Robot Navigation among External Autonomous Agents through Deep Reinforcement Learning using Graph Attention Network

Tianle Zhang [*,**] **Tenghai Qiu** [*,**] **Zhiqiang Pu** [*,**] **Zhen Liu** [*,**] **Jianqiang Yi** [*,**]

[*] *School of Artificial Intelligence, University of Chinese Academy of Sciences, Beijing 100049, China (e-mail: tianle-zhang@outlook.com, tenghai.qiu@ia.ac.cn, zhiqiang.pu@ia.ac.cn, liuzhen@ia.ac.cn, jianqiang.yi@ia.ac.cn).*
[**] *Institute of Automation, Chinese Academy of Sciences, Beijing, 100190, China*

Abstract: Finding collision-free and efficient paths in an uncertain dynamic environment is a challenge for robot navigation tasks, especially when there are external autonomous agents that also have decision-making abilities in the same environment. This paper develops a novel method based on DRL with graph attention network (GAT) to solve the problem of robot navigation among external autonomous agents (other agents). Specifically, GAT is adopted to describe the robot and other agents as a specific graph, and extract the spatial structural influence features of other agents on the robot from the graph. Multi-head attention mechanism is utilized to calculate the weights of interactions between the robot and other agents. This GAT uses observations of an arbitrary number of other agents in dynamic environments. Furthermore, the proposed method is combined with optimal reciprocal collision avoidance to improve its safety in new environments. Various simulations demonstrate that our method has good performance and robustness in different environments.

Keywords: Robot navigation, deep reinforcement learning (DRL), graph attention network.

1. INTRODUCTION

With rapid developments in recent decades, robots play an increasingly important role in life, such as service robots and logistics robots. In a common mobile robot application scenario, a robot faces the challenge of avoiding collisions among pedestrians. The pedestrians not only are taken as moving obstacles, but also make autonomous decisions constantly. Since neither the robot nor pedestrians know the intents of the other, the trajectory of the robot may easily oscillate. This phenomenon causes the collision avoidance problem as a robot navigating in a world of external autonomous agents (regarded as other agents). In addition, when the robot have local observation in uncertain dynamic environments and the communications among agents are limited, in order to reach the goal efficiently and safely, the robot needs to perceive and anticipate the policies and intents of other agents. These can not be measured directly, but can be inferred indirectly. Hence, robot navigation with safety and efficiency among external autonomous agents with decision-making abilities remains challenging.

In existing works, the research of robot navigation can be roughly classified into two categories, i.e. non-learning based and learning based methods. The non-learning based methods are further divided into reaction based and trajectory based methods. The former (van den Berg et al. (2011); Fiorini and Shiller (1998)) only considers one-step interactions with obstacles about the surrounding environment. For instance, optimal reciprocal collision avoidance (ORCA)(van den Berg et al. (2011)) provides sufficient conditions to avoid collisions in short term, which is an valuable property. However, the reaction based methods are short-sighted as they do not consider the future states of other agents. These methods can lead agents to generate oscillatory and unnatural behaviors. In contrast, the trajectory based method (Kuderer et al. (2012)) can anticipate the future states of other agents. But, they need to infer the intents of other agents (e.g. initial states and goals) through planning feasible paths for all neighboring agents in the environment. It is computationally expensive and requires more information about the surrounding environment.

Recent studies on robotic navigation have focused on learning based methods that develop an effective and efficient interaction rule by learning a value or policy function that implicitly encodes cooperation behaviors (Chen et al. (2017b)). Moreover, deep reinforcement learning (DRL) is used to optimize behavior policy through maximizing cumulative reward. The learning based methods on robot navigation are also divided into two categories: sensor-level and agent-level. The sensor-level methods (Long et al.

(2018); Fan et al. (2018)) directly map the raw sensor readings (2D laser scans or images) to desired, collision-free steering commands with end-to-end training (Long et al. (2017)). In the sensor-level methods, static and dynamic obstacles are processed as the same. However, dynamic obstacles have different characteristics from static obstacles in practical environments. In contrast, the agent-level methods map the observable states (e.g. shapes, velocities, and position) of other agents to steering commands (Everett et al. (2018); Chen et al. (2017b); Chen et al. (2017a)). Agent-level representations of the world extracted from multiple sensors (e.g. cameras and lidar) can express different motion characteristics of other agents. Moreover, the policies and intents of other agents are indirectly inferred with agent-level representations through DRL. However, in most existing agent-level methods, the robot receives information about the surrounding environment (e.g. the observable states of other agents) in the form of a lumped vector with everything stacked together, without utilizing the natural spatial structure of other agents in the real environment.

In reality, a multi-agent system can be naturally described as a graph structure. Meanwhile, since graph neural network (GNN) can be implemented to extract features of graph-structured data, it is widely used and deeply researched in non-Euclidean space. Among GNN, graph attention network (GAT)(Velickovic et al. (2017)) is a novel GNN architecture that operates on graph-structured data, leveraging masked self-attentional layers to address the interactions of nodes in a graph. Therefore, it is a natural idea that the graph-structure data of multiple agents can also be operated through GNN.

In this paper, we propose a novel method based on DRL with GAT to solve the problem of robot navigation among external autonomous agents that also have decision-making abilities. Specifically, GAT is used to describe the robot and other agents as a special graph, and extract the influence features of other agents on the robot from the graph. In GAT, multi-head attention mechanism is implemented to calculate the weights of interactions between the robot and other agents. Furthermore, ORCA is combined to improve the safety of the robot in new environments. Simulation results demonstrate the benefits of the proposed method.

2. PRELIMINARIES

In this section, the problem formulation of robot navigation will be described in details, and then, the ORCA method will be recapped briefly.

2.1 Problem Formulation

As shown in Fig. 1, a navigation task where one robot (blue circle) need to find a collision-free path among n external autonomous agents (blank circles) is investigated in this paper. The red arrows represent their direction. The robot with local observations can not communicate with other agents. This can be expressed as a sequential decision-making problem in the framework of RL (Chen et al. (2017b); Everett et al. (2018); Chen et al. (2019)). For simplicity, we assume that the geometry

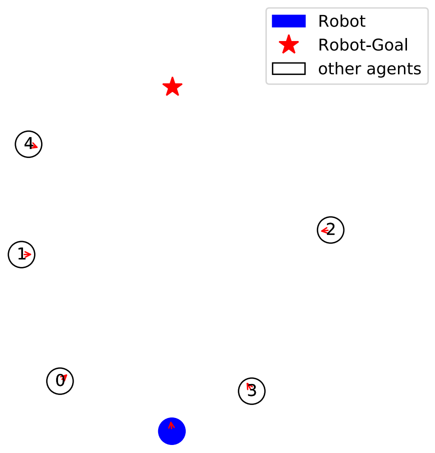

Fig. 1. Robot navigation among external autonomous agents (other agents).

of the robot and other agents are modeled as a disc with an actual shape radius. For the robot and other agents, the position $\boldsymbol{p}_i(t) = [p_{ix}(t), p_{iy}(t)]$, velocity $\boldsymbol{v}_i(t) = [v_{ix}(t), v_{iy}(t)]$ and radius r_i can be observed by the others ($i = 0$ represents the robot, $i = 1, ..., n$ represents other agents), and the goal position $\boldsymbol{p}_g = [p_{gx}, p_{gy}]$, preferred speed v_{pref} and its direction $\Theta(t)$ is unobservable by the others. Its own state of the robot is represented as $\boldsymbol{s}^{self}(t) = \boldsymbol{s}_0^o(t) + [p_{gx}, p_{gy}, v_{pref}, \Theta(t)]$ in which $\boldsymbol{s}_0^o(t) = [p_{0x}(t), p_{0y}(t), v_{0x}(t), v_{0y}(t), r_0]$ can be observed by other agents. The states of other agents that can be observed by the robot are $\boldsymbol{s}_1^o(t), \boldsymbol{s}_2^o(t), ..., \boldsymbol{s}_n^o(t)$. We define $\boldsymbol{s}^{all}(t) = [\boldsymbol{s}^{self}(t), \boldsymbol{s}^o(t)]$, where $\boldsymbol{s}^o(t) = [\boldsymbol{s}_0^o(t), \boldsymbol{s}_1^o(t), \boldsymbol{s}_2^o(t), ..., \boldsymbol{s}_n^o(t)]$. The action of the robot is defined as $\boldsymbol{a}(t) = \boldsymbol{v}_0(t)$.

Deep V-learning method defined in Chen et al. (2019) is used to learn an approximate optimal value function, $V^*(\boldsymbol{s}^{all}(t))$. The value function implicitly encodes the estimated time of the robot to its goal. Meanwhile, an optimal policy, $V^*(\boldsymbol{s}^{all}(t))$ is used to generate $\pi^*(\boldsymbol{s}^{all}(t))$ which is a function of state mapping optimal action, $\boldsymbol{s}^{all}(t) \rightarrow \boldsymbol{a}(t)$:

$$\begin{aligned}\pi^*(\boldsymbol{s}^{all}(t)) = & \underset{\boldsymbol{a}(t)}{argmax}\, R_t(\boldsymbol{s}^{all}(t), \boldsymbol{a}(t)) + \gamma^{\Delta t \cdot v_{pref}} \\ & \int_{\boldsymbol{s}^{all}(t+\Delta t)} P(\boldsymbol{s}^{all}(t+\Delta t)|\boldsymbol{s}^{all}(t), \boldsymbol{a}(t)) \\ & V^*(\boldsymbol{s}^{all}(t+\Delta t)) d\boldsymbol{s}^{all}(t+\Delta t)\end{aligned} \quad (1)$$

where $\gamma \in (0, 1)$ is a discount factor, Δt is the time step, $P(\boldsymbol{s}^{all}(t+\Delta t)|\boldsymbol{s}^{all}(t), \boldsymbol{a}(t))$ is the probability of state transition from time t to time $t+\Delta t$, and $R_t(\boldsymbol{s}^{all}(t), \boldsymbol{a}(t))$ is the reward received by taking actions in current state. we improve the formulation of the sparse reward function defined in Chen et al. (2017b) and Everett et al. (2018),

$$R_t(\boldsymbol{s}_t^{all}, \boldsymbol{a}_t) = \begin{cases} 1 & if & \boldsymbol{p}_0(t) = \boldsymbol{p}_g \\ -0.25 & elseif & d_t < 0 \\ (d_t - D) * 0.5\Delta t & elseif & d_t < D \\ 0 & otherwise & \end{cases} \quad (2)$$

where D is the threshold of an uncomfortable distance between the robot and other agents, and d_t is the minimum distance between the robot and other agents.

2.2 A Recap Of ORCA

In a nutshell, ORCA requires two steps to determine a collision-free velocity $\boldsymbol{v}_A^+$ for a robot A. Firstly, it computes a set of velocities that form a permitted velocity space for the robot. If the robot selects a velocity in the permitted velocity space, it will not collide with other agents in a time horizon τ. The permitted velocity space is denoted as $ORCA_A^\tau$. Next, among these permitted velocities, the robot selects the collision-free velocity as a velocity that locates inside the permitted velocity space and is closest to its current preferred velocity $\boldsymbol{v}_A^{pref}$, i.e.,

$$\boldsymbol{v}_A^+ = \underset{\boldsymbol{v}\in ORCA_A^\tau}{argmin} \ ||\boldsymbol{v} - \boldsymbol{v}_A^{pref}||. \tag{3}$$

where $\boldsymbol{v}_A^{pref}$ is the preferred velocity used to guide toward its goal $\boldsymbol{p}_g$. Generally, for good performance, the parameters of ORCA need to be tuned carefully for different scenarios.

3. APPROACH

In this section, we propose a new method to solve the problem of robot navigation among external autonomous agents (other agents). The method involves a two-stage design, including DRL stage and DRL combined with ORCA stage. In the first stage, a DRL with GAT method is designed. Specifically, GAT is adopted to extract the influence features of other agents on the robot, and then, concatenating the influence features and the its own state of the robot inputs to a value network which estimates the value function. Finally, the DRL velocity of the robot is obtained by (1). In the second stage, the velocity obtained in the first stage is regarded as the preferred velocity of ORCA, and the final velocity performed of the robot is acquired through the ORCA method.

In the following, GAT is firstly introduced, and then, a navigation network architecture is presented. Finally, the combined framework of DRL and ORCA is given.

3.1 Graph Attention Network

In existing works, the information of an environment is mostly stacked together, which does not make full use of the spatial graph structure of the environment. In this work, GAT is adopted to describe the environment (the robot and other agents) as a special graph and extract features from the graph. Meanwhile, multi-head attention mechanism is adopted to compute the weights of interactions between the robot and other agents. Finally, the influence features of other agents on the robot are obtained. The policies and intentions of other agents are encoded implicitly by the influence features.

we define a graph $G := (V, E)$, where each node $v \in V$ is either other agent or the robot, and there exists an edge $e \in E$ between two nodes if the nodes can detect each other through their own sensors (e.g. laser or camera).

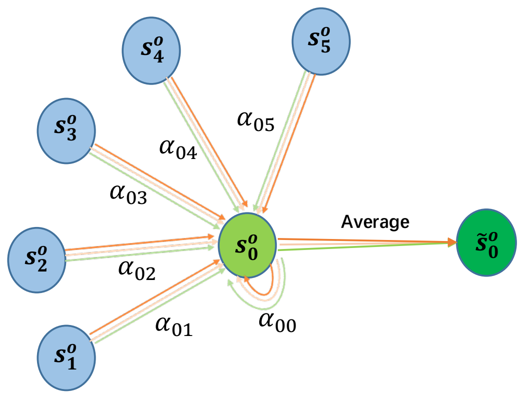

Fig. 2. Graph Attention Network with Multi-Head Attention Mechanism.

Static obstacles can be regarded as agents with speed of 0. Meanwhile, in order to make states more obvious, the observation states of the robot and other agents are transformed into:

$$\boldsymbol{s}_i^o = [p_{ix}, p_{iy}, v_{ix}, v_{iy}, r_i, d_i, r_i + r_0], i = 0, 1, ..., n \tag{4}$$

where $\boldsymbol{s}_0^o$ is the state of the robot that can be observed by other agents, $\boldsymbol{s}_1^o, \boldsymbol{s}_2^o, ..., \boldsymbol{s}_n^o$ are the states of other agents that can be observed by the robot, r_0 is the radius of the robot, $r_1, r_2, ..., r_n$ are the radius of other agents, and $d_1, ..., d_n$ are the distance between other agent $1, ..., n$ and the robot respectively. Moreover, $d_0 = 0$ represents the distance between the robot and itself.

Each node v_i has its own feature vector $\boldsymbol{s}_i^o$. Meanwhile, Each edge has a corresponding attention weight. Attention weights matrix $\boldsymbol{\alpha} = \{\alpha_{ij}\}$ can be obtained through two steps. First of all, attention correlation coefficients are calculated as:

$$c_{ij} = \begin{cases} LeakyReLU(\boldsymbol{a}^T[\boldsymbol{s}_i^o\boldsymbol{W}||\boldsymbol{s}_j^o\boldsymbol{W}]) & if \quad A_{ij} = 1 \\ 0 & otherwise \end{cases} \tag{5}$$

where $\boldsymbol{A} = \{A_{ij}\}$ is the adjacency matrix of the graph, $||$ is the concatenation operation, $LeakyReLU$ is a nonlinear activation function, $\boldsymbol{W}$ is a learnable parameter matrix of the corresponding input linear transformation and $\boldsymbol{a}^T$ is a learnable parameter vector. It should be noted that both $\boldsymbol{W}$ and $\boldsymbol{a}^T$ are obtained through DRL which will be addressed later. We only compute c_{ij} for nodes $j \in N_i$, where N_i is some neighborhood of node v_i in the graph. Next, the attention weights computed by the attention mechanism can be expressed as:

$$\alpha_{ij} = \frac{exp(c_{ij})}{\sum_{k\in N_i} exp(c_{ik})} \tag{6}$$

According to the above, the features of each node and attention weights between nodes have been obtained. Hence, we can calculate the influence features of other nodes on the node i, i.e.,

$$\tilde{\boldsymbol{s}}_i^o = \sigma(\sum_{j\in N_i} \alpha_{ij}\boldsymbol{s}_j^o\boldsymbol{W}) \tag{7}$$

where σ is a nonlinear activation function. Since one-head attention mechanism is unstable in learning process, multi-head attention mechanism can be used to improve stability in training. Specifically, K independent attention mechanisms execute the transformation of (7), and then their features are averaged, leading to the following output features representation:

$$\tilde{\boldsymbol{s}}_i^o = \frac{1}{K}\sum_{k=1}^{K}\sigma\Big(\sum_{j\in N_i}\alpha_{ij}^k \boldsymbol{s}_j^o \boldsymbol{W}^k\Big) \quad (8)$$

where K is the head number of attention mechanism (i.e. K-head attention mechanism). In this work, we only need to obtain the influence features (i.e. $\tilde{\boldsymbol{s}}_0^o$) of other agents on the robot in the set of GAT output $\tilde{\boldsymbol{s}}_i^o$. For example, the graph attention network with three-head attention mechanism and 5 other agents is shown in Fig. 2. The blue circles represent 5 other agents, and the green circle represents the robot. $\boldsymbol{s}_1^o, \boldsymbol{s}_2^o, \boldsymbol{s}_3^o, \boldsymbol{s}_4^o, \boldsymbol{s}_5^o$ are the observation states of other agents, and $\boldsymbol{s}_0^o$ is the observation state of the robot. The influence features of 5 other agents on the robot, $\tilde{\boldsymbol{s}}_0^o$, are obtained through GAT.

3.2 Navigation Network Architecture

In this work, the proposed DRL network framework is an end-to-end network architecture. In the process of the robot navigation, the intentions and policies of other agents are unknown. Particularly, the number of other agents in the environment is uncertain. In each step, actions of other agents will affect the behavior of the robot. Hence, it is important to process all observation states of other agents.

To solve these problems, as described in Section 3.1, GAT is adopted to describe the robot and other agents as a specific graph and extract the influence features of other agents on the robot from the graph. The influence features implicitly encode the intentions and policies of other agents. Meanwhile, since the parameters of GAT are shared and independent of the number of nodes, it can accept the inputs of the observation states of any number of other agents.

Based on the GAT module, the navigation network architecture is designed as Fig. 3. Inputing the observation state of other agents and the robot, $\boldsymbol{s}_i^o$, into GAT, the influence features of other agents on the robot, $\tilde{\boldsymbol{s}}_0^o$, are obtained in the output of GAT. $\tilde{\boldsymbol{s}}_0^o$ is fed into a fully-connected layer which outputs the feature vector, $\boldsymbol{s}^g$, which represents the features extracted from the environment around the robot. Then, $\boldsymbol{s}^e$ is formed through concatenating $\boldsymbol{s}^g$ with its own state of the robot, $\tilde{\boldsymbol{s}}^{self}$, which is expressed as:

$$\tilde{\boldsymbol{s}}^{self} = transform(\boldsymbol{s}^{self}) = [d_g, v_{pref}, \Theta, v_{0x}, v_{0y}, r_0] \quad (9)$$

where $d_g = ||\boldsymbol{p}_g - \boldsymbol{p}_0(t)||$ represents the distance between the robot and its goal position. The feature vector $\boldsymbol{s}^e$ contains information about the robot and the environment around it. Then, $\boldsymbol{s}^e$ is fed into a value network consisting of three fully-connected layers. Finally, the value network outputs the estimated value of current state. Under the condition of getting the value, the DRL velocity $\boldsymbol{v}_0^{DRL}(t)$ of the robot is obtained by (1).

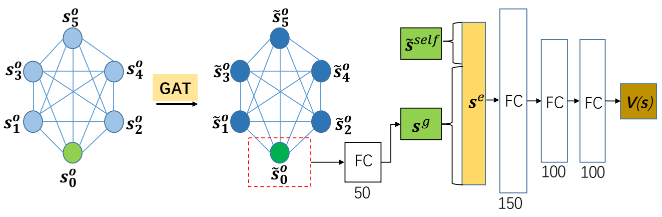

Fig. 3. Network architecture of the proposed method.

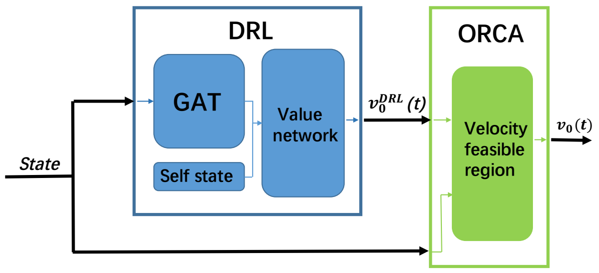

Fig. 4. Illustration of the combined framework of DRL and ORCA

In the training phase, it is difficult to train the value network from absolute random parameters. Therefore, this paper proposes a two-stage training method which includes imitation learning and RL. In the first stage, ORCA is used as a model-guidance method that performs in the environment and generates the demonstration trajectories. Then, the value network is trained by imitation learning on the state-value pairs generated from the demonstration trajectories. After that, RL is implemented based on the first stage work. A behavior policy will be gradually optimized in this stage.

3.3 Combined Framework of DRL and ORCA

The instability of DRL in practical application is inevitable. On the contrary, the ORCA method provides sufficient conditions for collision avoidance in navigation tasks, and it is relatively safe for robot navigation in practical application. Hence, we can combine DRL with ORCA to improve the safety of the robot navigation. The most important benefit of the combination of DRL and ORCA is the combination of short-term benefits and longer-term returns. Since the DRL method can be used to maximize the cumulative reward of the current state, the robot navigating through DRL receives long-term benefits. Meanwhile, ORCA provides the guarantee of collision avoidance in a reaction-based mode when DRL does not work in reality. Metaphorically, DRL is more like a long-term planner, while ORCA is a short-term executor. Therefore, The method combining DRL with ORCA is effective and efficient.

The proposed combined framework of DRL and ORCA is shown in Fig. 4. In each step, first of all, the velocity $\boldsymbol{v}_0^{DRL}(t)$ is obtained through the DRL method proposed in Section 3.2 at time t. After that, $\boldsymbol{v}_0^{DRL}(t)$ is considered as the preferred velocity of ORCA. The robot selects the collision avoidance velocity as the velocity within the permitted velocity space and closest to $\boldsymbol{v}_0^{DRL}(t)$, i.e.,

$$\boldsymbol{v}_0(t) = \underset{\boldsymbol{v} \in ORCA_A^\tau}{argmin} \; ||\boldsymbol{v} - \boldsymbol{v}_0^{DRL}(t)||. \tag{10}$$

where $\boldsymbol{v}_0(t)$ is the final velocity performed of the robot. Therefore, considering the long-term benefits and short-term collision avoidance, the robot finally chooses the suitable velocity $\boldsymbol{v}_0(t)$.

4. SIMULATIONS

4.1 Simulation Settings

The navigation network in this work is implemented with Pytorch in Python. Meanwhile, a simulation environment for robot navigation among external autonomous agents is established in Python. ORCA is used as the policy method of other agents in the simulation environment, and the policy of the robot is our proposed method. In the beginning, other agents are randomly positioned on circle of radius 4 m. Their goals and starting positions are symmetrical about the original point. That is, the simulation environment of training is circle crossing scenarios. The other agents and the robot have holonomic kinematic constraints, which means they can move in any direction.

Two state-of-the-art methods, ORCA (van den Berg et al. (2011)) and LSTM-RL (Everett et al. (2018)), are implemented as baseline methods. In the LSTM-RL method, the states of other agents are fed in reverse order of distance to a LSTM network, meaning that the closest other agent should have the biggest effect on the robot. The main difference between our method and the basic methods is that the features of the environment states are extracted with GAT and the combination of DRL and ORCA is implemented. We refer to the method with GAT as GATRL and the method combining GATRL and ORCA as GATRL-ORCA for the experiment.

4.2 Implementation Specifications

We set that the sizes of the learnable parameters $\boldsymbol{W}$ and $\boldsymbol{a}$ are (7,100) and (200,1) respectively. The GAT inputs the 7-dim states of the agent (4), and outputs a 100-dim feature ($\tilde{\boldsymbol{s}}_i^o$ in Fig. 3). There is six-head attention in attention mechanism ($K = 6$). The activation functions used in value networks are RELU. The time interval of each step is $\Delta t = 0.25$. The threshold of uncomfortable distance is $D = 0.2$. The elements in adjacency matrix $\boldsymbol{A}$ are all one. Meanwhile, in the process of training, when one of the following three conditions is satisfied, the episode will be terminated, i.e.: successful arrival at the goal, collision with other agents and timeout (T = 25) respectively.

4.3 Simulation Results

In order to train a better generalization model, in the training phase, a training simulation experiment is designed to make the robot navigate among five external autonomous agents with decision-making ability in circle crossing scenarios. To fully evaluate the effectiveness and efficiency of the proposed method, we test our proposed method and compare it with baseline methods. The test results are shown in Table 1.

As expected, the GATRL-ORCA method shows good performance in most metrics (success rate, safety level and cumulative return). This is because it not only has the long-term benefits of DRL, but also has the guarantee of short-term collision avoidance through ORCA. On the contrary, both GATRL and LSTMRL have higher failure rate than GATRL-ORCA due to the instability of the DRL method in the new environment and there is no guarantee of collision avoidance. Since they consider the long-term situation, their cumulative returns are relatively high. Then, the ORCA method provides a collision avoidance guarantee in the short term but does not consider the long-term situation, so the cumulative return is relatively low. Meanwhile, Since the safety distance between the robot and other agents is not considered, the duration of discomfort is longer than other methods, and the average minimum distance is also very small. In addition, GATRL is superior to LSTMRL in terms of navigation time and cumulative returns performance. Our analysis shows that the LSTMRL method considers that the closest agent to the robot has the greatest impact on the robot, but does not consider the relationship between the robot and other agents. In contrast, since GATRL uses GAT to extract the influence features of other agents on the robot, the robot can have a deeper understanding of the environment around it. Meanwhile, GATRL-ORCA is outperforming the GATRL in terms of both the cumulative reward and the success rate. Though not by a large margin, this improvement indicate the benefits of combining GATRL with ORCA.

In order to verify the stability and generalization of our method, our method was performed in square crossing scenarios, where the number of other agents increases from 5 to 10. The results are shown in Table 2. The data shows that our method has no failure cases when the number of other agents is 5 in our simulation, and the robot can reach the goal fast and get a high cumulative reward. Meanwhile, there are no collision cases in a more complex environment where the number of other agents is 10. Therefore, our method has good robustness in dynamic and complex environments.

What is more, the proposed method can also be extended to a decentralized multi-agent system without communication. A multi-agent system simulation environment where each agent with local observation uses a common policy is established. Agents share all the learnable parameters including those of graph attention network and value network. At each step, since each agent receives different observations, sharing parameters does not prevent them from behaving differently. Then, agents can avoid collision with each other and reach their respective goals. The test results of simulation is shown in Table. 3. As the number of agents increases, the method also shows a high success rate. Meanwhile, from the analysis of average minimum distance and discomfort frequency data, we can see that they took more cautious actions. Thence, our method framework can be successfully applied to decentralized multi-agent systems without communication.

Table 1. Performance of baseline methods and our method in circle crossing scenarios with five other agents. Discomfort frequency means duration where the robot is too close to other agents. "Average min dis" refers to as average minimum distance between the robot and other agents. Extra time to goal is computed on successful cases. 'Reward' refers to as the average cumulative reward of the robot

Methods	Extra time to goal (s) (Avg / 75th / 90th percentile)	% Discomfort frequency / Average min dis (m) / Reward	% failures(% collisions / % timeout)
ORCA	9.97/10.50/11.00	28/0.07/0.2601	0.0 (0.0 / 0.0)
LSTMRL	11.65/12.25/13.50	6/0.11/0.2662	6.0 (6.0 / 0.0)
GATRL (Ours)	**9.83**/10.75/11.50	11/0.11/**0.3096**	6.0 (6.0 / 0.0)
GATRL-ORCA (ours)	**10.72**/11.50/12.75	10/0.11/**0.3127**	**0.0** (0.0 / 0.0)

Table 2. The test performance of the GATRL-ORCA method (ours) in square crossing scenarios

Number of other agents	Extra time to goal (s) (Avg / 75th / 90th percentile)	% Discomfort frequency / Average min dis (m) / Reward	% failures(% collisions / % timeout)
5	**10.07**/10.75/12.25	12/0.11/0.3317	**0.0** (0.0 / 0.0)
10	**11.20**/12.25/14.75	25/0.11/0.2457	**3.0** (0.0 / 3.0)

Table 3. Navigation performance in decentralized, non-communicating multi-agent system through the GATRL-ORCA method (ours)

Number of agents	Extra time to goal (s) (Avg / 75th / 90th percentile)	% Discomfort frequency / Average min dis (m) / Reward	% failures(% collisions / % timeout)
3	14.08/15.00/16.25	5/0.13/**0.3240**	**7.0** (3.0 / 4.0)
4	15.33/16.50/19.00	8/0.14/**0.3144**	**13.0** (4.0 / 9.0)
5	12.92/14.00/15.75	17/0.13/**0.3457**	**2.0** (2.0 / 0.0)
6	16.16/18.25/19.50	20/0.13/**0.3077**	**15.0** (2.0 / 13.0)

5. CONCLUSION

In this work, GATRL-ORCA is proposed to deal with the problem of robot navigation among external autonomous agents. Specifically, GAT is adopted to describe the robot and other agents as a specific graph, and extract the influence features of other agents on the robot from the graph. The different influence weights of other agents on the robot are computed with multi-head attention mechanism. Furthermore, the combined framework of DRL and ORCA is implemented to enhance the safety of our method in new environments. Our method shows good performance and robustness in dynamic, changeable environments.

6. ACKNOWLEDGMENTS

This work was supported by the National Key Research and Development Program of China under Grant 2018AAA0102402 and 2018AAA0101005, and Innovation Academy for Light-duty Gas Turbine, Chinese Academy of Sciences, No. CXYJJ19-ZD-02.

REFERENCES

Chen, C., Liu, Y., Kreiss, S., and Alahi, A. (2019). Crowd-robot interaction: Crowd-aware robot navigation with attention-based deep reinforcement learning. In *2019 International Conference on Robotics and Automation (ICRA)*, 6015–6022.

Chen, Y.F., Everett, M., Liu, M., and How, J.P. (2017a). Socially aware motion planning with deep reinforcement learning. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 1343–1350.

Chen, Y.F., Liu, M., Everett, M., and How, J.P. (2017b). Decentralized non-communicating multiagent collision avoidance with deep reinforcement learning. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, 285–292.

Everett, M., Chen, Y.F., and How, J.P. (2018). Motion planning among dynamic, decision-making agents with deep reinforcement learning. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 3052–3059.

Fan, T., Long, P., Liu, W., and Pan, J. (2018). Fully distributed multi-robot collision avoidance via deep reinforcement learning for safe and efficient navigation in complex scenarios. *CoRR*, abs/1808.03841.

Fiorini, P. and Shiller, Z. (1998). Motion planning in dynamic environments using velocity obstacles. *The International Journal of Robotics Research*, 17(7), 760–772.

Kuderer, M., Kretzschmar, H., Sprunk, C., and Burgard, W. (2012). Feature-based prediction of trajectories for socially compliant navigation.

Long, P., Fan, T., Liao, X., Liu, W., Zhang, H., and Pan, J. (2018). Towards optimally decentralized multi-robot collision avoidance via deep reinforcement learning. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, 6252–6259.

Long, P., Liu, W., and Pan, J. (2017). Deep-learned collision avoidance policy for distributed multiagent navigation. *IEEE Robotics and Automation Letters*, 2(2), 656–663.

van den Berg, J., Guy, S.J., Lin, M., and Manocha, D. (2011). Reciprocal n-body collision avoidance. In C. Pradalier, R. Siegwart, and G. Hirzinger (eds.), *Robotics Research*, 3–19. Springer Berlin Heidelberg, Berlin, Heidelberg.

Velickovic, P., Cucurull, G., Casanova, A., Romero, A., Liò, P., and Bengio, Y. (2017). Graph attention networks. *ArXiv*, abs/1710.10903.