies have been proposed for MRTA in Healthcare Facilities. Emotional robots model use artificial emotions to endow the robot with emotional factors, which makes the robot more intelligent and adjust its behavior choice through the emotional mechanism. The introduction of emotional and personal factors improves the diversity and autonomy of robots.

Jonathan How’s team focused on using consensus-based auction algorithms to solve decentralized task assignments and utilized game theory to prove that the consensus-based bundle algorithm converges to a pure strategy Nash equilibrium. Johnson et al. used implicit coordination to satisfy allocation constraints.

Some relative localization algorithms for MAS use some prior information such as initial positions of the agents and exact positions of landmarks. Shames et al. presented the theoretical basis of relative localization between agents by using prior knowledge in maps. Zhou and Roumeliotis proposed a mechanism for relative localization in which agents try to observe each other in-time if they have a chance with unknown initial correspondence. Zkucur and Levent establishes the relationship between relative observation and map fusion. However, in most cases, agents cannot get enough prior knowledge for relative location. Some statistical estimation algorithms, such as Kalman filtering, particle filtering, Markov localization perform better in this case.

Most of the relative observation approaches serve for map fusion. In the mechanism proposed by this article, the data from relative observations between agents are also applied for optimizing the pose graph of the observed agents which improves the efficiency of data utilization.

**DQN and rainbow**

Q-learning is an off-policy learning method in reinforcement learning. It uses Q-table to store the cumulative value of each state–action pair. According to Bellman equation, when the strategy of maximizing $Q$ value is adopted in every step, the $Q(s_t, a_t)$ can be calculated as equation (1)

$$Q(s_t, a_t) = r_t + \gamma \max_{a'} Q(s_{t+1}, a')$$

where $Q(s_t, a_t)$ refers to the cumulative discounted reward when agent’s action is $a_t$ at state $s_t$. $s_t$ is the state on time $t$ and $s_{t+1}$ is the state on time $t + 1$. $a_t$ is the action adapted by agent at time $t$. The state of the system is transformed to $s_{t+1}$ from $s_t$ for the action $a_t$ taken by agent. $r_t$ is the immediate reward for the transition from $s_t$ to $s_{t+1}$. $\gamma$ is discounted factor. When the state and action space are high-dimensional or continuous, the maintenance of Q-table $Q(s, a)$ will become unrealistic. Therefore, with the problem of updating Q-table transformed into a problem of fitting function, the basic idea of DQN is to use the neural network to make the Q function $Q(s, a)$ approach the optimal Q value by updating the parameter $\theta$ which refers to all weights of the deep neural network. The specific sequence of DQN is shown in Algorithm 1.

---

**Algorithm 1.** The training process of DQN.

1: Initialize replay memory D to capacity N
2: Initialize Q-function with random weights $\theta$
3: Initialize Q-function with weights $\theta = \theta$
4: for episode := 1 to M do
  5:   Initialize sequence $s_1 = \{x_1\}$ and preprocessed sequence $\phi_1 = \phi(s_1)$
  6:   for t := 1 to T do
    7:     select $a_t$, then observe reward $r_t$ and $s_{t+1}$
    8:     Processed $\phi_{t+1} = \phi(x_{t+1})$ and store transition $(\phi_t, a_t, r_t, \phi_{t+1})$ in D
  9:     Sample mini-batch of transition $(\phi_j, a_j, r_j, \phi_{j+1})$
10:    Set $y_j = \{r_j, r_j + \gamma \max_{a'} Q(\phi_{j+1}, a', \theta)\}$ if episode terminates at step $j + 1$
11:    end for
12:   Perform a gradient descent step on $(y_j - Q(\phi_j, a_j, \theta))$ w.r.t. network parameter $\theta$
13: end for
14: $i \rightarrow i + 1$
15: end for

---
function, \( M \) and \( t \) respectively represent the number of iterations and the number of training samples. However, the combination of deep learning (DL) and reinforcement learning (RL) will inevitably lead to some problems, such as:

1. DL needs a large number of labeled samples for supervised learning; RL only has the reward of state-action pairs as return value.
2. The sample of DL is independent, and the state of RL is correlative.
3. The distribution of DL targets is fixed; the distribution of RL is always changing, that is, it is difficult to reproduce the situation that was trained before.
4. Previous studies have shown that the use of non-linear networks to represent value functions is unstable.

This article used “Rainbow” to build the training model of MAS-DQN. Rainbow\(^3\) is an integrated DQN. It mainly integrates the following algorithms to try to solve the above problems:

1. Double Q-Learning constructs two neural networks with the same structure but different parameters: behavior network and target network. When updating the model, the best action at time \( t + 1 \) is selected by behavior network, then the target value of the optimal action at \( t + 1 \) time is estimated by target network. By decoupling these two steps, it can reduce the impact of Q-Learning method on overestimation of value.
2. Priority replay buffer makes the model choose more samples that can improve the model to use the replay memory more efficiently.
3. Dueling DQN decomposes the value function into two parts, making the model easier to train and expressing more valuable information.
4. Distributional DQN changes the output of the model from computational value expectation to computational value distribution in order to let the model can learn more valuable information.
5. Noisy DQN adds some exploratory ability to the model by adding noise to the parameters, which is more controllable.
6. In order to overcome the weakness of slow learning speed in Q-learning earlier stage, multistep learning uses more steps of reward, the target value can be estimated more accurately in the early stage of training, thus speeding up the training speed.

Rainbow model integrates the advantages of the above models which proves the possibility of integrating the above features successfully.

**The network for propose mechanism**

**Observation function for MAS-DQN based on ORBSLAM**

In deep RL, the local information obtained by an agent is called observation. Unlike previous deep RL algorithms, the observation function of the proposed mechanism models all agents and determines the behavior of each agent at the same time instead of for only one agent. In this section, ORBSLAM feature vector and output of the observation function are introduced.

**ORBSLAM feature vector.** ORBSLAM\(^4\) is one of the most commonly used system among modern visual SLAM frameworks. It runs in three threads: In tracking thread, the system tracks motion of the platform by using tracked feature points; local mapping thread is used for constructing global map and optimizing local map, loop closure thread takes responsibility for detecting loop when agent revisited to an already mapped area, which can prevent increase of the accumulative errors. ORBSLAM is deployed on each agent independently to simultaneously construct the map. In our system, the state of the environment consists of all pose estimation states of agents which are mainly determined by three items: map points, key frames, and loop closure detection. Some variables related to the three items are used to compose the state vector of the agent called ORBSLAM feature vector whose members are introduced as follow:

1. The first of component of the ORBSLAM feature vector is a variable related to the map point. The map point is constructed from the matched point features. We include the number of map points observed from current frame in feature vector.
2. The second of component of the feature vector is a variable related to the key frames which are the selected frames according to strict requirements to ensure that they contain sufficient map information. Some key frames proved to contain insufficiently accurate map information will be gradually removed. In order to estimate the content of map information at nearby locations which play a decisive role in pose estimation, the number of the generated new key frames and that of the eliminated old key frames from our last sampling time in the state vector are included in the state vector.
3. The third of component of the feature vector is a variable related to the result of loop closure. Loop closure detection is the problem that a mobile agent determines whether itself has returned to a previously visited location. If a closed-loop is detected, it will impose a strong constraint on global optimization. In feature vector, we add the time interval of the frame from last loop closure detected to measure the impact of loop closure to the current frame.
Algorithm 2. Observation function of the proposed algorithm.

| input: O : Set of ORBSLAM feature vector. |
| input: t : Current timestamp. |
| output: X : Global perception of the current state of the environment. |

1. function ObservationFunction(O,j)
2. for i = 1: n do
3. for j = 1: m do
4. if t-i+i > 0 then
5. x.extend(O(t-i+i,j))
6. else
7. x.extend(O(1,j))
8. end if
9. j = j + 1
10. end for
11. i = i + 1
12. end for
13. end function

4. Another critical factor is the collection of distances from the agent to other agents. The longer the distance from the agent to target agent is, the bigger difficult to it to assist the target. The distance is calculated by D* algorithm. The characteristic of D* algorithm is that when the terrain between the target point and the source point is known, the shortest path and the shortest distance can be obtained. If not fully known, the shortest path in the known terrain and the estimated distance between the target point and the source point can be output. An ORBSLAM feature vector contains $m-1$ ($m$ is the number of agents) such distances, representing the distance between the agent and the other $m-1$ agents, respectively.

An ordered arrangement of all the above variables is an ORBSLAM feature vector. On the next subsection, the observation function base on the ORBSLAM feature vector will be introduced.

The observation function. The output of the observation function represents the perception of the overall SLAM state of the whole MAS. The ORBSLAM feature vectors of all agents at the same time form an observation frame. Considering the continuity of the SLAM process, the previous states should be taken into count when the system makes macro-decision for each agent, so that the output of the observation function should include the current frame and the fixed number of preceding frames. These frames consist of an observation frame set, which eventually flattens into a vector as the output of the observation function. The specific steps of the observation function are shown in Algorithm 2.

While the parameter $n$ is the number of observation frames, $m$ is the total number of agents in the system, and $O(i,j)$ is the ORBSLAM2 feature vector of agent $j$ at time $i$.

**Reward function**

The reward function is generally used to reflect the immediate benefit of an agent reaching a new state through action. In our proposed algorithm, the transition error is used to measure the reward, which is shown in the equation (3)

$$\text{loss} = \sqrt{(x' - x)^2 + (y' - y)^2 + (z' - z)^2} \quad (3)$$

In order to embody the collaboration between agents, the reward function also adds the influence of agent’s target on his decision-making. The final reward function of agent is as equation (4)

$$r(i) = -(\mu \cdot \text{loss}(i) + \varepsilon (1 - \mu) (\text{loss}_k(t) - \text{loss}_{k-1}(t))) \quad (4)$$

where $\text{loss}_k(i)$ represents the loss of agent $i$ at time $k$. $(\text{loss}_k(t) - \text{loss}_{k-1}(t))$ denotes the contribution of agent $i$ to his target $t$ to assist. As the selfishness of the agent, $\mu$ is used to control the frequency of agent reorganization, $\varepsilon$ depends on the result of the assistance, and its value, such as the equation (5), is discussed in detail in the next subsection

$$\varepsilon = 1 \quad \text{When the target agent is successfully assisted by this agent}$$
$$\varepsilon = 0 \quad \text{When the target is itself or the position for collaboration is not reached}$$
$$\varepsilon = -1 \quad \text{When the collaboration is failed} \quad (5)$$

**Relative observation between agents in the proposed mechanism**

In the proposed mechanism, agents execute the order given by MAS-DQN through mutual observation with each others. As shown in Figure 2, red agents are assigned to assist
blue agents. First, they shall reach the predetermined observation position. If all the predetermined positions cannot be reached, the observation will be judged as a failure, agent obtains the pose of the target through the relative localization of the target agent. The obtained position and posture will provide a strong constraint for the target agent to optimize the pose graph. In another word, it is used to improve the accuracy of correcting pose. This observation actually has the same effect as the loop closure detection. The way of calculating the pose of the target agent is introduced consequently.

Assume that agent $i$ observes a feature point $p_{ij}^{c}$ of target agent $j$ whose coordinate in agent $i$’s camera coordinate system is $p_{ij}^{c}$, then the coordinate $p_{ij}^{w}$ in the world coordinate frame based on agent $i$’s pose can be obtained by equation (6)

$$p_{ij}^{w} = R_{i} p_{ij}^{c} + t_{i} \quad (6)$$

where $R_{i}$ refers to agent $i$’s rotation matrix which converts camera coordinates into world coordinates and $t_{i}$ refers to translation matrix. $R$ and $t$ together constitute the pose of robot camera. The world coordinates of one feature point are the consistent, the relationship between the world coordinates of point $p$ and the coordinates of point $p$ in the camera coordinate system of agent $j$ is shown in equation (7).

$$p_{w} = R_{j} p_{r} + t_{j} \quad (7)$$

where $R_{j}$ and $T_{j}$ is the pose of agent $j$, $p_{r}$ is the coordinate of feature point $p$ in agent $j$’s 3-D model. Because there may be multiple agents observing agent $j$ at the same time, $p_{w}$ may have multiple values, which makes it impossible to
obtain the pose of agent \( j \) directly. Nonlinear optimization method can be used to continuously approximate the true value of pose. Firstly, the error of pose estimation is defined as equation (8)

\[
e = \frac{1}{2} \sum_i \sum_k || p^k_j - (R'_j * p'_k + t'_j) ||^2_2
\] (8)

where \( p^k_j \) is the coordinate of \( k \)th point of agent \( j \) observed by agent \( i \). \( R'_j, t'_j \) is the estimated value of agent \( j \)'s pose matrix at this moment. \( R'_j, t'_j \) can be expressed by Lie algebra \( \exp(\xi) \), then the optimization of the target function can be expressed as

\[
\min_{\xi} \left( \frac{1}{2} \sum_i \sum_k || p^k_j - \exp(\xi) ||^2_2 \right)
\] (9)

The derivative of error term with respect to pose can be obtained by using Lie algebraic perturbation model, and the value of \( R'_j, t'_j \) can be obtained iteratively when \( e \) is the minimum value by using nonlinear optimization methods such as the gradient descent method. There are different solutions to solve this problem, such as gradient descent algorithm, Gauss Newton algorithm, Levenberg–Marquardt algorithm, and so on. In this article, Gauss Newton algorithm is adopted. Its time complexity is \( O(TM) \), where \( T \) is the total number of iterations, and \( M \) is the total number of error terms.

**Effectiveness proof of the proposed algorithm**

In this part, we will prove the effectiveness of our algorithm. That is to give the demonstration that after experience replay, the proposed DQN can make macro-decision to improve the efficiency of the whole MAS for each agent after training. We use the utility theory to model the performance of the whole MAS. As shown in the previous section, the effect of SLAM of a single agent \( x \) can be regards the immediate utility value of \( x \) which is associated with loss(x)

\[
U(x) = -\text{loss}(x)
\] (10)

The expectation of the immediate utility value of a non-cooperative MAS is as follows

\[
E(U(X)) = \frac{\sum_{x \in X} u(x)}{|X|}
\] (11)

The calculation of that in collaborative MAS is relatively complex since the effect of collaboration between agents on the overall system utility value should be considered, we use a graph to represent the cooperation of the whole system. An agent will be regard as a node in the graph and a directed edge from the itself to the target will be established as shown in Figure 3. If the collaboration is not executed at that time, the edge is the solid line. If the collaboration is just completed, the edge is a double-line edge. If the collaboration fails, the line is a dashed line, and if there is no collaboration object, the arrow points to itself.

In summary, to calculate the cumulative reward of all agents is to calculate the total cumulative utility value. MAS-DQN will gradually learn the cumulative utility function \( U(S, A(X)) \). Where \( S \) and \( A(X) \) respectively refer to the state space and action space of MAS.

The last layer of MAS-DQN is the decision-making layer. All the layers before the decision layer are called feature extraction layer. The feature extraction layer is actually a high-level abstraction of ORBSLAM feature vector. The weights connected with the decision-making level determine the discrepancy of different decision-making of agents. That is, in the face of the same extracted high-level characteristics, different action produce different cumulative reward. This situation is caused by the different attributes between agents, which DQN learns through the cooperation between agents.

**Sequences of the proposed mechanism**

The implementation steps of the proposed overall cooperation mechanism for multi-agent SLAM are shown in Algorithm 3.
The system consists of mobile agents and an organizer which MAS-DQN is deployed on. The following is a detailed explanation of this algorithm: Initialize the whole algorithm and all variables (01). Organizer broadcasts the requirements for synchronizing agent information and waits for the reply of all the agents (03–04). Each agent responds to the organizer with their world coordinates, built local maps, ORBSLAM feature vectors, and assistance status (05–07). The local maps are not the final form of maps. They are only 2-D maps that provide information for agent global navigation. After the organizer obtains the information returned by all agents (08), the first thing to do for it is to merge the local maps of all agents and feedback the global maps to agents (09–10). The second mission is to get the output of the observation function according to the set of ORBSLAM feature vector of all agents (11). Next, the immediate reward of each agent is calculated (12). \[
T(\phi_{t-1}, q_{t-1}, r_t, \phi_t) \]
is added to experience reply buffer (15–16). After obtaining enough sample information, MAS-DQN will be trained (17–21). An agent x decodes its own goals from the organizer’s instructions a (23). Agent x’s target means is the agent which it assists, if the target of it is itself x, x will SLAM independently (24–25). If x is not assigned a target, it will stop waiting for assistance (26–27). If x gets one target, it will follow the calculated path to approach his intended location and update his assistance status (28–54). Life is a constant, which is set up to prevent assistance from going on indefinitely (30–36). This loop will continue until time runs out (02–57).

**Algorithm 3. Complete algorithm.**

1: initialize the MAS-DQN and agents
2: for \( t = 1, T \) do
3: \( \text{sendRequest}() \)
4: \( \text{wait for } \text{response}() \)
5: for each \( x \in X \) do
6: \( x.\text{send}(\phi, x.\text{localmap}, x.\phi, x.\text{assistantStatus}) \)
7: \( \text{end for} \)
8: \( \text{getAllResponse}() \)
9: \( \text{globalMap} \leftarrow \text{mergeMaps}([\text{localMaps}]) \)
10: \( \text{broadcast}(\text{globalMap}) \)
11: \( \text{generateObservation}() \)
12: \( \text{calculateReward}() \)
13: end MAS-DQN with \( \phi \) and get every agent’s action \( a \)
14: \( \text{broadcast}(a) \) for each \( x \in X \)
15: if \( t > 1 \) then
16: \( T \rightarrow \text{getTransition}(\phi_{t-1}, q_{t-1}, r_t, \phi_t) \)
17: \( \text{addExperienceReplyBuffer}(T) \)
18: if \( |\text{ReplyBuffer}| > \text{Minimum quantity} \) then
19: \( \text{MAS-DQN train}(|\text{ReplyBuffer}|) \)
20: \( \text{end if} \)
21: \( \text{end if} \)
22: for each \( x \in X \) do
23: \( x.\text{target} \rightarrow x.\text{getAction}() \)
24: if \( x.\text{target} = x \) then
25: \( x.\text{randomMove}() \)
26: else if \( x.\text{target} = \text{None} \) then
27: \( x.\text{stop}() \)
28: else
29: \( x.\text{path} \rightarrow \text{calculatePath}(x.\text{target}) \)
30: if \( x.\text{target} \neq x.\text{target}_{t-1} \) then
31: \( x.\text{life} \rightarrow 0 \)
32: else
33: \( x.\text{life} \rightarrow x.\text{life} + 1 \)
34: if \( x.\text{life} = \text{Life then} \)
35: \( x.\text{assistantStatus} \rightarrow \text{fail} \)
36: \( \text{end if} \)
37: \( \text{end if} \)
38: \( x.\text{nextStep}(x.\text{path}) \)
39: if \( x.\text{stop}() \) then
40: \( x.\text{observe}(x.\text{target}) \)
41: if \( x.\text{get}(x.\text{target}) \) then
42: \( x.\text{successful} \rightarrow \text{true} \)
43: \( x.\text{assistantStatus} \rightarrow \text{success} \)
44: \( \text{broadcast}(x.\text{assistantComplete}(x.\text{target})) \)
45: else
46: \( x.\text{assistantStatus} \rightarrow \text{fail} \)
47: \( \text{end if} \)
48: \( \text{end else} \)
49: if another agent completed assistance to \( x.\text{target} \) then
50: \( x.\text{assistantStatus} \rightarrow \text{ fail} \)
51: \( \text{end else} \)
52: \( x.\text{assistantStatus} \rightarrow \text{approaching} \)
53: \( \text{end if} \)
54: \( \text{end if} \)
55: \( \text{end if} \)
56: \( \text{end for} \)
57: \( \text{end for} \)

Finally, the output is imported to MAS-DQN to get the action of each agent which will be broadcasted (13–14). \( T(\phi_{t-1}, q_{t-1}, r_t, \phi_t) \) is added to experience reply buffer (15–16). After obtaining enough sample information, MAS-DQN will be trained (17–21). Agent x decodes its own goals from the organizer’s instructions a (23). Agent x’s target means is the agent which it assists, if the target of it is itself x, x will SLAM independently (24–25). If x is not assigned a target, it will stop waiting for assistance (26–27). If x gets one target, it will follow the calculated path to approach its intended location and update its assistance status (28–54). Life is a constant, which is set up to prevent assistance from going on indefinitely (30–36). This loop will continue until time runs out (02–57).

**Experiment**

The simulation experiments introduced in this section is to evaluate the performance of the proposed mechanism and MAS-DQN with different parameters. The experiment was implemented on the Robot Operating System (ROS) platform which can be used to simulate the real physical environment. We build one simulation mobile agent with the “telobot” model which equips “Kinect” as vision sensor. The initial pose of each robot is known and ORBSLAM is deployed independently on each robot. The experimental world is shown as Figure 4.

The main indicators to measure the effect of SLAM in MAS are translation root mean square error (RMSE), orientation RMSE of the tracks of agents. Although agents in experimental environments have the same model, we can embody the differences of their attributes by changing their ability values related to motion and vision sampling. Their maximum angular acceleration, linear acceleration, maximum angular velocity, and maximum linear velocity obey...
normal distribution \(N(\mu, \sigma_1)\), in which \(\sigma_1\) is used to control the discreteness of the ability values of agents, so that the performance of the algorithms can be tested when the value of agents' attributes vary with different degrees. When it comes to visual sensors, we add noise that obeys the normal distribution \(N(\mu, \sigma_2)\) to the images captured by visual sensor, and the variance of the camera noise of different agents \(\sigma_2\) still obeys the normal distribution \(N(1, \sigma_1)\).

We set up four experiments to verify the effectiveness of our algorithm. The following are the details of all experiments and the corresponding experimental results. The experiments have been done with \(\sigma_1 = 0.2\). The time step is set as 0.5 s. All the comparative experiments were done after the convergence of training process for MAS-DQN.

Experiment 1: This experiment shows the role of collaboration in multi-agent SLAM. The results are shown in Figure 5 in which the two algorithms for comparison are explained as follows:

- Case1: The proposed collaboration mechanism based on MAS-DQN.
- Case2: Each agent SLAM independently without any collaboration.

The proposed algorithm reduce translation RMSE (m) by 35.68% orientation RMSE (°) by 32.02% compared with the mechanism without coordination.

We use these two different mechanisms to generate octomaps \(^{38}\) of the same environment, which are shown in Figures 6 and 7.

Experiment 2: The main purpose of this experiment is to compare mechanism that agents collaborate with each other without considering attributes and states. The
The experimental results are shown in Figure 8, which illustrate that the proposed algorithm reduces translation RMSE by 28.33% and orientation RMSE by 31.68%.

- Case 1: The proposed collaboration mechanism based on MAS-DQN.
- Case 2: Relative localization based on cluster matching algorithm without distance IR in the environment.\(^{39}\)

Experiment 3: This experiment is used to compare the proposed algorithm with other feature-based MRTA algorithms that take agent attributes as constants. The experimental results are shown in Figure 9.

- Case 1: The proposed collaboration mechanism based on MAS-DQN.
- Case 2: The proposed collaboration mechanism based on emotional recruitment model.
- Case 3: The proposed collaboration mechanism based on auction model.

### Table 1. Translation RMSE (m) with different variance.

| Algorithm                  | Variance | 0.15 | 0.2  | 0.25 |
|----------------------------|----------|------|------|------|
| Emotional recruitment model|          | 0.0638 | 0.0692 | 0.0752 |
| Auction model              |          | 0.0657 | 0.0717 | 0.0761 |
| Proposed algorithm         |          | 0.0568 | 0.0602 | 0.0643 |

### Table 2. Orientation RMSE with different variance.

| Algorithm                  | Variance | 0.15 | 0.2  | 0.25 |
|----------------------------|----------|------|------|------|
| Emotional recruitment model|          | 4.689 | 4.849 | 5.620 |
| Auction model              |          | 4.792 | 5.208 | 5.776 |
| Proposed algorithm         |          | 3.326 | 3.733 | 3.928 |

The proposed algorithm reduces translation RMSE (m) by 13.00% and orientation position RMSE (°) by 23.01% compared with the mechanism based on emotional recruitment model. Compared with auction model, the MAS-
DQN reduced translation RMSE by 16.03% and orientation RMSE by 28.32%.

Experiment 4: In order to test the effectiveness of our algorithm in dealing with different degrees of difference in agent attributes, experiment 4 were performed when variances \( \sigma_1 \) was set to different values. The result of comparative experiment is show in Tables 1 and 2. The proposed method reduces the average growth rate of translation RMSE by 16.58% and 26.10% compared with auction model and emotional recruitment model. The average growth rate of orientation RMSE is reduced by 10.04% and 7.48%.

Experiment 5: This experiment mainly compares our algorithm and the multi-agent SLAM algorithm called MR-vSLAM which only uses the prior information of map to collaborate in the aspect of localization. The experimental results are shown in the Figure 10, which illustrate that the proposed algorithm reduces translation RMSE by 7.81% and orientation RMSE by 13.80%.

Table 3 compares the computational costs of several algorithms mentioned in this article. Where the \( n \) is the number of agent, \( k \) is the number of natural landmark, \( L \) is the number of hidden layers of the MAS-DQN, \( C_l \) is the number of neurons of the \( l \)th hidden layer.

### Conclusion

This article proposes a novel centralized multi-agent cooperative SLAM mechanism with considering the status and attributes of agents. The algorithm applies a new observation function based on ORBSLAM to apperceive the SLAM state of the whole MAS, a rainbow-based deep RL framework called MAS-DQN is designed to learn the overall utility function \( U(S, A(x)) \) of the system, and the effectiveness of the framework is proved. The simulation results show MAS-DQN reduces transition RMSE (m) by 13.00%, 13.06%, and 7.81% compared with emotional recruitment model, auction model, and MR-vSLAM, the orientation RMSE (°) is also reduced by 24.42%, 29.64%, and 13.80% which is considered as great improvement on recently researched approaches. This algorithm inescapable brings huge computational cost to organizer, which makes the real time of the system unable to be guaranteed. In the future, we will focus on distributed learning system to reduce the burden.

### Declaration of conflicting interests

The author(s) declared no potential conflicts of interest with respect to the research, authorship, and/or publication of this article.

### Funding

The author(s) disclosed receipt of the following financial support for the research, authorship, and/or publication of this article: This work was supported by a special fund project of Harbin Science and Technology Innovation Talents Research (grant no. RC2013XK010002) and the National Natural Science Foundation of China (grant no. 61375081).

### ORCID iDs

Zhaoyi Pei https://orcid.org/0000-0002-6478-6263  
Muhammad Zuhair Qadir https://orcid.org/0000-0002-9723-3403
References

1. Rahman MS, Mahmud MA, Oo AMT, et al. Multi-agent approach for enhancing security of protection schemes in cyber-physical energy systems. *IEEE Trans Ind Inform* 2017; 13(2): 436–447.

2. Azimi S, Delavar MR, and Rajabifard A. Developing a multi-agent based modeling for smart search and rescue operation. *Geoinformation for disaster management conference*, Girne, Cyprus, 14 March 2018, pp. 133–157. Cham: Springer.

3. Du SL, Sun XM, Cao M, et al. Pursuing an evader through cooperative relaying in multi-agent surveillance networks. *Automatica* 2017; 83: 155–161.

4. Saaidia R, Bellamine MS, and Amor AB. Temporal performances evaluation of multi-robot demining system inspired by ant behavior. *Int J Comput Sci Inf Secur* 2016; 14(6): 16.

5. Espina MV, Grech R, De Jager D, et al. Multi-robot teams for environmental monitoring. *Innov Def Support Syst* 2011; 336: 183–209.

6. Shkurti F, Xu A, Meghjani M, et al. Multi-domain monitoring of marine environments using a heterogeneous robot team. *2012 IEEE/RSJ international conference on intelligent robots and systems*, Vilamoura, Portugal, 7–12 October 2012, pp. 1747–1753. IEEE.

7. Rodriguez S, De Paz JF, Villarrubia G, et al. Multi-agent information fusion system to manage data from a WSN in a residential home. *Inf Fusion* 2015; 23: 43–57.

8. Jemaa AB, Ltifi H, and Ayed MB. Visual intelligent remote healthcare monitoring system using multi-agent technology. *2016 2nd international conference on advanced technologies for signal and image processing (ATSIP)*, Monastir, Tunisia, 21–23 March 2016, pp. 465–471. IEEE.

9. Surmann H, Berninger N, and Worst R. 3D mapping for multi hybrid robot cooperation. *2017 IEEE/RSJ international conference on intelligent robots and systems (IROS) 2017*, Vancouver, BC, Canada, 24–28 September 2017, pp. 626–633. IEEE.

10. Mur-Artal R, Montiel JMM, and Tardós JD. ORB-SLAM: a versatile and accurate monocular SLAM system. *IEEE Trans Robot* 2017; 31(5): 1147–1163.

11. Lazaro MT, Paz LM, Pinies P, et al. Multi-robot SLAM using condensed measurements. In: *Proceedings of the 2013 IEEE/RSJ international conference on intelligent robots and systems (IROS)*, Tokyo, Japan, 3–7 November 2013, pp. 1069–1076. IEEE.

12. Grisetti G, Kümmerle R, Stachniss C, et al. A tutorial on graph-based SLAM. *IEEE Intell Transp Syst Mag* 2011; 2: 31–43.

13. Lauritzen PH, Nair RD, and Ulrich PA. A conservative semi-Lagrangian multi-tracer transport scheme (CSLAM) on the cubed-sphere grid. *J Comput Phys* 2010; 229(5): 1401–1424.

14. Eliazar AI and Parr R. DP-SLAM 2.0. In: *Proceedings of the IEEE international conference on robotics and automation*, New Orleans, LA, USA, 26 April–1 May 2004, pp. 1314–1320. IEEE.

15. Huletski A and Kartashov D. A SLAM research framework for ROS. *Proceedings of the 12th Central and Eastern European software engineering conference in Russia*, Moscow, Russia, October 2016. Article No.12.

16. Al-Yafi K, Lee H, and Mansouri A. Mtap-masim: a multi-agent simulator for the mobile task allocation problem. In: *IEEE international workshop on enabling technologies: infrastructures for collaborative enterprises*, Groningen, Netherlands, 29 June 2009, pp. 25–27.

17. Colin B and Veloso M. Mobile robot task allocation in hybrid wireless sensor networks. In: *Intelligent robots and systems (IROS)*, Taipei, Taiwan, 18–22 October 2010, pp. 2932–2937. IEEE.

18. Liu C and Kroll A. A centralized multi-robot task allocation for industrial plant inspection by using a* and genetic algorithms. In: *Proceedings, part II 11th international conference artificial intelligence and soft computing*, Zakopane, Poland, 29 April–3 May 2012, pp. 466–474.

19. Higuera J and Dudek G. Fair subdivision of multi-robot tasks. In: *2013 IEEE international conference on robotics and automation (ICRA)*, Karlsruhe, Germany, 6–10 May 2013, pp. 3014–3019. IEEE.

20. Wadhwania S, Kim DK, Omidshafiee S, et al. Policy distillation and value matching in multiagent reinforcement learning. arXiv preprint arXiv:1903.06592, 2019.

21. Farinelli A, Iocchi L, and Nardi D. Distributed on-line dynamic task assignment for multi-robot patrolling. *Auton Robots* 2017; 41(6): 1321–1345.

22. Luo L, Chakraborty N, and Sycara K. Provably-good distributed algorithm for constrained multi-robot task assignment for grouped tasks. *IEEE Trans Robot* 2015; 31(1): 19–30.

23. Das GP, McGinnity TM, Coleman SA, et al. A distributed task allocation algorithm for a multi-robot system in healthcare facilities. *J Intell Robot Syst* 2015; 80(1): 33–58.

24. Safari M, Xu H, and Carrillo LRG. Brain emotional learning-based intelligent controller for flocking of multi-agent systems. *American control conference*, Seattle, WA, USA, 24–26 May 2017. IEEE.

25. Choi HL, Brunet L, and How JP. Consensus-based decentralized auctions for robust task allocation. *IEEE Trans Robot* 2009; 25(4): 912–926.

26. Brunet L, Choi HL, and How JP. Consensus-based auction approaches for decentralized task assignment. *AIAA guidance, navigation and control conference and exhibit*, Honolulu, Hawaii, 18–21 August 2008, p. 6839.

27. Choi HL, Kim KS, Johnson LB, et al. Potential game-theoretic analysis of a market-based decentralized task allocation algorithm. In: Asama H and Fukuda T (eds) *Distributed autonomous robotic systems*. Tokyo: Springer, 2016, pp. 207–220.

28. Johnson L, Choi HL, and How JP. The hybrid information and plan consensus algorithm with imperfect situational awareness. In: Asama H and Fukuda T (eds) *Distributed autonomous robotic systems*. Tokyo: Springer, 2016, pp. 221–233.
29. Saim M, Munawar K, and Al-Saggaf U. An overview of localization methods for multi-agent systems. *Int J Eng Res Appl* 2017; 7(01): 19–24.

30. Shames I, Fidan B, Anderson BD, et al. Self-localization of mobile agents in the plane. In: *2008 3rd international symposium on wireless pervasive computing santorini*, Greece, 7–9 May 2008, pp. 116–120. IEEE.

31. Zhou XS and Roumeliotis SI. Multi-robot SLAM with unknown initial correspondence: the robot rendezvous case. In: *Proceedings of the 2006 IEEE international conference on intelligent robots and systems*, Beijing, China, 9–15 October 2006, pp. 1785–1792. IEEE.

32. Zkucur NE and Levent NH. Cooperative multi-robot map merging using Fast-SLAM. In: *RoboCup 2009*. Berlin: Springer-Verlag, 2010.

33. Martinelli A, Pont F, and Siegwart R. Multi-robot localization using relative observations. *Proceedings of the 2005 IEEE international conference on robotics and automation*, Barcelona, Spain, 18–22 April 2005. IEEE.

34. Prorok A and Martinoli A. A reciprocal sampling algorithm for lightweight distributed multi-robot localization. *2011 IEEE/RSJ international conference on intelligent robots and systems*, San Francisco, CA, USA, 25–30 September 2011. IEEE.

35. Burgard W. Experiences with an interactive museum tour-guide robot. *Artif Intell* 1999; 114: 3–55.

36. Mnih V and Kavukcuoglu K. Human-level control through deep reinforcement learning. *Nature* 2015; 518(7540): 529–533.

37. Hessel M, Modayil J, Van Hasselt H, et al. Rainbow: combining improvements in deep reinforcement learning. *Thirty-second AAAI conference on artificial intelligence*. New Orleans, LA, USA, February 2–7 2018.

38. Hornung A, Wurm KM, Bennewitz M, et al. OctoMap: an efficient probabilistic 3D mapping framework based on octrees. *Auton Robot* 2013; 34(3): 189–206.

39. Rashid AT and Frasca M. Multi-robot localization and orientation estimation using robotic cluster matching algorithm. *Robot Auton Syst* 2015; 63:108–121.

40. Quande Y. *Research of multi-robot cooperative SLAM based on Vision [D]*. Harbin: Harbin Institute of Technology, 2016.