2026 IEEE International Conference on Robotics and Automation (ICRA 2026)
June 1-5, 2026. Vienna, Austria

Decentralized Triangulation Formation Without Communication: A Vision Transformer Based Learning Approach

Xinchi Huang, Guang Yang, and Yi Guo

Abstract— **Multi-robot cooperative control has traditionally relied on model-based distributed methods, but separating perception and control in such pipelines often introduces processing delays and error accumulation. This paper presents a novel decentralized control strategy for multi-robot triangulation formation using Vision Transformers (ViTs). Unlike existing methods that rely on robot-to-robot communication or centralized coordination, the proposed approach learns an end-to-end control policy that scales to an arbitrary number of robots, solely using on-board sensor data. By segmenting LiDAR-based occupancy maps into patches and processing them as sequences, the ViT encoder captures spatial relationships in the environment. A subsequent multi-layer perceptron outputs control commands that drive each robot to form a planar triangulation with prescribed inter-robot distances—all without explicit communication among robots. The learned policy is validated in both simulations and real-world experiments on a group of RoboMaster platforms. Experimental results demonstrate robust and scalable formation performance across diverse conditions.**

I. Introduction

The last decade has witnessed substantial technological advances in multi-robot systems, enabling a vast range of applications, including autonomous transportation systems, multi-robot exploration, rescue, and security patrols. Multi-robot systems demonstrated notable advantages over single-robot systems, such as enhanced efficiency in task execution, reconfigurability, and fault tolerance. Particularly, the capability to self-organize via local interaction gives rise to various multi-robot collective behaviors (e.g., flocking, formation, area coverage) to achieve team-level objectives [1]–[3].

Multi-robot learning has long been an active research area [4], [5]. Nonetheless, it isn't until recent years that challenges originating from real-world complexities could be handled with the advent of deep reinforcement learning techniques. Various multi-robot control problems such as learning to communicate [6]–[8], path planning [9]–[12], coordinated control [13]–[19] have been tackled using learning-based methods. Despite the remarkable progress in multi-robot learning, the architecture design and learning of scalable computational models that accommodate emerging information structures is still an open questions.

Vision Transformer (ViT) models have gained prominence in the field of computer vision, setting new benchmarks as state-of-the-art models across a variety of tasks such as image classification [20], semantic segmentation [21], and object detection [22]. ViTs segment images into patches and process these patches as sequences. This method allows ViTs to grasp the full context of an image. With a self-attention mechanism, ViTs can prioritize the most relevant segmentation of an image, which is suitable for handling occupancy maps where meaningful pixels are interspersed among vast areas of irrelevant data.

In this paper, we study a multi-robot formation control problem using a ViT-based learning approach, which aims to find decentralized control policies that operate on robot sensor observations without robot-to-robot communication. The formation is defined for the multi-robot team to achieve triangulation of neighboring robots that constitute a planar graph with prescribed equidistant edge lengths. We demonstrate that the learned decentralized control policy is scalable to varying sizes of multi-robot teams while trained with a fixed number of robots in a robot simulator. Real robot experiments demonstrate the effectiveness of the developed method.

The main contributions of this paper is the ViT-based decentralized control for multi-robot formation. Compared to existing work of learning-based end-to-end control of multi-robot formation [23], [24], the use of ViT significantly improves efficiency of feature extraction from the robot's observation. Our proposed method does not need inter-robot communication, which facilitates decentralized implementation and dynamic re-configuration (i.e., adding or dropping robots from the team).

The remainder of the paper is organized as follows. Section II presents the model of the holonomic robots with four Mecanum wheels and the formulation of our control problem. The vision transformer-based training and on-line control methods are described in Section III. The simulation results and the real robot experiments are presented in Section IV and Section V, respectively. We provide a discussion and comparison to other related works in Section VI . Finally, the paper is concluded in Section VII.

II. Problem Statement

In this paper, we consider a multi-robot system consisting of N holonomic mobile robots with four Mecanum-wheels. Denote the ith robot's position and orientation in a global coordinate (i.e. the world frame) as $\boldsymbol{p}_i \triangleq [x_i, y_i]^T$ and θ_i, respectively. The robot velocity control is defined as $\mathbf{u}_i = [u_{ix}, u_{iy}, \omega_i]^T$, where u_{ix} and u_{iy} are the linear velocity of the robot in the x and y directions of the robot frame, respectively, and ω_i is the angular velocity. The kinematic

The authors are with the Department of Electrical and Computer Engineering, Stevens Institute of Technology, Hoboken, NJ 07030, USA. Emails: {xhuang60, gyang11, yguo1}@stevens.edu

model of the ith robot, $i \in \{1, ..., N\}$, is given by the discrete-time model:

$$\begin{bmatrix} x_i(t+1) \\ y_i(t+1) \\ \theta_i(t+1) \end{bmatrix} = \begin{bmatrix} x_i(t) \\ y_i(t) \\ \theta_i(t) \end{bmatrix} + \boldsymbol{G}(t) \begin{bmatrix} u_{ix}(t) \\ u_{iy}(t) \\ \omega_i(t) \end{bmatrix}, \quad (1)$$

where $\boldsymbol{G}(t)$ is given below, and ΔT is the sampling period,

$$\boldsymbol{G}(t) = \begin{bmatrix} \Delta T \cos\theta_i(t) & -\Delta T \sin\theta_i(t) & 0 \\ \Delta T \sin\theta_i(t) & \Delta T \cos\theta_i(t) & 0 \\ 0 & 0 & \Delta T \end{bmatrix} \quad (2)$$

Note that the relationship between the robot's velocities and the Mecanum-wheel's velocities are given in references such as [25]. In this paper, we focus on learning the robot control policy at the kinematic level, and the lower-level wheel control is not discussed in the paper (as it is provided by the robot manufacturer).

To formulate our cooperative control problem, we use the proximity graph terminology to define the triangulation formation of robots. The robot team proximity graph is defined as a Gabriel graph [1], denoted as $\mathcal{G} = (V, E)$, where $V = v_1, ..., v_N$ denotes the set of vertices corresponding to the robots located at the positions $\boldsymbol{p}_1, ..., \boldsymbol{p}_N \in \mathbb{R}^2$, and E represents the set of edges. An edge exists between vertices v_i and $v_j \in V$, denoted as $v_i, v_j \in E$, if and only if the circle with a diameter of $|\boldsymbol{p}_i - \boldsymbol{p}_j|$ encompassing both vertices does not enclose any other vertex within its boundary. Note that the Gabriel graph is a nearest neighbor interaction graph to a large degree, and the proximity graph $\mathcal{G}$ solves the combinatorial coverage problem if it is a perfect planar triangulation [1].

The objective of cooperative control is to find a decentralized control input for each robot such that, starting from positions that the proximity graph $\mathcal{G}$ is connected, the group of robots achieves triangulation formations with a predefined distance between robots d^*, for all pairs of robots $\{v_i, v_j\} \in E$. That is, $\|\boldsymbol{p}_i - \boldsymbol{p}_j\| \to d^*$ as $t \to \infty$, $\forall\{v_i, v_j\} \in E$. Note that in this configuration, the proximity graph $\mathcal{G}$ is subject to changes over time, as the set of neighboring robots for each robot changes as they move.

To solve the cooperative control problem defined above, we propose a learning-based method and train a neural network policy defined as:

$$\mathbf{u}_i = \pi(\mathbf{o}_i, \boldsymbol{\Theta}), i \in \{1, ..., N\} \quad (3)$$

where the policy uses each robot's own observation $\mathbf{o}_i$, and $\boldsymbol{\Theta}$ represents the neural network parameters onboard of the ith robot. The policy is decentralized, relying only on local observation of each robot. After training, during online robot motion control, the policy computes robot velocities end-to-end through a feed-forward pass in a decentralized manner.

III. The Method

The overview of the proposed learning-based multi-robot cooperative control is shown in Fig. 1. For each robot i, the raw 2-D LiDAR scan $S_i(t)$ is transformed into a binary occupancy map $o_i(t)$ that serves as input to the ViT encoder. The complete transform procedure is summarized in Appendix A. These occupancy maps serve as input to neural networks. The decentralized control policy is parameterized by a deep neural network (DNN) that consists of a ViT encoder and a multilayer perceptron (MLP) network. Each robot in the team has a copy of the trained DNN model whose parameters are the same. The Transformer Encoder extracts features from an occupancy map obtained by the robot's own onboard sensor. The MLP layers compute the robot velocity control $\mathbf{u}_i$ as the output of the DNN algorithm that drives the robot. The DNN parameters are trained offline using demonstration data and are loaded during online robot control to control the robot velocity control through a feed-forward pass in a decentralized manner. In this section, we explain each function block as shown in Fig. 1, and present both the training and the online control algorithms.

A. Vision Transformer

We adopt the Transformer architecture in our method, drawing inspiration from the Vision Transformer [20]. We treat occupancy maps as images and separate the occupancy map $\mathbf{o}$ with the shape of (H, W) into N of 2D patches with the shape of (P, P). Then we apply learnable linear projections $\mathbf{E}$ to the patches to get the patch embedding $\mathbf{E}(\mathbf{o}_p) \in \mathbb{R}^{P^2 \times D}$. Then we add a learnable position embedding $\mathbf{E}_{pos} \in \mathbb{R}^{(N+1) \times D}$ to the sequence of embedding to retain positional information. The formulation below describes the embedding:

$$z_0 = [\mathbf{o}_p^1\mathbf{E}; \mathbf{o}_p^2\mathbf{E}; \cdots; \mathbf{o}_p^N\mathbf{E}] + \mathbf{E}_{pos} \quad (4)$$

The embedding is then passed to multiple transformer layers consisting of alternating layers of multi-headed self-attention (MSA) [26] and Multi-Layer perceptron (MLP). Layernorm (LN) is applied before every MSA and MLP block:

$$z_l' = MSA(LN(z_l)) + z_{l-1} \quad (5)$$

$$z_l = MLP(LN(z_l')) + z_l' \quad (6)$$

where z_l denotes the output of each transformer layer l. After the transformer layers, the features z_l are fed into an MLP to compute a robot control command $\mathbf{u}$.

B. Policy Training

1) Expert Controller: A model-based controller [1] for the multi-robot triangulation formation problem is employed to control the robot to form the desired formation. It achieves triangulation formations by minimizing the cost function associated with robots i and j, i.e.,

$$U_{ij} = \frac{1}{2}(\|\boldsymbol{p}_i - \boldsymbol{p}_j\| - d^*)^2, \forall\{i, j\} \in E \quad (7)$$

where robot j belongs to the neighboring robots, $\mathcal{N}(i)$, defined by the Gabriel graph $\mathcal{G}$.

The potential function takes on its minimum at the prescribed inter-robot distance, d^*, assuming single-integrator dynamics of the robots, the control law is given by

$$\mathbf{v}_i = K_f \sum_{j \in \mathcal{N}(i)} \frac{\|\boldsymbol{p}_i - \boldsymbol{p}_j\| - d^*}{\|\boldsymbol{p}_i - \boldsymbol{p}_j\|} \cdot (\boldsymbol{p}_i - \boldsymbol{p}_j) \quad (8)$$

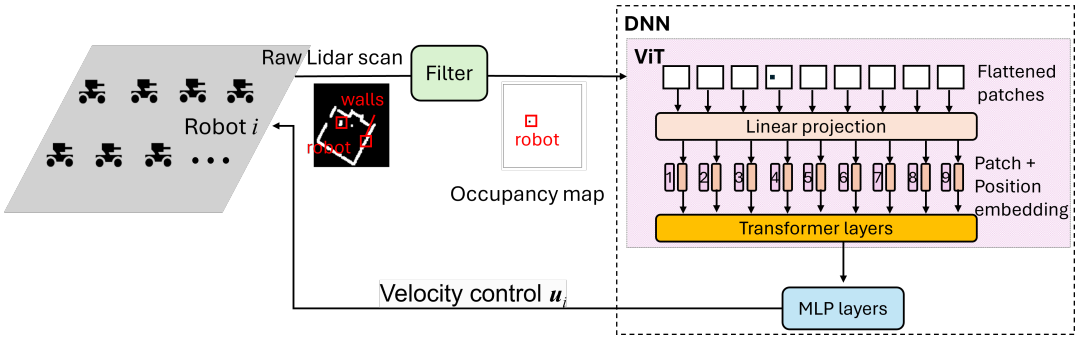


Fig. 1: The overall diagram of ViT-based decentralized formation control.

where K_f is the control gain. When the inter-robot distance is greater than d^*, the controller exerts attractive force. When the inter-robot distance is smaller than d^*, the controller repels the robots away from each other. At convergence, the neighboring robots form triangulations with the distance d^*.

The control input $\mathbf{v}_i$, computed by the expert controller for the single-integrator model, is transformed to the robot's velocity via the following coordinate transformation method:

$$\begin{bmatrix} u_{ix} \\ u_{iy} \\ \omega_i \end{bmatrix} = \begin{bmatrix} \cos\theta_i & \sin\theta_i \\ -\sin\theta_i & \cos\theta_i \\ 0 & 0 \end{bmatrix} \cdot \mathbf{v}_i \tag{9}$$

Then, the holonomic robot can be controlled by the expert controller after transformation.

2) DNN Model Training: We train the policy (3) offline in a robot simulator using a learning-from-demonstration (LfD) approach. Specifically, we define a loss function that measures the discrepancy between the policy's output and the expert control:

$$\mathcal{L}(\mathbf{\Theta}) = \frac{1}{N}\sum_{i=1}^{N}\frac{1}{2}\left\|\pi(\mathbf{o}_i, \mathbf{\Theta}) - \mathbf{u}_i^*\right\|^2 \tag{10}$$

The neural network parameters $\mathbf{\Theta}$ are iteratively updated to minimize this loss, enabling the ViT-based policy to progressively refine its performance from expert demonstrations.

The complete training procedure is summarized in Algorithm 1. The training dataset $\mathcal{D}$ is collected in a simulator consisting of robot position $\mathbf{p}$ and occupancy maps $\mathbf{o}$. The dataset contains K batches, each with N robots' position, denoted as $\mathbf{P}(s) = [\mathbf{p}_1(s), \mathbf{p}_2(s), \ldots, \mathbf{p}_N(s)]$ and their corresponding observations (i.e. occupancy map), denoted as $\mathbf{O}(s) = [\mathbf{o}_1(s), \mathbf{o}_2(s), \ldots, \mathbf{o}_N(s)]$ where $s = 1, 2, \ldots, K$.

We begin by initializing the model parameters $\mathbf{\Theta}$ and proceed with K training iterations. At each iteration s, we sample a batch of robot position $\mathbf{P}(s)$ and observations $\mathbf{O}(s)$ from the demonstration dataset $\mathcal{D}$ (Lines 3–4). For each robot $i \in \{1, \ldots, N\}$, we compute the expert control $\mathbf{u}_i^*$ using (8) (Line 5), then obtain an occupancy map $\mathbf{o}_i(s)$ that serves as input to the policy network (Line 6). The network's predicted control $\mathbf{u}_i$ is generated by forward propagation through the DNN with parameters $\mathbf{\Theta}_s$ (Line 7).

Algorithm 1 DNN Model Training

Input: Demonstration dataset $\mathcal{D}$
Output: Trained model parameters $\mathbf{\Theta}$
1: Initialize the model parameters $\mathbf{\Theta}$
2: **for** $s = 1$ to K **do**
3: Sample a batch of position $\mathbf{P}(s)$ and observations $\mathbf{O}(s)$
4: **for** each robot $i = 1$ to N **do**
5: Obtain expert control command $\mathbf{u}_i^*$ with (8) and (9)
6: Obtain the occupancy map $\mathbf{o}_i(s)$ for robot i
7: Compute the control from the model $\mathbf{u}_i \leftarrow \pi(\mathbf{o}_i, \mathbf{\Theta})$ for each robot i
8: Collect the model output and reference control $\{\mathbf{u}_i, \mathbf{u}_i^*\}$
9: **end for**
10: Calculate the loss $\mathcal{L}$ using (10)
11: Update parameters by gradient descent: $\mathbf{\Theta_{s+1}} \leftarrow \mathbf{\Theta_s} - \beta\nabla\mathcal{L}(\mathbf{\Theta}_s)$
12: **end for**
13: **return** The trained model parameters $\mathbf{\Theta}$

We collect both the predicted controls $\mathbf{u}_i$ and the corresponding expert controls $\mathbf{u}_i^*$ to evaluate the loss function (10) (Line 10). This loss quantifies the discrepancy between the network output and the expert reference. Finally, the model parameters are updated via gradient descent (Line 11), using the learning rate β to scale the gradient of the loss function with respect to $\mathbf{\Theta}$. Repeating this process for K iterations progressively refines the policy. The parameters are updated via gradient descent.

C. Online Cooperative Control

Once the policy network has been trained via Algorithm 1, we deploy the trained DNN model on each robot to coordinate the motion of each robots, as summarized in Algorithm 2. At each time step t, every robot i collects a raw LiDAR scan $\mathbf{S}_i(t)$ (Line 3). This scan is then transformed into an occupancy map $\mathbf{o}_i(t)$ using the LiDAR preprocessing method described in Algorithm 3 in Appendix A(Line 4). The policy π, parameterized by the trained weights $\mathbf{\Theta}$,

subsequently infers the control action $\mathbf{u}_i(t)$ (Line 5). Finally, each robot executes its respective control command (Line 6).

Algorithm 2 Online Cooperative Control of Robot i

Input: Current LiDAR scan $\mathbf{S}_i(t)$ for each robot i at time step t

Output: Control action $\mathbf{u}_i(t)$ for each robot i at time step t

1: Load the DNN model parameters trained by Algorithm 1
2: **for** time step $t = 1$ to T **do**
3: Obtain the raw LiDAR scan $\mathbf{S}_i(t)$ for robot i
4: Convert the the raw LiDAR scan $\mathbf{S}_i(t)$ to occupancy map $\mathbf{o}_i(t)$ using Algorithm 3 in Appendix
5: Compute the control action $\mathbf{u}_i(t)$ using policy $\pi(\mathbf{o}_i, \mathbf{\Theta})$
6: Execute the control action $\mathbf{u}_i(t)$
7: **end for**

IV. Simulation Results

A. Simulation Environment and Parameters

We use ROS Gazebo as the simulation platform. We select a team of Robotmaster EP mobile robots equipped with LiDar sensor Youyeetoo LD20 sensors. The sensing range of the Lidar sensor is set to 2 meters. The resulting occupancy map generated from the sensor readings had dimensions of 100 pixels × 100 pixels, with a granularity of 0.0016 square meters per pixel. The simulation time step is set to 0.01s.

An Intel Core i7-12700K CPU (12 cores at 3.6 GHz) and an NVIDIA RTX 3090 GPU are used for both neural network training and testing in simulation. The PyTorch framework is utilized to implement the ViT architecture and handle all computations during training and testing.

Our ViT implementation uses 3 transformer layers with eight attention heads each, a latent dimension $D = 256$, and an output MLP with 2 layers having a dimension of $D_{MLP} = 512$. The chosen patch size P is 10.

B. Performance Metrics

We define the formation error between the neighboring robots i and j at time t as: $\mathcal{E}_{i,j}(t) = |\|\boldsymbol{p}_i(t) - \boldsymbol{p}_j(t)\| - d^*|$. The group formation error at any time t is defined as: $\mathcal{E}_g(t) = \frac{1}{N}\sum_{j\in\mathcal{N}(i)} \mathcal{E}_{i,j}(t)$. Accordingly, the group formation error $\mathcal{E}_{group}$ is defined as the temporal average of $\mathcal{E}_g(t)$ over the last 5s before the end of the simulation.

During the testing phase, we define the convergence of the multi-robot system based on the group formation error. Specifically, we consider the system as "converged" if the temporal average of the group formation error $\mathcal{E}_g(t)/d^*$ over the most recent 5 s is less than a predefined threshold of 10%.

1) *Success rate*: $Rate = n_{success}/n$, is the proportion of successful cases over total number of tested cases n. A simulation run is considered successful if it converges before the end of the simulation.
2) *Converge time*: $T_{converge}$ is the time when a simulation run converges. That is, the first time that the temporal average of the group formation error over the most recent 5 s reaches the 10% threshold and then keeps decreasing.
3) *Group formation error* defined in percentage: $\mathcal{E}_{group}/d^*$.

C. Training Procedure

The training dataset is collected in a simulator. To get the training dataset, we conducted 600 simulation runs with $N = 7$ robots with initial positions within a 5 m × 5m square area, while their initial orientations are chosen randomly from the range of $[0, 2\pi]$. Algorithm 1 is executed for the training. The desired triangulation formation is set at a distance of $d^* = 1$. Each simulation run lasted for 1000 time steps, during which we recorded the position p_i of the robots and their observations $\mathbf{o}_i$ at each time step, so we get 600,000 data pairs. To enhance scalability, we introduced another 400,000 data pairs by randomly placing the seven robots within a 5m × 5m square and recorded their position and observations. The model is trained using Algorithm 1 for a single epoch during training. The learning rate, β, is set to 0.01, and the size of mini-batch B, used to calculated gradient is set to 16. We employed the RMSprop optimizer [27] for model optimization.

D. Simulation Results

After training with $N = 7$ robots, we tested our decentralized cooperative control for various numbers of robots in the simulator by running Algorithm 2. The robots are placed randomly within a 5m × 5m square, with their initial orientations are chosen randomly from the range $[0, 2\pi]$. The desired triangulation formation is set at a distance of $d^* = 1$. In the following, we present both the simulation and statistical results of these experiments.

Fig. 2 provides snapshots of the robots achieving triangulation formations in the Gazebo simulator at different time intervals, with the team size $N = 9$ in (a) and $N = 13$ in (b).

In Fig. 3, we show the testing results for different robot team sizes, specifically $N = 5, 7, 9, 11, 13$. The top sections of the figures display the time histories of inter-robot distances, that is, $d_{ij}(t), i = 1, \ldots, N, j \in \mathcal{N}(i)$, while the bottom sections illustrate the trajectories of each robot. In particular, the results demonstrate that the robot team successfully achieves the desired triangulation formation while maintaining the desired neighboring distance of $d^* = 1$m.

To assess scalability, we conducted a series of testing experiments encompassing 100 different initial robot conditions for team size varying from $N = 5$ to $N = 13$ robots. The success rates achieved for various robot-team sizes are detailed in Table I. This tolerance signifies that the group formation error $\mathcal{E}_{group}/d^*$ remains below 10%, as defined in Section IV-B. This result demonstrates the scalability of the trained DNN policy, which successfully extends to different team size while trained with a 7-robot team.

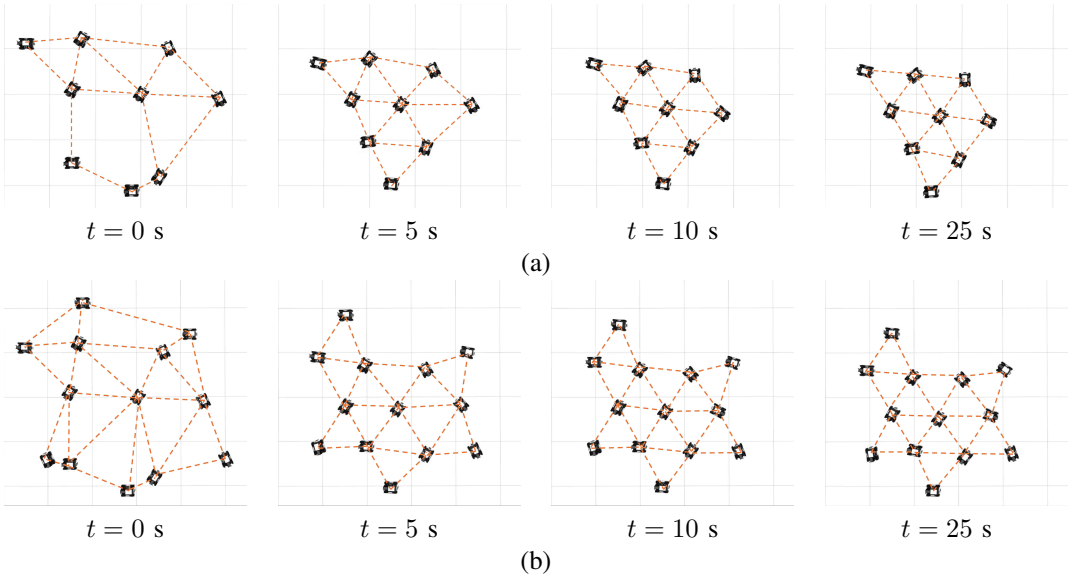


Fig. 2: Snapshots of formation control in Gazebo simulator at different time t. (a): 9 robot team; (b): 13 robot team. The dotted lines indicate the proximity graph $\mathcal{G}$

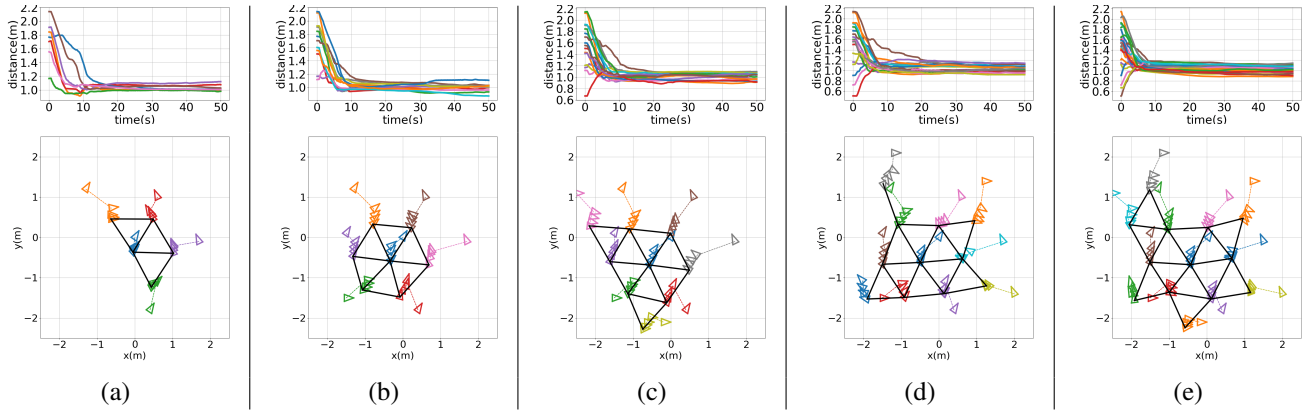


Fig. 3: Testing results for (a) a five-robot team ($N = 5$), (b) a seven-robot team ($N = 7$), (c) a nine-robot team ($N = 9$), (d) a eleven-robot team ($N = 11$), (e) a thirteen-robot team ($N = 13$). Top: Time histories of inter-robot distance, $d_{ij}(t), i = 1, \ldots, N, j \in \mathcal{N}(i)$; Bottom: Robot trajectories with robot positions (denoted by colored small triangles) drawn every 10 seconds, and the black solid lines indicate the final formation achieved.

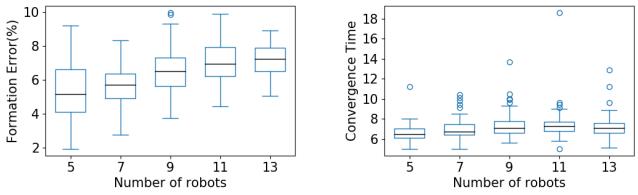
TABLE I: Success rates over 100 runs

Number of robots:	5	7	9	11	13
Success rate % :	100	99	99	99	100

We show more detailed results of the statistical performance evaluation in Fig. 4, for a group of robots of sizes from 5 to 13, starting from 100 different initial conditions. In Fig. 4 (a), we present a box plot depicting the group formation error $\mathcal{E}_{group}$, as defined in Section III-B.1. In particular, the median formation errors for different teams between 5% and 7%. Fig.4 (b) displays another box plot, the convergence time, $T_{converge}$, as defined in Section IV-B. In this case, the median convergence time typically falls around 7s. Fig. 4 shows the robustness and scalability of our trained DNN model. Despite being initially trained with a seven-robot team, the learned controller effectively adapts to various multi-robot team sizes, achieving satisfactory performance in terms of both the group formation error and the convergence time.

E. Dynamically-Changing Team Size

Our approach operates in a fully decentralized manner, allowing for a dynamic change in the number of robots within the team. Consequently, robots can autonomously reconfigure into a new formation when the team size changes. As illustrated in Fig. 5 (a), a team initially composed of 6 robots forms a triangular formation at $t = 20$ s. At $t = 21$ s, a new robot is added from the top right, then the 7 robot team forms a triangular formation $t = 40$ s. In Fig. 5 (b), a team composed of $N = 7$ robots forms a triangular formation at $t = 20$ s. At $t = 21$ s, the robot at the center is removed from the team, disrupting the previously achieved formation.



(a) (b)

Fig. 4: Box plot for 100 initial conditions of multi-robot testing. The central mark in each box is the median, the edges of the boxes are the 25th and 75th percentiles, the whiskers extend to the maximum/minimum, and the circles represent outliers. (a) Group formation error for multi-robot teams with sizes 5, 7, 9, 11, and 13; (b) Convergence time for multi-robot teams with sizes 5, 7, 9, 11, and 13.

The remaining 6 robot moves to achieve a new triangular formation, and successfully achieves it at $t = 40$ s. These results show that our approach is robust to the dynamically-changing team size.

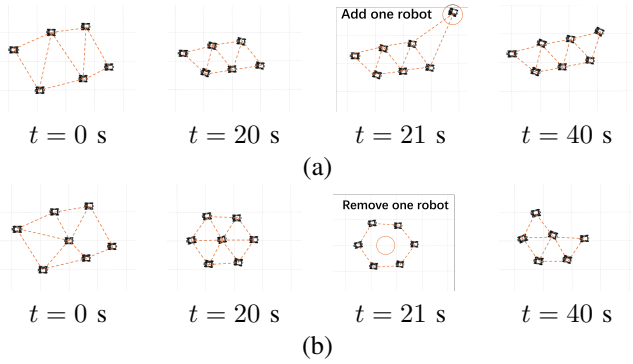


Fig. 5: Snapshots of formation control in Gazebo simulator at different time t when adding and removing robots. (a) A 6-robot team forms a formation at $t = 20$s then one robot is added to the team at $t = 21$s, and a new formation is formed at $t = 40$s; (b) A 7- robot team forms a formation at $t = 20$s, then one robot is removed from the team at $t = 21$s, and a new formation is formed at $t = 40$s.

V. Real Robot Experiments

We customized four RoboMaster EP Core robots by DJI and added a Youyeetoo LD20 2D LiDAR sensor for observation, and an NVIDIA Jetson Nano board for on-board processing, as shown in Fig. 6, the Jetson Nano board integrates a quad-core ARM Cortex-A57 CPU and a 128-core Maxwell GPU with 4 GB LPDDR4 memory (25.6 GB/s bandwidth) in a compact 5–10 W power envelope. The markers shown in figure are for ground truth recording using a motion capture system (i.e., OptiTrack), but are not used in controlling the robots. The real robot experiments was carried out in 10 W power mode, the memory usage is approximately 17 MB, the onboard inference requires about 0.12 s to process each LiDAR-based occupancy map (equivalent to an effective control frequency of about 8 Hz).

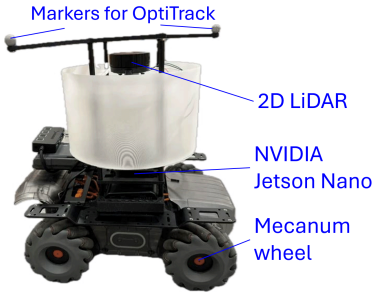


Fig. 6: Customized RoboMaster EP Core robots with LiDAR sensor and NVIDIA Jetson Nano for onboard computing. The markers are mounted for performance evaluation using OptiTrack.

We conducted a series of real-robot experiments with the robots controlled by the decentralized control strategy as outlined in Algorithm 2. The DNN model is trained in the simulator. The four robots are initially placed randomly within a 4m $\times$ 4m square area, with their initial orientations are chosen randomly from the range $[0, 2\pi]$.The desired triangulation formation is set at a distance of $d^* = 1$ m.

Fig. 7 presents the snapshots of the positions of the robots at various time steps in two specific testing scenarios. Fig. 8 presents the corresponding time histories of the inter-robot distances (Top) and the trajectories (Bottom) of each robot. The sampling rate of these plots matches the effective control frequency of the onboard inference (about 8 Hz), ensuring that the depicted trajectories directly reflect the decision-making resolution of the decentralized controller. We can see that the four robots achieved triangulation formation successfully after 15s.

Statistical results are shown in Fig. 9 (a), we present a bar plot depicting the group formation error, $\mathcal{E}_{group}$, as defined in Section IV-B. Figure 9 (b) displays the convergence time, $T_{converge}$. In this case, the median formation errors typically fall around 9%, and the median convergence time typically falls around 14 seconds. The real-world performance generally aligns with the simulation results, but some discrepancies arise from hardware constraints. In particular, the 2D LiDAR operates at a relatively low frequency (6Hz), limiting the robot's ability to rapidly detect and respond to other robots' position changes. Additionally, the mechanical structure—including friction, backlash, and other nonlinearities inherent to gear and drive systems—introduces variations not captured in the simulator.

VI. Discussion

Conventional methods, exemplified by dynamic model-based expert controllers [1], [3], [28], provide rigorous mathematical guarantees for formation convergence but depend heavily on sequential perception-and-control pipelines. Such architectures introduce delays and cumulative sensing errors, reducing performance robustness in practical deployments.

This research contributes an alternative learning-from-demonstration (LfD) based method, directly utilizing on-

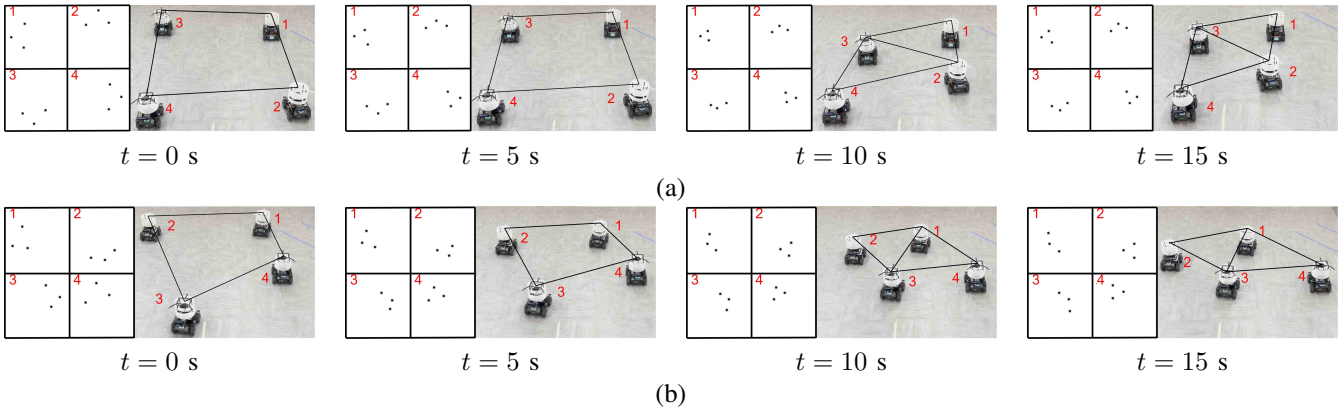


Fig. 7: Real robot testing results for formation control in two testing cases (a) and (b). In each snapshot, the left box shows the occupancy map of each robot with their real-world positions shown on the right.

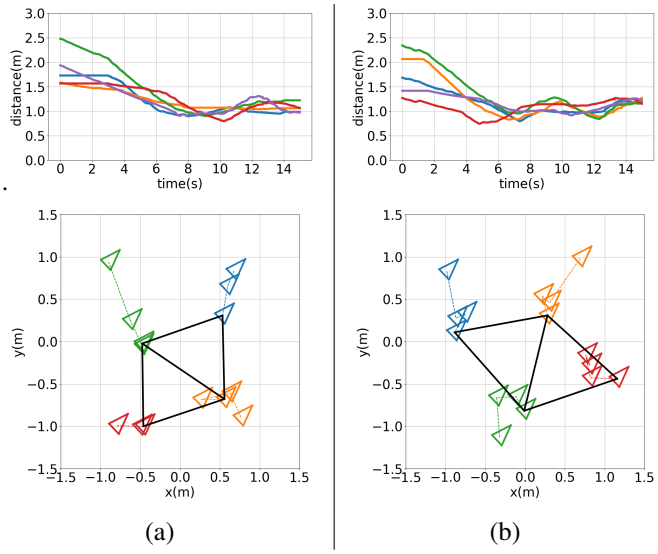


Fig. 8: (Top) Time histories of inter-robot distance, $d_{ij}(t), i = 1, \ldots, 4, j \in \mathcal{N}(i)$; (Bottom) Robot trajectories with robot positions (denoted by colored small triangles) drawn every 5 seconds.

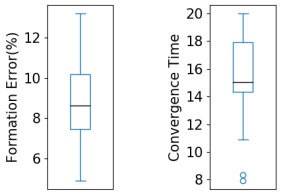


Fig. 9: (a) Group formation error of 9 different initial conditions of real robot testing. (b) Convergence time of 9 different initial conditions of real robot testing.

board robot sensing in an end-to-end framework. Unlike multi-robot reinforcement learning (RL) techniques [9], [13], [14], [29], our LfD approach leverages expert demonstrations to significantly reduce the policy search space, resulting in faster training convergence.

Comparing to existing LfD methods that use CNNs or GNNs, CNN-based policies [29], [30] effectively capture local features from LiDAR occupancy maps but are limited in modeling long-range dependencies. GNN-based methods [9], [24], [31], [32] provide scalability and permutation invariance through message passing, yet require predefined communication graphs and fixed team sizes, constraining flexibility. Our approach overcomes these limitations by eliminating the need for predefined communication structures or explicit inter-robot messaging, thereby enabling flexibility in accommodating dynamic team sizes and varying inter-robot relationships.

VII. Conclusion

In this paper, we have presented a novel decentralized multi-robot control for triangulation formation. We designed ViT-based algorithms for learning scalable control policies. Experimental validation was performed in the robot simulator, as well as on real robot platforms. The evaluation results have shown satisfactory performance for varying size of multi-robot teams.

Appendix

A. Raw LiDAR Scan Processing

The occupancy map generating procedure is elaborated in Algorithm 3. This approach involves projecting the LiDAR-generated points onto a two-dimensional grid (Line 1) and applying a Gaussian filter to smooth this grid (Line 2). Due to noise and potential obstructions in real-world environments, we implemented a filtering approach that removes noise and walls from the scans before converting them into occupancy maps. The grids are processed to identify and remove connected groups of points that are either excessively large, likely representing walls (Lines 6–7), or too small, indicative of noise (Lines 8–9). In our experiments, the Gaussian kernel is set to a size of 3×3. A cell $m_{i,j}$ in the grid is considered to be part of a connected group c if any of its neighboring cells—$m_{i-1,j}$, $m_{i+1,j}$, $m_{i,j-1}$, or $m_{i,j+1}$—also belong to c. A cell $m_{i,j}$ without neighboring cells is treated as its own connected group.

Algorithm 3 Raw LiDAR Scan Processing

Input: Raw Lidar scan $\mathbf{S}_i(t)$ for robot i at time step t
Output: Occupancy map $\mathbf{o}_i(t)$ for time step t

Project raw Lidar scan $\mathbf{S}_i(t)$ onto a 2D grid matrix M
Smooth M by applying a Gaussian filter with kernel size $k \times k$
Convert all non-zero elements in M to 1 to indicate occupied spaces
Identify all connected components of 1s in M, storing each as a separate element in list L
for each connected component c in L **do**
  if the size of c exceeds an upper threshold **then**
    Set all elements in c within M to 0 (clearing overly large obstacles)
  else if the size of c is below a lower threshold **then**
    Set all elements in c within M to 0 (removing noise)
  end if
end for
return Processed occupancy map $\mathbf{o}_i(t)$ for time step t

REFERENCES

[1] M. Mesbahi and M. Egerstedtsca, *Graph theoretic methods in multi-agent networks*. Princeton University Press, 2010.

[2] F. L. Lewis, H. Zhang, K. Hengster-Movric, and A. Das, *Cooperative Control of Multi-Agent Systems: Optimal and Adaptive Design Approaches*. Springer London, 2014.

[3] Y. Guo, *Distributed Cooperative Control: Emerging Applications*. Wiley, 2017.

[4] P. Stone and M. Veloso, "Multiagent systems: A survey from a machine learning perspective," *Autonomous Robots*, vol. 8, no. 3, pp. 345–383, 2000.

[5] S. Gronauer and K. Diepold, "Multi-agent deep reinforcement learning: a survey," *Artificial Intelligence Review*, pp. 1–49, 2021.

[6] S. Sukhbaatar, R. Fergus *et al.*, "Learning multiagent communication with backpropagation," *Advances in neural information processing systems*, vol. 29, pp. 2244–2252, 2016.

[7] J. Foerster, I. A. Assael, N. De Freitas, and S. Whiteson, "Learning to communicate with deep multi-agent reinforcement learning," *Advances in neural information processing systems*, vol. 29, 2016.

[8] J. Jiang and Z. Lu, "Learning attentional communication for multi-agent cooperation," *Advances in neural information processing systems*, vol. 31, 2018.

[9] J. Blumenkamp, S. Morad, J. Gielis, Q. Li, and A. Prorok, "A framework for real-world multi-robot systems running decentralized gnn-based policies," in *Proc. IEEE Int. Conf. Robot. Automat.*, Philadelphia, PA, USA, 2022, pp. 8772–8778.

[10] Y. Wang, Y. Yue, M. Shan, L. He, and D. Wang, "Formation reconstruction and trajectory replanning for multi-uav patrol," *IEEE/ASME Transactions on Mechatronics*, vol. 26, no. 2, pp. 719–729, 2021.

[11] M. F. Rahaman, X. Li, K. M. Zakaria, A. M. Al, K. Buse, and S. A. Bashar, "Enhancing multi-robot formation control and navigation using virtual structures and improved path planning algorithms," in *Proc. Int. Conf. Innovative Research in Applied Science, Engineering and Technology.*, Fez, Morocco, 2024, pp. 1–8.

[12] S.-M. Lee and J.-U. Lee, "Multi-robot formation planning in maze-like environments consisting of narrow passages using graph search," *IEEE Access*, 2024.

[13] Y. Yan, X. Li, X. Qiu, J. Qiu, J. Wang, Y. Wang, and Y. Shen, "Relative distributed formation and obstacle avoidance with multi-agent reinforcement learning," in *Proc. IEEE Int. Conf. Robot. Automat.*, Philadelphia, PA, USA, 2022, pp. 1661–1667.

[14] A. Agarwal, S. Kumar, K. Sycara, and M. Lewis, "Learning transferable cooperative behavior in multi-agent teams," in *Proc. Int. Conf. Autonomous Agents Multiagent Systems*, Auckland, New Zealand, 2020, pp. 1741–1743.

[15] K. M. Kabore and S. Güler, "Distributed formation control of drones with onboard perception," *IEEE/ASME Transactions on Mechatronics*, vol. 27, no. 5, pp. 3121–3131, 2021.

[16] C. Bai, P. Yan, W. Pan, and J. Guo, "Learning-based multi-robot formation control with obstacle avoidance," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 8, pp. 11 811–11 822, 2021.

[17] S. Li, R. Kong, and Y. Guo, "Cooperative distributed source seeking by multiple robots: Algorithms and experiments," *IEEE/ASME Transactions on Mechatronics*, vol. 19, no. 6, pp. 1810–1820, 2014.

[18] Z. Wan, C. Jiang, M. Fahad, Z. Ni, Y. Guo, and H. He, "Robot-assisted pedestrian regulation based on deep reinforcement learning," *IEEE Transactions on Cybernetics*, vol. 50, no. 4, pp. 1669–1682, 2020.

[19] J. Obradović, M. Križmančić, and S. Bogdan, "Decentralized multi-robot formation control using reinforcement learning," in *Proc. Int. Conf. Information Communication and Automation Technologies*, Sarajevo, Bosnia and Herzegovina, 2023, pp. 1–7.

[20] A. Dosovitskiy, "An image is worth 16x16 words: Transformers for image recognition at scale," *arXiv preprint arXiv:2010.11929*, 2020.

[21] P. Wang, S. Wang, J. Lin, S. Bai, X. Zhou, J. Zhou, X. Wang, and C. Zhou, "One-peace: Exploring one general representation model toward unlimited modalities," *arXiv preprint arXiv:2305.11172*, 2023.

[22] Z. Zong, G. Song, and Y. Liu, "Detrs with collaborative hybrid assignments training," in *Proc. IEEE/CVF Int. Conf. Computer Vision*, Paris, France, 2023, pp. 6748–6758.

[23] C. Jiang, Z. Chen, and Y. Guo, "Learning decentralized control policies for multi-robot formation," in *Proc. IEEE/ASME Int. Conf. Adv. Intell. Mechatronics*, Hong Kong, China, 2019, pp. 758–765.

[24] C. Jiang, X. Huang, and Y. Guo, "End-to-end decentralized formation control using a graph neural network-based learning method," *Frontiers in Robotics and AI*, vol. 10, 2023.

[25] H. Taheri, B. Qiao, and N. Ghaeminezhad, "Kinematic model of a four mecanum wheeled mobile robot," *International journal of computer applications*, vol. 113, no. 3, pp. 6–9, 2015.

[26] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in neural information processing systems*, vol. 30, 2017.

[27] S. Ruder, "An overview of gradient descent optimization algorithms," 09 2016.

[28] J. Cortés and M. Egerstedt, "Coordinated control of multi-robot systems: A survey," *SICE Journal of Control, Measurement, and System Integration*, vol. 10, no. 6, pp. 495–503, 2017.

[29] M. Li, Y. Jie, Y. Kong, and H. Cheng, "Decentralized global connectivity maintenance for multi-robot navigation: A reinforcement learning approach," in *Proc. IEEE Int. Conf. Robot. Autom.*, Philadelphia, PA, USA, 2022, pp. 8801–8807.

[30] F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, and G. Monfardini, "The graph neural network model," *IEEE transactions on neural networks*, vol. 20, no. 1, pp. 61–80, 2008.

[31] E. Tolstaya, F. Gama, J. Paulos, G. Pappas, V. Kumar, and A. Ribeiro, "Learning decentralized controllers for robot swarms with graph neural networks," in *Proc. Conf. Robot Learn.*, 2020, pp. 671–682.

[32] A. Moallemy-Oureh, S. Beddar-Wiesing, R. Nather, and J. M. Thomas, "Fdgnn: Fully dynamic graph neural network," *arXiv preprint arXiv:2206.03469*, 2022.