Abstract—This article investigates the multirobot cooperative navigation problem based on raw visual observations. A fully end-to-end learning framework is presented, which leverages graph neural networks to learn local motion coordination and utilizes deep reinforcement learning to generate visuomotor policy that enables each robot to move to its goal without the need of environment map and global positioning information. Experimental results show that, with a few tens of robots, our approach achieves comparable performance with the state-of-the-art imitation learning-based approaches with bird-view state inputs. We also illustrate our generalizability to crowded and large environments and our scalability to ten times number of the training robots. In addition, we demonstrate that our model trained for multirobot case can also improve the success rate in the single-robot navigation task in unseen environments.

Note to Practitioners—With the development of intelligent industrial and logistic systems, robotic transportation systems are widely implemented. However, existing multirobot path coordination and navigation approaches are basically under some unreasonable assumptions, which are very hard to be implemented in practical scenarios. This article aims to greatly promote the real application of learning-based multirobot cooperative navigation approach, in order to achieve the following. First, we introduce an end-to-end reinforcement learning framework instead of the commonly used imitation learning strategy, as the latter one needs exhaustive training data to cover all the scenarios and does not have the required generalizability. Second, we directly use the raw sensor data instead of the commonly used bird-eye-view semantic observations, as the latter one is generally not representative of practical application scenario from the robot perspective and cannot solve the occlusion issue. Third, we interpret our learned model to illustrate which parts of the input and shared observations contribute most to the robots’ final actions. The above interpretability ensures predictability (thus safety) of our visuomotor policy in practical applications. Our learned visuomotor policy has the ability to coordinate dozens of robots by only using raw visual observations in unknown environments without map nor global localization information, this is the first time in the literature. Our future work includes solving the sim-to-real issue and conducting physical experiments.

Index Terms—Cooperative navigation, multirobot system, reinforcement learning (RL), visuomotor.

I. INTRODUCTION

MULTIROBOT cooperative navigation, which requires each robot to navigate to its goal position autonomously while ensuring motion coordination (i.e., collision avoidance) with other robots, is of great importance in several application scenarios, such as multirobot cooperative transportation [1], multiagent navigation [2], [3], drone and robot swarm formation control [4], [5], and multivehicle platoon control [6], [7]. Traditional approaches include the centralized approach [8], decentralized approach [9], and hierarchical approach [10]. Centralized approach aims at finding out the optimal multirobot spatiotemporal trajectories with as lower computational complexity as possible, and however, it can only be used in a deterministic environment and its scalability to large-scale problems cannot be ensured [10]. The decentralized approach utilizes local traffic rules or motion coordination strategies to achieve flexibility in robot scales, and however, it cannot handle the local crowded space and generalizes poorly to various environments [11]. In order to combine the advantages of the above two categories, the hierarchical approach introduces a high-level planner to optimize overall system performance and utilizes local motion coordination to ensure safety. However, their high computational complexity and the requirement of high-frequency global information storage/retrieval greatly limit their applications.

More recently, learning-based approaches have been investigated to solve the multirobot cooperative navigation problem,
which learns fully distributed local action policies by aggregating local observations and shared information from neighboring robots [3], [12], [13]. Learning-based approaches have achieved promising performances, especially under system uncertainties and dynamics. However, current models are trained on datasets that assume complete state observability of the local environment from the bird-eye view. These are generally not representative of real-world application scenario from the robot perspective and cannot be achieved when onboard sensors are occluded. In addition, they have implicitly divided the raw perception, state estimation (bird-view state observation), and navigation policy learning into separate functional modules, which prohibits any potentially direct feedback between the raw sensor data and the final navigation policy and also cannot be learned in a fully end-to-end manner in practical implementations.

In this article, we utilize the first-person-view raw observations as system input and learn visuomotor policy for multirobot cooperative navigation tasks in unknown environments. Our main contributions can be summarized as follows.

1) To the best of the authors’ knowledge, this article builds the first reinforcement learning (RL) framework for the multirobot cooperative navigation task with first-person-view visual data. The proposed system is fully distributed and only local observations are required, i.e., each robot does not need the environment map nor any global location information. These move a big step to achieve the real implementation in practical applications.

2) A fully end-to-end learning framework is presented to bridge the semantic gap from the raw perception data to the end decision-making. First, deep learning techniques are utilized to extract high-dimensional features from raw visual observations and compact them into efficient representations. Second, graph neural networks (GNNs) are implemented to learn information sharing and aggregation among neighboring robots for efficient local motion coordination. Third, the RL strategy is implemented to generate a proper visuomotor policy for robot control. In contrast to imitation learning (IL)-based works, our approach does not need exhaustive expert data and scales well to unseen scenarios.

3) Comprehensive validation results show that, for a few tens of robots, our RL-based visuomotor policy achieves a comparable performance with the state-of-the-art IL approaches with bird-view state observations. We demonstrate our scalability to large-scale robot networks (ten times number of the training robots) and crowded and large environments (25 times crowdedness of the training case). In addition, we demonstrate that our multirobot visuomotor training can also benefit the single-robot RL navigation ability in unknown environments.

II. BACKGROUND AND RELATED WORK

A. Traditional Approaches

As mentioned above, traditional multirobot cooperative navigation approaches can be classified into centralized, decen-

tralized, and hierarchical approaches. Centralized approaches, represented by the conflict-based search (CBS) [8], ensure completeness and optimality under some certain metrics such as makespan or flowtime [8]. The main shortage of these approaches is that they assume that the deterministic environment, dynamics, and uncertainties in the execution stage will greatly degrade the actual robot navigation performance [10]. Decentralized approaches, represented by the priority-based [14] and time-window-based [15] approaches, introduce motion priority, local traffic rule, or velocity coordination mechanism [9] to solve local conflicts. Decentralized approaches scale well to large robot groups, and however, due to their incompleteness and the need of predefined strategies, their performance declines in crowded local spaces and generalizes poorly to various scenarios [11]. Hierarchical approaches, represented in [10], [16], optimize system-level performance such as the traffic flow and working throughput in a centralized manner and coordinate robots’ spatiotemporal trajectories locally, thus improving the optimality and scalability simultaneously. However, as a centralized information-sharing system is required for high-level scheduling and multiple decentralized local planning centers are typically implemented for coordination [10], both the high system synchronization requirements and the frequent global information storage/retrieval need greatly limit their applications.

B. Learning-Based Approaches

Pioneered by PRIMAL [3], both the IL and RL approaches have been studied for solving the multirobot navigation and motion coordination problems [11]–[13], [17]. Basically, compared to RL-based works, IL-based approaches have advanced successful rate and scale well to hundreds or even thousands of robots [3], [13]. However, they rely on supervised learning with expert planners and exhaustive data that are hard to be obtained, especially to cover all the unexpected scenarios. In order to let the model to explore more potential solutions, RL-based approaches become more and more popular in most recent days, which have better generalizability and are more adaptive to various environments [18]. However, RL-based approaches commonly suffer from the reward sparsity and sample inefficiency issues, especially in the initial training stage. Almost all the existing learning-based approaches assume complete observability of the local environment and input the abstract state observation from the bird-eye view. These are typically not representative of real-world scenarios from the robot’s own perspective. In addition, in order to achieve this, they need to divide the raw perception, state estimation, and navigation policy learning into separate functional submodules with several preprocessing operations and introduce hand-tuned parameters. In this article, we directly utilize raw visual sensor data as the network input and implement a fully end-to-end framework to learn visuomotor policy.

C. Visuomotor Learning for Robot Navigation

Visuomotor policy learning has been investigated for drone navigation [19], mobile manipulation [20], and self-driving
In this article, we consider the 3-D continuous environment space $\mathcal{W} \subseteq \mathbb{R}^3$, which contains a set of $N_s$ static obstacles $\mathcal{S} = \{s_1, \ldots, s_{N_s}\}$ randomly located on the ground. As shown in Fig. 1, a set of $N_r$ mobile robots $\mathcal{R} = \{r_1, \ldots, r_{N_r}\}$ navigate on the ground plane of the free space $\mathcal{W} \setminus \mathcal{S}$. The robot goal positions $\mathcal{G} = \{g_1, \ldots, g_{N_r}\}$ are also randomly located on the ground plane of the free space $\mathcal{W} \setminus \mathcal{S}$. Let $p_i(t)$ denote the position of $r_i$ at time step $t$, and the objective of the multirobot cooperative navigation can be defined as: for a maximum time step $t_m$, $\forall i$, $p_i(t_m) = g_i$, and $\forall i, j, t$, we require $p_i(t), p_j(t) \in \mathcal{W} \setminus \mathcal{S}$ and $p_i(t) \neq p_j(t)$, i.e., each robot $r_i$ should reach its goal position $g_i$ without any collisions with static obstacles or other robots.

In our system formulation, we make the following assumptions. First, each robot $r_i$ can communicate with its neighbors $r_j \in \mathcal{N}_i(t) = \{r_j | D(p_i(t), p_j(t)) < C_r\}$, where $D(\cdot, \cdot)$ represents the Euclidean distance and $C_r$ is the communication range. Second, we do not consider additional dynamic obstacles except for the moving robots in the environment. Third, each robot can obtain rough direction information of its goal during movements. Please note that here, we do not assume the specific coordinates (exact location) of the goal relative to the robot. In addition, except for this direction information, we do not need any other global location nor relative distance information of the goal and neighboring robots, also we do not need the global environment map. This aims to ensure the implementation of our multirobot cooperative navigation approach in any unseen environments. Fourth, the robots have no orientation and their visual observations are omnidirectional, and the coordinate system of each robot is defined in the bottom right of Fig. 1(a). The omnidirectional observation and movement assumptions aim to facilitate each robot to learn the relative positioning information of the communicated neighbors as well as their motion intentions. Note that this assumption can be easily alleviated in practical implementations by using the “rotate-move-rotate” strategy in each moving step of each robot, and only a compress sensor is required.

As shown in Fig. 1(b), at each time step $t$, each robot $r_i$ obtains the local observation $O_i^j$, which contains four input channels, the first three channels are the omnidirectional RGB image of its surrounding environment with the size of $384 \times 54$, and the last channel is a virtual image with the same size that contains the goal direction information. Similar to the omnidirectional image, the leftmost and rightmost columns in the goal channel correspond to $0^\circ$ and $360^\circ$ in the coordinates system defined in Fig. 1(a), and then, the column (with a width of 20) corresponded to the goal direction $\phi_i$ is highlighted with the white color and all the other columns are with the black color.

We consider the velocity control of the robot in the continuous environment space. As shown in Fig. 1(a), the robot action set is defined as $A = \{l_{f}, v_{b}, l_{d}, b_{r}, v_{g}\}$, i.e., at each time step $t$, each robot can move front, back, left, or right with a constant velocity or stay in its current location.

Based on the above formulations, we now build our multirobot cooperative navigation framework. For each robot, our end-to-end learning framework includes three main modules as follows.

1) Observation Feature Learning: As shown in Fig. 2, the input tensor of each robot’s network is a time series of the multichannel observations $\{O_i^j, O_{i-1}^j, \ldots, O_{i-N_r}^j\}$. Then, we utilize a series of convolution layers followed by a fully connected (FC) layer to extract...
the motion intentions of more distant robots in advance to plan its future motions and avoid potential future collisions. Fortunately, through multihop routing [26] or multiround information sharing [13], each robot can obtain its multihop neighbors’ information while still maintaining the fully distributed framework. In order to balance the complexity and performance, we consider two-hop neighbors’ information in local coordination information learning. We define \( \Psi_e = \{ \psi_{ij}^{1} \} \in \mathbb{R}^{N_r \times N_r} \) as the adjacency matrix of the extended communication graph, i.e., for three robots \( r_i, r_j \) and \( r_k \), if \( r_i \in \mathcal{N}_e(t) \) and \( r_j \in \mathcal{N}_e(r_k) \) simultaneously but \( r_j \not\in \mathcal{N}_e(t) \), then \( \psi_{ij}^{1} = \psi_{jk}^{1} = 1 \), otherwise \( \psi_{ij}^{1} = 0 \). We further define \( \Psi_d = \{ \psi_{ij}^{d} \} = \Psi_d + I_N, \Psi_e = \{ \psi_{ij}^{e} \} = \Psi_e + I_N \), which introduce the self-loop information.

As shown in Fig. 3, the local coordination information learning module inputs each robot’s observation features \( F_c \) extracted by the observation feature learning module and two GNN layers are implemented to aggregate neighbors’ features. More specifically, the input feature matrix of the first GNN layer is \( H^{(0)} = [F_c^{1}, \ldots, F_c^{N_r}]^{T} \), where the \( i \)-th row is the feature vector of the robot \( r_i \). The output of the first GNN layer is

\[
H^{(1)} = \left[ \sigma \left( F_{d}(H^{(0)}, \Psi_d, W_d^{(1)}) \right), \sigma \left( F_{e}(H^{(0)}, \Psi_e, W_e^{(1)}) \right) \right]
\]

where \( F_{d}(\cdot, \cdot, \cdot) \) and \( F_{e}(\cdot, \cdot, \cdot) \) represent the graph convolution operation on the direct communication graph \( \Psi_d \) and extended communication graph \( \Psi_e \), respectively, \( W_d^{(1)} \) and \( W_e^{(1)} \) represent learnable weight matrices, and \( \sigma(\cdot) \) represents the elementwise nonlinear activation function. Then, we introduce the second GNN layer to fully exploit the feature aggregation and extraction ability of GNN and the final output of the local coordination information learning module is

\[
H^{(2)} = \left[ \sigma \left( F_{d}(H^{(1)}, \Psi_d, W_d^{(2)}) \right), \sigma \left( F_{e}(H^{(1)}, \Psi_e, W_e^{(2)}) \right) \right] = [F_G^{1}; F_G^{2}; \ldots; F_G^{N_r}]^{T}
\]

where \( F_G \) represents the feature vector of robot \( r_i \).

In this article, we utilize the following two kinds of GNN to formulate the graph convolution operation \( F_G(H^{*}, \Psi_{e}, W_{e}^{*}) \) in each graph convolution layer (where \( \sigma \) can be \( \cdot \) or \( d \), \( \cdot \) can be \( 0 \) or \( 1 \), and \( \ast \) can be \( 1 \) or \( 2 \)).

1) **Graph Convolution Network (GCN):** Define the diagonal degree matrix \( M_{d} = \{ m_{ii}^{d} \} \in \mathbb{R}^{N_r \times N_r} \), where \( m_{ii}^{d} = \sum_{j \in \mathcal{N}_d(t)} \psi_{ij}^{d} \), and then, GCN has the following formulation:

\[
F_{d}(H^{*}, \Psi_{d}, W_{d}^{*}) = M_{d}^{-\frac{1}{2}} \Psi_{d} M_{d}^{-\frac{1}{2}} H^{*} W_{d}^{*}
\]

Normalizing the adjacency matrix by using the diagonal degree matrix contributes to unify the scale of each robot’s feature vector. The above GCN is based on the graph Fourier transform theory and is well simplified by using the first-order Chebyshev polynomial approximation, which can prevent the gradient vanishing or exploding problem.

2) **Graph Attention Network (GAT):** We compute a convolution similar to GCN introduced above but use the attention mechanism [27] to weight the adjacency matrix instead of using the normalized adjacency matrix

\[
F_{e}(H^{*}, \Psi_{e}, W_{e}^{*}) = \alpha_{e} H^{*} W_{e}^{*} + b_{e}^{*}
\]

In the following, we will describe our coordination information learning and visuo-motor policy learning modules.

IV. LOCAL COORDINATION INFORMATION LEARNING

In order to achieve cooperative navigation, the learned observation feature of each robot can be shared locally among neighboring robots. Sharing the observation features helps the robots to learn their relative positions and enables the robots to extend perception range by fusing neighbors’ observation features. In addition, as the goal direction information is also contained in the observation features, each robot can also obtain the moving intentions of its neighbors, thus facilitating the learning of local motion coordination and conflict avoidance. In this article, we introduce the GNN module to achieve the above local coordination information learning.

We first define \( \Psi_d = \{ \psi_{ij}^{d} \} \in \mathbb{R}^{N_r \times N_r} \) as the adjacency matrix, which describes the neighboring relations, i.e., if robot \( r_j \in \mathcal{N}_e(t) \), then \( \psi_{ij}^{d} = \psi_{ji}^{d} = 1 \); otherwise \( \psi_{ij}^{d} = 0 \) (here, the subscript \( d \) represents the direct communication). In our preliminary testings, we find that only aggregating the information from the direct neighbors is not adequate for local motion coordination, as each robot should consider high-dimensional features \( F_c \) from the input tensor. For simplification, we remove the time dimension \( t \) here.

2) **Local Coordination Information Learning:** Inspired by [13], [25], we share the learned observation features among neighboring robots and introduce GNN layers to aggregate the shared messages for learning the local coordination information. For each robot \( r_i \), a high-dimensional coordination feature \( F_c \) is obtained by aggregating the visual observation as well as motion intention of local neighbors.

3) **Visuo-motor Policy Learning:** Finally, we concatenate \( F_d \) and \( F_c \) to form the input of visuo-motor policy learning module and use a series of FC layers to estimate the value function and the quality of each state-action pair. The action with the best quality will be chosen with the highest probability.

Our main principle in the above system design is to keep our network structure as parameter economic as possible, in order to make the network easy training and ensure the real-time performance. Please note that, if necessary, one can always introduce more complex network structures in each of the above three modules to further improve model capacity and performance.

Fig. 2. Network structure of the observation feature learning module, where we input time series observations and utilize convolution layers to extract high-dimensional features for the input observation of each robot.

Convolution layer: BN: batch normalization. FC: fully connected layer.
The reward function $\Upsilon$ aims to integrate each robot’s own visual observation and the neighbors’ observation at the same time, thus helping the robot to navigate to its goal, i.e., $\Upsilon_{ij} = -\gamma_1 (N_m) < 0$ if the robot arrives its goal, i.e., $\gamma_1 (N_m) > 0$.

4) An extra-large positive reward $\gamma_4$ if the robot arrives its goal, i.e., $\Upsilon_{ij} = -(\gamma_1 (N_m) + \gamma_4 > 0$.

The optimal path can be calculated by using the Dijkstra or A* method. An implementation problem is that we consider continuous environment space in this article, and however, the optimal path does not need to be carried out in the discrete environment with topology graphs. In order to address this issue, we temporally discretize the free space $W \setminus S$ to obtain a grid topology map and the discretization step is set to one unit length. During training, the current and target position of each robot will be approximated to the nearest nodes in the discretized map, based on which the optimal path will be generated. Introducing the optimal path information aims to provide each robot a dense reward at each step. Please note that the optimal path information is only used in the training.

In addition, in order to decrease the computational complexity, an event-triggered strategy is introduced, i.e., we only update the optimal path when the robot’s position in the discretized map changes in this time step.

**B. Curriculum Learning Strategy**

Although we provide a dense reward function for each robot, the model training of multirobot cooperative navigation is still very challenging, as we build a fully end-to-end RL framework with raw visual inputs. In this article, we utilize the curriculum learning strategy to increase sample efficiency and accelerate the training speed. More specifically, we first train our model in a simple and structured environment [as shown in Fig. 4(a)] and then transfer the trained network to a more complex and clutter environment [as shown in Fig. 4(b)] for further training. Both of the environments are with the size of $60 \times 60$ and contain five robots. The obstacle density in the second environment is about 1%.

We first use the simple and structured environment shown in Fig. 4(a), and the robots’ initial and goal positions are generated more frequently in the open area located at the center of the environment so that robots can learn the visual characteristics of the goal more easily during the early training.
stage. At the same time, a few robots’ goal positions will also appear behind the obstacles, which allows the robots to have some basic obstacle-avoidance capabilities. The first stage training will automatically stop when the following conditions are met.

1) Reward curve is stable (evaluated by its variance) for more than 100 thousand steps.
2) Average episode reward is above \( N_e \) for more than 100 thousand steps.
3) The average number of collisions in the past five rounds is less than \( N_c \) for each robot.

We then transform the trained model into a complex and crowded environment, as shown in Fig. 4(b). The robot cannot always see its goal directly at the initial position, and the distance from the robot’s initial location to the goal is also enlarged. Due to the preliminary navigation and obstacle-avoidance capabilities learned in the previous stage, the robots can navigate to their goal location in the complex environment, but with an unskilled manner (large detours). The second stage of training can further strengthen the robots’ navigation and obstacle-avoidance abilities and shorten the time of goal searching in the complex environment.

VI. IMPLEMENTATION

A. Model Parameters

We use proximal policy optimization [28] as the RL algorithm in our implementation, which is an actor–critic method based on a combination of policy and value gradient. The default parameters are set as follows: the reward parameters \( \gamma_1 = 10, \gamma_2 = 1, \gamma_3 = 20, \gamma_4 = 10, N_v = 20, N_r = 3, \) and \( N_m = 400 \). The observation feature learning module is described in Table I with a flatten operation and an FC layer with 512 unit. The activation functions are described in Table I followed with a flatten operation and an FC layer with 512 unit. The activation functions are ReLU.

For the local coordination information learning module, \( F_i^G \in \mathbb{R}^{512}, F_i^d \in \mathbb{R}^{512}, H^{(0)}, H^{(1)}, H^{(2)} \in \mathbb{R}^{N \times 512}, W^{(1)}_d, W^{(2)}_d \in \mathbb{R}^{512 \times 256}, \) and \( W^{(1)}_v, W^{(2)}_v \in \mathbb{R}^{512 \times 256} \). In the visuomotor policy learning module, we use three FC layers with 512, 256, and 64 units, respectively.

B. Training and Testing

We train our model with Intel i7-8700K CPU and one NVIDIA GTX 1080Ti GPU, Python 3.7 with TensorFlow 2.3. The initial learning rate is \( 4 \times 10^{-3} \) and will linearly decay to \( 4 \times 10^{-6} \) in one million steps. The training optimizer is Adam. We use curriculum learning strategy to help robots converge to a reasonable strategy faster, which has been explained in detail in Section V-B. During the training process, episode switching conditions include: 1) all agents reach their own goal and 2) simulation step of current episode reaches a maximum number defined as 300, which is smaller than maximum simulation steps \( N_m \) in testing, as we expect the robot to collect more representative data and filter out invalid and failed cases during training. If one of the above conditions is satisfied, we will end the current episode and start a new one. In training, the static obstacles in the environments shown in Fig. 4 are fixed in different episodes, but the robots’ initial and goal locations are changed.

We introduce a different number of robots (1, 5, 10, 30, and 50), static obstacle densities (4%, 8%, and 12%), and environment sizes (40 × 40, 60 × 60, and 100 × 100) to build the various of testing environments. We will comprehensively test the generalizability and scalability of different model structures in different environments, which will be introduced in detail in the following. The following metrics are used for performance evaluation.

1) Success Rate: As mentioned before, we set a maximum step \( N_m = 400 \) for all the tests and count the number \( N_{ra} \) of the robots that reach their goal within this time. Then, we calculate the successful rate \( (N_{ra}/N_r) \) of each evaluation.

2) Path Length:

\[
\text{Path Length} = \frac{1}{N_{ra}} \sum_{i=1}^{N_{ra}} \frac{\mathcal{L}_i}{\mathcal{L}_{\text{opt}}}
\]  

where \( \mathcal{L}_i \) is the actual moving distance of each robot \( r_i \) and \( \mathcal{L}_{\text{opt}} \) is the length of the optimal path between the robot’s initial position and goal, which is defined in Section V-A. Please note that in the multirobot path planning case, the shortest path \( \mathcal{L}_{\text{opt}} \) typically cannot be achieved, even for the optimal solution, each robot should also take a detour or temporarily stop for locally coordinating its motion with other robots.

3) Moving Step: This measure corresponds to the average moving steps of the robots that arrive their goal successfully. For those robots which cannot reach their goal, we use the maximum time step \( N_m \).

C. Baseline Models

In the results, we consider the following five different network structures and compare their performance for ablation study.

---

<table>
<thead>
<tr>
<th>TABLE I</th>
<th>NETWORK STRUCTURE OF THE OBSERVATION FEATURE LEARNING MODULE</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>kernel size</td>
</tr>
<tr>
<td>Cov1</td>
<td>(3,5)</td>
</tr>
<tr>
<td>Maxpool1</td>
<td>(2,2)</td>
</tr>
<tr>
<td>Cov2</td>
<td>(4,4)</td>
</tr>
<tr>
<td>Maxpool2</td>
<td>(2,2)</td>
</tr>
<tr>
<td>Cov3</td>
<td>(4,4)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>TABLE II</th>
<th>COMPUTATIONAL COMPLEXITY OF EACH BASELINE MODEL</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Parameters</td>
</tr>
<tr>
<td>SRV</td>
<td>14,086,896</td>
</tr>
<tr>
<td>MRV-C | 14,611,184</td>
<td>32.796ms</td>
</tr>
<tr>
<td>SC</td>
<td>14,614,236</td>
</tr>
<tr>
<td>MRV-A | 14,873,328</td>
<td>33.154ms</td>
</tr>
<tr>
<td>SC</td>
<td>14,875,376</td>
</tr>
</tbody>
</table>
1) **MRV-C**: This is our complete model, which utilizes GCN described in Section IV for the local coordination information learning module. Here, MRV represents multirobot visual-based cooperative navigation and C represents GCN.

2) **MRV-A**: This is our complete model, which utilizes GAT described in Section IV for the local coordination information learning module. Here, A represents GAT.

3) **MRV-C \( w. SC \)**: In this model, we utilize GCN in the local coordination information learning module and directly input the learned feature vector \( F^C_i \) to the visuomotor policy learning module (do not concatenate \( F^G_i \) with the observation feature \( F^C_i \)). Here, \( w. SC \) represents “without the skip connection of the observation feature \( F^C_i \).”

4) **MRV-A \( w. SC \)**: In this model, we utilize GAT in the local coordination information learning module and directly input the learned feature vector \( F^G_i \) to the visuomotor policy learning module without concatenating the observation feature \( F^C_i \).”

5) **SRV**: In this model, each robot independently learns its visuomotor policy without communicating with others (treats other robots as noncommunicative dynamic obstacles), i.e., we directly input the observation feature vector \( F^C_i \) to the visuomotor policy learning module.

Here, SRV represents the single-robot visual navigation.

In Table II, we show the number of parameters and the real runtime per step of each baseline model mentioned above. The results show that most of the parameters (and also the spent computation time) are in the observation feature learning module. This implies that introducing the local coordination feature learning module does not largely increase the storage and computation complexity, and however, as will be shown in Section VII, it will largely improve the performance.

### VII. Results

In this section, we conduct comprehensive validations and comparisons to demonstrate the effectiveness, generalizability, and scalability of our approach. All the validations are conducted in Unity using ML-Agents toolkit [29].

#### A. Convergence Performance in Training

As mentioned in Section V-B, we introduce the curriculum learning strategy to increase the sample efficiency, especially in the initial training period. The cumulative reward of each episode in each training stage is shown in Fig. 5(a), where the dotted line indicates the actual switch time in the training of each model. The average moving step of each model in each episode is shown in Fig. 5(b). Please note that in training, we set the time-out moving step as 300 and will terminate this iteration at this time if any robot cannot reach its goal within 300 steps.

The training curves in Fig. 5 show that the following conditions hold.

1) Introducing the skip connection (i.e., the concatenation of \( F^C_i \) and \( F^G_i \)) in the input of the visuomotor policy learning module greatly improves the convergence speed in the first training stage. In the presence of skip connection, the GNN module can only focus on the efficient feature sharing and aggregation task, while it does not need to simultaneously learn to integrate the visual features extracted by the observation feature learning module. Fig. 5(a) shows that MRV-A \( w. SC \) even cannot converge in the first training stage, which results from the great challenges of learning the perception, attention policy, information aggregation, and visuomotor control modules simultaneously. Please note that each robot does not know what its goal looks like (in our simulator, the goal of all the robots looks the same) and it can only be learned after the robot receives the responding reward, and this also increases the difficulty of the initial learning stage.

2) SRV, MRV-C, and MRV-A converge quickly in the initial stage. The introduced goal direction information and the proposed dense reward help to increase the sample efficiency during the initial training stages.

3) The switch times of MRV-C and MRV-A are earlier than that of SRV. In addition, MRV-C performs better than MRV-A as the latter one has more learnable parameters.

4) SRV has the best performance only in the very early training stage, and the potential reason is that it has less learnable parameters and obtains more rewards in simple tasks in the initial training stage. However, as it is very hard for SRV to learn the motion coordination ability, its overall convergence speed is slower than MRV-C and MRV-A.

#### B. Evaluation Results

In this section, we test different networks on testing environments that have the same map size and robot number with the training environments, but the testing environments have a larger obstacle density (about 4%). The results in Table III
show that the following conditions hold. First, both the success rates of MRV-C and MRV-A are higher than 95%, which are similar to the results obtained by the state-of-the-art imitation learning-based approach [13]. Note that we directly input the first-person-view visual observations and do not need an expert dataset for supervised learning, and thus, our approach is more promising in real applications. Second, MRV-A performs better than MRV-C since it has a very similar success rate but much lower path length and moving step.

### C. Scalability to Large-Scale Robot Networks

In this section, we directly use different models trained with five robots to test their scalability to large-scale problems with 10, 30, and 50 robots. The testing environments have the same map size ($60 \times 60$) with the training maps but have a larger obstacle density (about 4%). Combining the results in Tables III and IV, we can find that the following conditions hold.

1) By introducing the GNN with skip connection, success rate can be largely improved in all the cases. Even for the 50 robots case (ten times the training robot number), MRV-C still has about 78% success rate. In addition, the performance degradation of MRV-C and MRV-A in large-scale problems is also lower than SRV.

2) The performance of MRV-C \( \setminus w. \ SC \) declines largely with the increasing robot number and is even much worse than SRV. This indicates that, without skip connection, the model cannot learn how to efficiently aggregate the self-observation features and the shared messages from others. In large-scale problems, each robot receives a large number of shared messages, which will dilute the perception signal of the robot itself, so without an independent perception signal channel, the navigation performance in complex tasks cannot be maintained.

3) Comparing MRV-C with MRV-A, the former one has the higher success rate, whereas the latter one has the lower path length and moving step. This indicates that MRV-A has better local coordination performance. In addition, comparing SRV with MRV-C, we can find that the former one has the slightly lower path length but much smaller success rate (note that we only consider successful cases in path length), which implies that when local motion conflict occurs, SRV tends to wait in place, while MRV-C has the stronger ability to avoid conflicts by introducing local path detours.

### D. Generalizability to Crowded and Large Environments

In this section, we first test different models in the $60 \times 60$ map with ten robots, and we increase the static obstacle density to 8% and 12%. The results in Table V show that: 1) compared with SRV, the success rate degradation of MRV-C and MRV-A in crowded environments are much lower; 2) MRV-C performs better in success rates, while MRV-A has the shortest path length and moving step; and 3) comparing Tables IV and V, the path length of all the models remains basically unchanged, but moving step increases with the increase of the static density. This implies that, in the crowded environments, each robot is more likely to wait in place for local motion coordination.

We then test different models in an extremely crowded environment, i.e., the $40 \times 40$ map with 12% obstacle density and 50 robots (about 25 times crowdedness than the training case). In this environment, the multirobot cooperative navigation task becomes very challenging as each robot should handle frequent motion conflicts and the free space for local motion coordination is also very limited. From the results shown in Fig. 6(a), we can find that our approaches still have a success rate higher than 50%, which is much better than SRV. The differences between MRV-C and MRV-A in both metrics are not obvious. The potential reason is that, since in such an environment the space is very crowded, introducing attention...
cannot largely improve model ability. In order to verify this claim, we further test different models in a large environment, i.e., the $100 \times 100$ map with the same 12% obstacle density and 50 robots. The results in Fig. 6(b) show that MRV-A outperforms MRV-C as there are enough local coordination spaces, and thus, introducing attention will enable each robot to find out a more efficient local path. In addition, the variance of the density distribution of MRV-C and MRV-A in the large environment is also lower than that in the extremely crowded environment.

**E. Interpretation Results**

As shown in Fig. 7, we calculate the gradient of original input observations on the final robot actions and visualize the importance (heat value) of each pixel. The results answer the following questions.

1) Which Parts of Each Robot’s Own Observation Contribute Most to Its Final Action: As an example, Fig. 7(d) visualizes the contribution of each pixel in the original input of robot 3 to its own final action, which shows that the area corresponded to the goal direction contributes most. In addition, the directions that correspond to the movable paths are also highlighted (note that we only mark directions 1–4 as examples, and other highlighted areas also correspond to movable paths). The above results illustrate that the perception module of each robot has successfully learned the...
Contribute Most to the Robot’s Final Action:

1) Comparing \( \text{SRV} \) and \( \text{MRV-C} \), we can confirm that training in the multirobot case does improve the single-robot navigation performance, even in this case, there is not any shared messages in the testing. This implies that training in the multirobot case will improve the obstacle-avoidance ability and help robots to handle more complex scenarios. The success rates of \( \text{MRV-C} \) under both the 40 \( \times \) 40 and 60 \( \times \) 60 maps with 4\% obstacle density are 99.33\%, which means that there is only one failure case in the 150 tests.

2) With the increasing map size and obstacle density, the robot needs more moving steps to reach its goal, so the convergence speed of success rate decreases. In addition, the final success rate at the time-out step also decreases as the robot needs to travel a much longer distance while avoiding more obstacles. The results show that the degradation of success rate of \( \text{MRV-C} \) is much lower than \( \text{SRV} \).

G. Discussion

Based on the above results, we would like to provide a preliminary discussion of the current ability boundary of the IL-based versus RL-based models, and the first-person-view versus bird-eye-view approaches. We choose the state-of-the-art approaches [3], [13] for comparison, which inputs the abstract state observation from the bird-eye view. The IL-based GAT is utilized in [13], while the IL and RL combined approach is used in [3]. As our system input and model framework are totally different from those in [3], [13], we only provide a rough comparison on the success rate.

Li et al. [13] said that their models trained with ten robots can maintain the success rate above 92\% even as they increase the robot number from 10 to 60 on the 50 \( \times \) 50 map. In [3], success rate on the 80 \( \times \) 80 map with 64 robots is higher than 95\%. In our experiments, success rate in 60 \( \times \) 60 map is 95.87\% for five robots, 94.27\% for ten robots, and 87.13\% for 30 robots. The above results show that, in small-scale problems, our model achieves comparable performance with the IL-based approaches with bird-view state inputs. However, we are also noticed that the scalability of our first-person-view RL-based approach is worse than the IL-based approaches with bird-view state inputs, as our success rate drops more quickly in large-scale instances. In our testings, the maximum robot number is set to 50 due to the hardware limitation and the obtained success rate is about 77.92\% on the 60 \( \times \) 60 map; however, in [13], the success rate is above 80\% even on the 200 \( \times \) 200 map with 1000 robots. The results in [3] also show that the success rate is above 80\% on the 80 \( \times \) 80 map with 256 robots. A potential reason is that in the crowded environments with more robots, the raw visual observations are much more complex. As we only train the model with five robots, it is very hard for the perception network to handle the unseen complex observations. In addition, without any expert data, it is more challenging for RL model to learn the motion coordination policy.

F. Single-Robot Navigation

In this section, we test different models in the single-robot visual navigation task to see whether training in the multirobot case will also improve the single-robot navigation performance. As in this case, \( \text{MRV-C} \) and \( \text{MRV-A} \) are exactly the same, so we only compare \( \text{SRV} \) with \( \text{MRV-C} \) for evaluation. As shown in Fig. 8, different map sizes with different obstacle densities are considered and the success rate (proportion of destination-arrived robots) at each step of each model is plotted. The results show that the following conditions hold.

Fig. 8. Singe-robot navigation results. The left, middle, and right figures plot the results of 40 \( \times \) 40 map, 60 \( \times \) 60 map, and 100 \( \times \) 100 map, respectively. In each map, different static obstacle densities, i.e., 4\%, 8\%, and 12\%, are considered.

Based on the above results, we would like to provide a preliminary discussion of the current ability boundary of the IL-based versus RL-based models, and the first-person-view versus bird-eye-view approaches. We choose the state-of-the-art approaches [3], [13] for comparison, which inputs the abstract state observation from the bird-eye view. The IL-based GAT is utilized in [13], while the IL and RL combined approach is used in [3]. As our system input and model framework are totally different from those in [3], [13], we only provide a rough comparison on the success rate.

Li et al. [13] said that their models trained with ten robots can maintain the success rate above 92\% even as they increase the robot number from 10 to 60 on the 50 \( \times \) 50 map. In [3], success rate on the 80 \( \times \) 80 map with 64 robots is higher than 95\%. In our experiments, success rate in 60 \( \times \) 60 map is 95.87\% for five robots, 94.27\% for ten robots, and 87.13\% for 30 robots. The above results show that, in small-scale problems, our model achieves comparable performance with the IL-based approaches with bird-view state inputs. However, we are also noticed that the scalability of our first-person-view RL-based approach is worse than the IL-based approaches with bird-view state inputs, as our success rate drops more quickly in large-scale instances. In our testings, the maximum robot number is set to 50 due to the hardware limitation and the obtained success rate is about 77.92\% on the 60 \( \times \) 60 map; however, in [13], the success rate is above 80\% even on the 200 \( \times \) 200 map with 1000 robots. The results in [3] also show that the success rate is above 80\% on the 80 \( \times \) 80 map with 256 robots. A potential reason is that in the crowded environments with more robots, the raw visual observations are much more complex. As we only train the model with five robots, it is very hard for the perception network to handle the unseen complex observations. In addition, without any expert data, it is more challenging for RL model to learn the motion coordination policy.
VIII. Conclusion and Future Work

In this article, we build the visuomotor learning system for multirobot cooperative navigation using RL. The proposed approach is fully distributed and end-to-end, which can be implemented in unknown environments. Comprehensive results demonstrate the effectiveness, scalability, and generalizability of our models. The obtained results are very promising and move forward a large step to practical applications of multirobot cooperative navigation in industrial and logistic scenarios. Due to hardware limitations, we only test our models with no more than 50 robots in this article, as the raw visual data are much storage and computational costly compared with the state observation data used in the existing works. In the future, we will try to further improve our models’ ability in solving large-scale problems with hundreds of robots and also take the domain transfer issue into consideration to conduct physical robotic experiments.

Acknowledgment

The authors would like to thank Binyu Wang and Qingbiao Li for their helpful discussions. They would also like to thank NVIDIA Corporation for their supports of GPUs.

References


Qiming Liu received the B.Eng. degree in automation from Shanghai Jiao Tong University, Shanghai, China, in 2020, where he is currently pursuing the M.S. degree in control science and engineering. His current research interests include robot navigation, multiagent coordinated control, and reinforcement learning.

Ling Tang is currently pursuing the bachelor’s degree (Hons.) in automation with Shanghai Jiao Tong University, Shanghai, China. His current research interests include robot navigation and multirobot cooperation.

Kefan Jin received the B.Eng. degree in ocean engineering from Harbin Engineering University, Harbin, China, in 2016. He is currently pursuing the Ph.D. degree in ocean engineering with Shanghai Jiao Tong University, Shanghai, China. His current research interests include multiagent cooperation, unmanned surface vehicle, and autonomous driving systems.

Hongye Wang received the B.S. degree in detection, guidance and control technology from Northwestern Polytechnical University, Xi’an, China, in 2020. He is currently pursuing the master’s degree with the Department of Automation, Shanghai Jiao Tong University, Shanghai, China. His research interests include autonomous mobile robots, multirobot cooperation, and autonomous driving systems.

Ming Liu (Senior Member, IEEE) received the B.A. degree in automation from Tongji University, Shanghai, China, in 2005, and the Ph.D. degree from the Department of Mechanical Engineering and Process Engineering, ETH Zürich, Zürich, Switzerland, in 2013. He was a Visiting Scholar with Erlangen Nurnberg University, Erlangen, Germany, and the Fraunhofer Institute IISB, Erlangen. He is currently an Assistant Professor with the Department of Electronic and Computer Engineering, The Hong Kong University of Science and Technology, Hong Kong. His current research interests include autonomous mapping, visual navigation, topological mapping, and environment modeling.

Dr. Liu was a recipient of the Best Student Paper Award from IEEE MFI 2012, the Best Paper in Information Award from IEEE ICIA 2013, the Best RoboCup Paper from IEEE IROS 2013, the Best Conference Paper Award from IEEE CYBER 2015, the Best Student Paper Award from IEEE ICAR 2017, and the Best Paper in Automation Award from IEEE ICIA 2017. He received twice the innovation contest Chunhai Cup Winning Award in 2012 and 2013. He received the Wu Weijun AI Award in 2016 and the IROS Toshio Fukuda Young Professional Award in 2018. He serves on the Editorial Board of IEEE Robotics and Automation Letters.

Hesheng Wang (Senior Member, IEEE) received the B.Eng. degree in electrical engineering from Harbin Institute of Technology, Harbin, China, in 2002, and the M.Phil. and Ph.D. degrees in automation and computer-aided engineering from The Chinese University of Hong Kong, Hong Kong, in 2004 and 2007, respectively. He is currently a Professor with the Department of Automation, Shanghai Jiao Tong University, Shanghai, China. His current research interests include visual servoing, service robots, computer vision, and autonomous driving.

Dr. Wang was the General Chair of the IEEE RCAR 2016 and the Program Chair of the IEEE ROBIO 2014 and IEEE/ASME AIM 2019. He is the General Chair of IEEE ROBIO 2022. He served as an Associate Editor for the IEEE TRANSACTIONS ON ROBOTICS from 2015 to 2019. He is also an Associate Editor of IEEE TRANSACTIONS ON AUTOMATION SCIENCE AND ENGINEERING, IEEE Robotics and Automation Letters, Assembly Automation, and the International Journal of Humanoid Robotics, a Technical Editor of the IEEE/ASME TRANSACTIONS ON MECHATRONICS, and an Editor of the Conference Editorial Board (CEB) of IEEE Robotics and Automation Society.