Multi-Robot Guided Policy Search for Learning Decentralized Swarm Control

Chao Jiang, *Member, IEEE,* and Yi Guo, *Senior Member, IEEE*

***Abstract*—Multi-robot learning has been extensively studied recently. Developing provably-correct algorithms for learning decentralized control policies remains challenging. In this letter, we propose a sample-efficient multi-robot learning method based on guided policy search to learn decentralized swarm control policies. The proposed method uses distributed trajectory optimization to provide guiding trajectory samples for policy training. In turn, the learned policy is exploited to update trajectory optimization results so that the guiding trajectories are reproducible by the current policy. A learning algorithm is designed to alternate between distributed trajectory optimization and policy optimization, which eventually converges to the solution with good long-term performance. We demonstrate the effectiveness of our method in a multi-robot rendezvous problem. The simulation results in a robot simulator show that our method efficiently learn decentralized control policy with substantially less training samples.**

***Index Terms*—Multi-robot learning, distributed trajectory optimization, guided policy search, robotic swarm**

I. Introduction

MULTI-ROBOT learning problems are commonly formulated as Markov games [1] or decentralized partially observable Markov decision process (Dec-POMDP) [2]. Multi-robot learning has been investigated under the reinforcement learning (RL) paradigm to learn decentralized policies in the framework of concurrent learning [3], [4]. However, as the robots update their policies concurrently during learning, the environment becomes non-stationary. This could be problematic since previous experience used for learning becomes invalid as the dynamics of environment changes, which makes the learning unstable. In [5], a distributed online learning of motion control is adopted for multi-robot cooperation. In [6], a novel neural network policy that spans an entire robot swarm is proposed. The robot swarm communicates and collectively trains the policy in a decentralized manner. Alternatively, the framework of centralized learning and decentralized execution has been extensively adopted [7]–[11], where training is centralized and global information is accessible for each robot to evaluate and improve its policy, while only local information is used during decentralized execution.

A majority of existing work adopts the RL paradigm for the centralized training phase. However, designing appropriate reward functions that fully capture the learning objectives could be difficult for multi-robot systems in complex applications. Moreover, the search space in multi-robot learning could be substantially huge, thus sample-efficiency and algorithm tractability have been the main concerns for multi-robot reinforcement learning [12].

In this letter, we focus on sample-efficient multi-robot learning, and propose a novel multi-robot learning method based on the guided policy search (GPS) [13], [14] to learn decentralized control policies. The proposed method uses distributed trajectory optimization to generate guiding trajectory samples for policy training. In turn, the learned policy is exploited to update the guiding trajectories so that the guiding trajectories are reproducible by the current policy. The distributed trajectory optimization and the policy optimization are alternately performed based on alternating direction method of multipliers (ADMM) [15]. The algorithm eventually converges to a solution that exhibits good long-term performance. We verify the effectiveness of our method in a multi-robot rendezvous problem. The simulation results show that our method efficiently learn decentralized control policies with substantially less training samples.

The main contribution of this letter is the multi-robot guided policy search (MRGPS) method. Specifically, 1) a distributed trajectory optimization based on linear-quadratic-regulator (LQR) is designed for the guiding trajectory generation in policy training. This distributed trajectory optimization extends the single-robot guided policy search [13], [14] to multi-robot systems. 2) The learned decentralized policy achieves the robotic swarm control goal using the robot's local observation only and does not require inter-robot communication. 3) The proposed method provides a new framework for learning decentralized multi-robot control policies, and demonstrates sample-efficiency comparing to existing supervised learning methods. Note that a distributed variant of GPS is presented in [16], where multiple robots collect and share training samples in a distributed and asynchronous manner to jointly learn an optimal policy. Similar framework is also reported in [17]. However, the policy learning algorithms in these work essentially address single-robot control problems. In contrast, our method is designed to solve multi-robot control problems.

The reminder of this letter is organized as follows. Section II presents the problem formulation of the multi-robot guided policy search for decentralized swarm control. Section III provides the design details of the proposed policy learning algorithm. The experimental validation and evaluation are presented in Section IV. We conclude our work and discuss the future work in Section V.

This work was partially supported by the US National Science Foundation under Grant CMMI-1825709.

C. Jiang is with the Department of Electrical and Computer Engineering, University of Wyoming, Laramie, WY, 82071 USA e-mail: cjiang1@uwyo.edu.

Y. Guo is with the Department of Electrical and Computer Engineering, Stevens Institute of Technology, Hoboken, NJ 07030, USA e-mail: yguo1@stevens.edu.

II. Problem Formulation and Learning Framework

A. Multi-Robot Swarm Control

We consider a multi-robot system with N robots. To focus on high-level motion control, we assume that the dynamics of each robot is described by the discrete-time model

$$\boldsymbol{x}_i(t+1) = \boldsymbol{x}_i(t) + \boldsymbol{u}_i(t). \tag{1}$$

where $\boldsymbol{x}_i \in \mathbb{R}^2$ and $\boldsymbol{u}_i \in \mathbb{R}^2$ are the 2-dimensional position and control of the i-th robot, respectively.

We study a multi-robot rendezvous problem where the objective is that given any initial positions, the robots aggregate together such that the multi-robot team converges to a common position, i.e., $(\boldsymbol{x}_i - \boldsymbol{x}_j) \to 0$ as $t \to \infty$, and the robot velocities converge to 0, i.e., $\boldsymbol{u}_i \to \boldsymbol{u}_j \to 0$ as $t \to \infty$. We formulate the robotic swarm control as an optimization problem where the goal is to find the optimal robot trajectories $\{\boldsymbol{x}_i(t), \boldsymbol{u}_i(t)\}_{t=0}^{T}$ with the initial position $\boldsymbol{x}_i(0) = \boldsymbol{x}_i^{init}$, for $i = 1, .., N$, which minimize the following objective function:

$$\begin{aligned} \min_{\boldsymbol{X},\boldsymbol{U}} \quad & \sum_{i=1}^{N}\sum_{t=0}^{T} l\big(\boldsymbol{x}_i(t), \boldsymbol{u}_i(t)\big) = \\ & \sum_{i,t}\left(\sum_{j\in\boldsymbol{R}(i)} \|\boldsymbol{x}_i(t) - \boldsymbol{x}_j(t)\|_2^2 + \|\boldsymbol{u}_i(t)\|_2^2\right) \\ \text{s.t.} \quad & \boldsymbol{x}_i(t+1) = \boldsymbol{x}_i(t) + \boldsymbol{u}_i(t); \boldsymbol{x}_i(0) = \boldsymbol{x}_i^{init} \end{aligned} \tag{2}$$

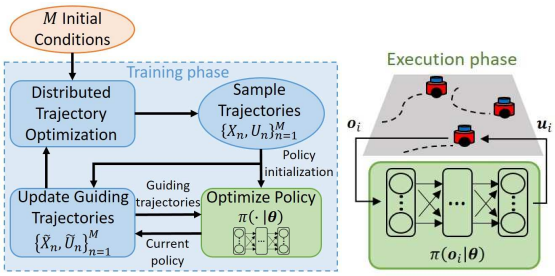
where $\boldsymbol{R}(i)$ is the set of neighboring robots of the i-th robot. We use $\{\boldsymbol{X}, \boldsymbol{U}\}$ to denote the set of trajectories of all N robots. The first term in the objective function forces the robots to minimize the relative distance to its neighboring robot $j \in \boldsymbol{R}(i)$ and the second term minimizes the control input of each robot whose velocity will converge to 0.

B. Decentralized Policy for Robotic Swarm Control

We aim to learn the decentralized control policy $\pi\big(\boldsymbol{o}_i|\boldsymbol{\theta}\big) = \boldsymbol{u}_i$, parameterized by $\boldsymbol{\theta}$, such that for any initial condition $\boldsymbol{x}_i^{init}$ of each robot, the control policy computes the velocity control output $\boldsymbol{u}_i$ from the robot's *local observation* $\boldsymbol{o}_i$ to achieve the objective of the swarm control defined in (2). The observation vector of each robot i, $\boldsymbol{o}_i = [\Delta_{i,i1}^T, ..., \Delta_{i,ij}^T]^T$, includes the relative position $\Delta_{i,ij}$ of its neighboring robot $j \in \boldsymbol{R}(i)$, with respect to the i-th robot's local coordinate system. It is worth noting that the robot's control policy requires neither the knowledge of global coordinate system nor the communication to its neighboring robots to acquire information regarding their global positions. Learning the control policy essentially is to find the optimal policy parameters $\boldsymbol{\theta}$ that achieve the performance objective averaged over M samples, that is

$$\begin{aligned} \min_{\boldsymbol{X},\boldsymbol{U}} \quad & \frac{1}{M}\sum_{n=1}^{M}\left[\sum_{i=1}^{N}\sum_{t=0}^{T} l\Big(\boldsymbol{x}_{n,i}(t), \pi\big(\boldsymbol{o}_{n,i}(t)|\boldsymbol{\theta}\big)\Big)\right] \\ \text{s.t.} \quad & \boldsymbol{x}_{n,i}(t+1) = \boldsymbol{x}_{n,i}(t) + \pi\big(\boldsymbol{o}_{n,i}(t)|\boldsymbol{\theta}\big); \boldsymbol{x}_i(0) = \boldsymbol{x}_i^{init} \end{aligned} \tag{3}$$

A control policy $\pi\big(\boldsymbol{o}_i|\boldsymbol{\theta}\big)$ can be constructed from expert demonstration through supervised learning of behaviors (e.g., [18], [19]). However, supervised learning is prone to compounding errors that deteriorate the long-term performance



(a) (b)

Fig. 1: The overview of the proposed multi-robot learning method based on guided policy search: (a) training phase; (b) execution phase.

[20]. Furthermore, to minimize the effect of compounding errors, a substantial number of training samples are required to reconstruct the control policy. In this letter we propose a novel multi-robot learning method based on guided policy search for learning decentralized control of robotic swarm.

C. Guided Policy Search via Trajectory Optimization

An overview of the multi-robot learning framework is shown in Fig. 1. We adopt the centralized training and decentralized execution paradigm. The centralized training is performed by the proposed MRGPS as shown in Fig. 1a. Specifically, the policy training is guided by the distributed trajectory optimization which, for each initial condition $n = 1, ..., M$, solves for the sample trajectories $\{\boldsymbol{X}_n, \boldsymbol{U}_n\}$ of all N robots. The sample trajectories provide a constricted search space for policy training, which avoids unnecessary exploration in the search space. The initial sample trajectories are used to initialize the policy π. The learned policy is exploited to update the guiding trajectories $\{\tilde{\boldsymbol{X}}_n, \tilde{\boldsymbol{U}}_n\}_{n=1}^{M}$ which in turn plays a part in the trajectory optimization such that the resulting sample trajectories are adapted to the current policy. Therefore, the guiding trajectories should accomplish the control objective and meanwhile should be reproducible by the learned policy. The learning process alternates between the policy optimization using the guiding trajectories and the distributed trajectory optimization with an augmented objective making the resulting sample trajectories resemble the output of the current policy. In such a manner, the policy will be eventually reconstructed from the guiding trajectories that are reproducible by the policy, and thus avoids the compounding errors issue. During online execution, as shown in Fig. 1b, the learned policy π is deployed on each robot to compute the decentralized control solely from the robot's local observation. The robots share the same policy parameters learned by the algorithm.

Remark 1: The sample-efficiency of the proposed method is achieved by using the distributed trajectory optimizer that iteratively constructs successful sample trajectories to direct the policy search to high-reward regions. Moreover, the trajectory optimizer iteratively adapts to the learned policy such that the training samples eventually come from the policy's own exploration.

Remark 2: It is worth noting that with the known model of robot dynamics in (1), one could directly solve the multi-robot rendezvous problem as formulated in (2) via distributed trajectory optimization. However, solving such a distributed trajectory optimization is time consuming and may not meet the online execution need. It also requires inter-robot communication to obtain neighboring robots' global positions. In contrast, our proposed approach learns neural network policies so that: 1) given any initial conditions, our approach produces robot control on-the-fly to achieve decentralized rendezvous control; and 2) our approach does not need inter-robot communication and can rely on the robot's local observation through sensing only.

More formally, the guided policy learning can be formulated as the following optimization problem:

$$\begin{aligned} \min_{\boldsymbol{X},\boldsymbol{U},\boldsymbol{\theta}} \quad & \frac{1}{M}\sum_{n=1}^{M} L(\boldsymbol{X}_n,\boldsymbol{U}_n) + R(\boldsymbol{X},\boldsymbol{U},\boldsymbol{\theta}) \\ \text{s.t.} \quad & \boldsymbol{x}_{n,i}(t+1) = \boldsymbol{x}_{n,i}(t) + \boldsymbol{u}_{n,i}(t); \boldsymbol{x}_{n,i}(0) = \boldsymbol{x}_{n,i}^{init} \end{aligned} \tag{4}$$

where $L(\boldsymbol{X}_n,\boldsymbol{U}_n) = \sum_{i,t} l\big(\boldsymbol{x}_{n,i}(t),\boldsymbol{u}_{n,i}(t)\big)$ is the cost of the swarm control, and $\{\boldsymbol{X}_n,\boldsymbol{U}_n\}$ denotes the trajectories of all N robots of the n-th sample, and

$$R(\boldsymbol{X},\boldsymbol{U},\boldsymbol{\theta}) = \sum_{n=1}^{M}\sum_{i=1}^{N}\sum_{t=0}^{T}\frac{1}{2}\left\|\pi\big(\boldsymbol{x}_{n,i}(t)|\boldsymbol{\theta}\big) - \boldsymbol{u}_{n,i}(t)\right\|_2^2 \tag{5}$$

is the regression cost of reconstructing the decentralized control policy. The solution to the above optimization problem is the decentralized control policy π, parameterized by the optimal parameters $\boldsymbol{\theta}^*$. Note that our goal is to learn the control policy that takes as input the robot local observation, $\boldsymbol{o}_i$, rather than the robot state, $\boldsymbol{x}_i$. Thus, $\pi(\boldsymbol{x}_{n,i}|\boldsymbol{\theta})$ in (5) will be evaluated with samples of robot local observations taken at the corresponding robot states, i.e., $\pi(\boldsymbol{o}_{n,i}|\boldsymbol{\theta})$. In the next section, we present the proposed MRGPS method that solves the optimization problem (4) for the swarm control policy.

III. Multi-Robot Guided Policy Search

A. Algorithm Overview

The MRGPS algorithm employs the alternating optimization scheme of ADMM [15] to solve (4). By introducing new variables $\tilde{\boldsymbol{X}}$ and $\tilde{\boldsymbol{U}}$, (4) is then reformulated as a constrained problem whose augmented Lagrangian can be written as

$$\begin{aligned} \mathcal{L}(\boldsymbol{\lambda}^x,\boldsymbol{\lambda}^u) = \arg\min \frac{1}{M}\sum_{n=1}^{M} L(\boldsymbol{X}_n,\boldsymbol{U}_n) + R(\tilde{\boldsymbol{X}},\tilde{\boldsymbol{U}},\boldsymbol{\theta}) + \\ \sum_{n=1}^{M}\Big(\frac{\rho}{2}\|\boldsymbol{X}_n - \tilde{\boldsymbol{X}}_n + \boldsymbol{\lambda}_{\boldsymbol{n}}^{\boldsymbol{x}}\|_2^2 + \frac{\rho}{2}\|\boldsymbol{U}_n - \tilde{\boldsymbol{U}}_n + \boldsymbol{\lambda}_{\boldsymbol{n}}^{\boldsymbol{u}}\|_2^2\Big) \end{aligned} \tag{6}$$

where $\boldsymbol{\lambda}^x$, $\boldsymbol{\lambda}^u$ are the Lagrangian multipliers; ρ is a constant; $\boldsymbol{X}_n$, $\boldsymbol{U}_n$ and $\tilde{\boldsymbol{X}}$, $\tilde{\boldsymbol{U}}$ are the trajectories minimizing the trajectory optimization cost and the policy optimization cost in (4), respectively. Then, the algorithm solves (6) by alternating between 1) the trajectory optimization with two additional quadratic objectives and 2) the policy optimization using the guiding trajectories.

The trajectories $\boldsymbol{X}_n$, $\boldsymbol{U}_n$ of the n-th sample is solved via the trajectory optimization:

$$\begin{aligned} \min_{\boldsymbol{X}_n,\boldsymbol{U}_n} \quad & \sum_{i=1}^{N}\sum_{t=0}^{T} l\big(\boldsymbol{x}_{n,i}(t),\boldsymbol{u}_{n,i}(t)\big) + \frac{\rho}{2}\|\boldsymbol{x}_{n,i}(t) - \tilde{\boldsymbol{x}}_{n,i}(t) + \\ & \boldsymbol{\lambda}_{\boldsymbol{n},\boldsymbol{i}}^{\boldsymbol{x}}(t)\|_2^2 + \frac{\rho}{2}\|\boldsymbol{u}_{n,i}(t) - \tilde{\boldsymbol{u}}_{n,i}(t) + \boldsymbol{\lambda}_{\boldsymbol{n},\boldsymbol{i}}^{\boldsymbol{u}}(t)\|_2^2 \\ \text{s.t.} \quad & \boldsymbol{x}_{n,i}(t+1) = \boldsymbol{x}_{n,i}(t) + \boldsymbol{u}_{n,i}(t); \boldsymbol{x}_{n,i}(0) = \boldsymbol{x}_{n,i}^{init} \end{aligned} \tag{7}$$

The optimization problem is the swarm control problem (2) with two additional quadratic costs that penalize the deviation from the guiding trajectories used to train the current policy. This ensures that the trajectories obtained via the trajectory optimization are reproducible by the current policy. We solve the optimization problem by the distributed trajectory optimization approach discussed in Section III-B.

In the policy optimization, the parameters $\boldsymbol{\theta}$ are optimized by solving the regression problem:

$$\min_{\boldsymbol{\theta}} R(\tilde{\boldsymbol{X}},\tilde{\boldsymbol{U}},\boldsymbol{\theta}) \tag{8}$$

given the guiding trajectories $\tilde{\boldsymbol{X}}$, $\tilde{\boldsymbol{U}}$ that provide the supervision data for training. As discussed in Section II-C, the policy function is evaluated using the robot's local observation $\boldsymbol{o}_i(t)$ sampled at the corresponding robot state $\tilde{\boldsymbol{x}}_i(t)$ from the guiding trajectories.

The guiding trajectories $\tilde{\boldsymbol{X}}_n$, $\tilde{\boldsymbol{U}}_n$ of the n-th sample are then updated by minimizing the following cost:

$$\begin{aligned} \min_{\tilde{\boldsymbol{X}}_n,\tilde{\boldsymbol{U}}_n} \sum_{i=1}^{N}\sum_{t=0}^{T}\frac{1}{2}\left\|\pi\big(\boldsymbol{o}_{n,i}(t)|\boldsymbol{\theta}\big) - \tilde{\boldsymbol{u}}_{n,i}(t)\right\|_2^2 + \frac{\rho}{2}\|\boldsymbol{x}_{n,i}(t) - \\ \tilde{\boldsymbol{x}}_{n,i}(t) + \boldsymbol{\lambda}_{\boldsymbol{n},\boldsymbol{i}}^{\boldsymbol{x}}(t)\|_2^2 + \frac{\rho}{2}\|\boldsymbol{u}_{n,i}(t) - \tilde{\boldsymbol{u}}_{n,i}(t) + \boldsymbol{\lambda}_{\boldsymbol{n},\boldsymbol{i}}^{\boldsymbol{u}}(t)\|_2^2 \end{aligned} \tag{9}$$

This minimization ensures that the guiding trajectories generated by the distributed trajectory optimization (7) resemble the outputs of the current policy, making the guiding trajectories reproducible by the current policy. By alternately optimizing (7)$\sim$(9), the trajectory optimizer that produces the guiding trajectories and the current policy are adapted to each other and eventually exhibit the same behavior at convergence.

Finally, $\boldsymbol{\lambda}^{\boldsymbol{x}}$ and $\boldsymbol{\lambda}^{\boldsymbol{u}}$ are updated by

$$\boldsymbol{\lambda}^{\boldsymbol{x}} = \boldsymbol{\lambda}^{\boldsymbol{x}} + (\boldsymbol{X} - \tilde{\boldsymbol{X}}); \boldsymbol{\lambda}^{\boldsymbol{u}} = \boldsymbol{\lambda}^{\boldsymbol{u}} + (\boldsymbol{U} - \tilde{\boldsymbol{U}}) \tag{10}$$

Thus, the optimization problem defined in (4) is solved by recursively solving (7)$\sim$(10) in an alternating manner. The proposed MRGPS algorithm is summarized in Algorithm 1.

Remark 3: The proposed MRGPS algorithm exploits the ADMM framework and thus has the same convergence guarantees. At convergence, the algorithm converges to the solution where the learned policy can be regarded as an approximate optimal controller, thus it not only succeeds from all initial states in training samples, but also generalizes to various new initial states. In fact, unlike the standard supervised learning that trains a policy model on a fixed set of training samples, the MRGPS constantly explores in high-reward regions provided by the trajectory optimizer. Greater quantity and diversity of the samples explored during training would help achieve better generalizability to new initial states.

In the subsequent subsections, we present the *distributed trajectory optimization* that solves the subproblem (7) and the *policy optimization* that solves the subproblem (8).

Algorithm 1: Multi-Robot Guided Policy Search

Input : M initial conditions $\{\boldsymbol{X}_n^{init}\}_{n=1}^M$;
Output: Decentralized control policy $\pi(\boldsymbol{o}_i|\boldsymbol{\theta})$;
1 Generate M sets of sample trajectories with distributed trajectory optimization, given the initial conditions $\{\boldsymbol{X}_n^{init}\}_{n=1}^M$;
2 Initialize the guiding trajectories $\tilde{\boldsymbol{X}}$, $\tilde{\boldsymbol{U}}$ with the trajectory optimization solution;
3 **while** *not converged* **do**
4 Optimize the trajectories of all robots, $\boldsymbol{X}_n$, $\boldsymbol{U}_n$, by solving (7) via the distributed trajectory optimization for samples $n = 1, ..., M$;
5 Optimize the policy parameter $\boldsymbol{\theta}$ by solving the regression problem (8) with the guiding trajectories $\tilde{\boldsymbol{X}}$, $\tilde{\boldsymbol{U}}$;
6 Update the guiding trajectories $\tilde{\boldsymbol{X}}$, $\tilde{\boldsymbol{U}}$ by solving the optimization problem (9);
7 Update the Lagrangian multipliers $\boldsymbol{\lambda}^{\boldsymbol{x}}$, $\boldsymbol{\lambda}^{\boldsymbol{u}}$ using (10);
8 **end**

B. Distributed Trajectory Optimization

The problem defined in (7) is essentially a multi-robot trajectory optimization with an augmented cost function. To solve (7), we design a distributed trajectory optimization algorithm which combines the distributed optimization algorithm [21] and the trajectory optimization based on the linear-quadratic regulator (LQR) [22], assuming a known robot dynamics model. Thereby, we extend the single-agent guided policy search method to multi-robot learning problems.

The cost function in (7) can be further written as (here the subscript n denoting the n-th sample is omitted)

$$\begin{aligned}&\min_{\boldsymbol{X},\boldsymbol{U}} \sum_{i=1}^{N}\sum_{t=1}^{T}\sum_{j\in\boldsymbol{R}(i)} \|\boldsymbol{x}_i(t)-\boldsymbol{x}_j(t)\|_2^2 + \|\boldsymbol{u}_i(t)\|_2^2 + \\ &\frac{\rho}{2}\|\boldsymbol{x}_i(t)-\tilde{\boldsymbol{x}}_i(t)+\boldsymbol{\lambda}_{\boldsymbol{i}}^{\boldsymbol{x}}(t)\|_2^2 + \frac{\rho}{2}\|\boldsymbol{u}_i(t)-\tilde{\boldsymbol{u}}_i(t)+\boldsymbol{\lambda}_{\boldsymbol{i}}^{\boldsymbol{u}}(t)\|_2^2\end{aligned} \tag{11}$$

where $\sum_{j\in\boldsymbol{R}(i)} \|\boldsymbol{x}_i-\boldsymbol{x}_j\|_2^2$ is a cross-coupled term. To decentralize the optimization, we adopt the distributed optimization method [21] where the cost function in (11) is decoupled using the dual decomposition method and then the resulting dual problems are solved for the optimal trajectories $[\boldsymbol{X}^*, \boldsymbol{U}^*]$. Specifically, after decoupling the Lagrangian and the dual function of the dual problem, (11) is equivalent to the following optimization problem:

$$\begin{aligned}&\min_{\boldsymbol{X},\boldsymbol{U}} \sum_{i=1}^{N}\sum_{t=1}^{T}\sum_{j\in\boldsymbol{R}(i)} \Big(\boldsymbol{d}_{ij}^T(t)-\boldsymbol{d}_{ji}^T(t)\Big)\boldsymbol{x}_i(t) + \|\boldsymbol{u}_i(t)\|_2^2 + \\ &\frac{\rho}{2}\|\boldsymbol{x}_i(t)-\tilde{\boldsymbol{x}}_i(t)+\boldsymbol{\lambda}_{\boldsymbol{i}}^{\boldsymbol{x}}(t)\|_2^2 + \frac{\rho}{2}\|\boldsymbol{u}_i(t)-\tilde{\boldsymbol{u}}_i(t)+\boldsymbol{\lambda}_{\boldsymbol{i}}^{\boldsymbol{u}}(t)\|_2^2\end{aligned} \tag{12}$$

where $\boldsymbol{d}_{ij}(t)$ and $\boldsymbol{d}_{ji}(t)$ are the variables introduced in the dual problem, which reflect the deviation from the desired relative position, $(\boldsymbol{x}_i - \boldsymbol{x}_j)$, at time t, and are estimated by robot i and j, respectively. The cost function in (12) is decoupled such that each robot has its own private cost.

Thus, solving (12) is equivalent to finding the solution to the private optimization problem of each robot in a distributed manner through the following iteration:

For each robot $i = 1, ..., N$, initialize $\boldsymbol{d}_{ij}$ for all $j \in \boldsymbol{R}(i)$, and then:

1. Compute the optimized trajectory of the robot, $\boldsymbol{x}_i^*$, that minimizes the private cost for each $i = 1, ..., N$ in (12).
2. Update $\boldsymbol{d}_{ij} \leftarrow \boldsymbol{d}_{ij} + \alpha_k \left((\boldsymbol{x}_i^* - \boldsymbol{x}_j^*) - \frac{\boldsymbol{d}_{ij}}{2}\right)$.

Step 1 and 2 are repeated until the solution converges. Particularly, Step 1 can be solved by existing trajectory optimization algorithms with the robot dynamics described in (1). We exploit the LQR-based trajectory optimizer here.

C. Policy Optimization via Regression

Learning the decentralized control policy $\pi(\boldsymbol{o}_i|\boldsymbol{\theta})$ essentially is to solve the following regression problem:

$$\min_{\boldsymbol{\theta}} \sum_{n=1}^{M}\sum_{i=1}^{N}\sum_{t=1}^{T} \frac{1}{2}\left\|\pi(\boldsymbol{o}_{n,i}(t)|\boldsymbol{\theta}) - \boldsymbol{u}_{n,i}(t)\right\|_2^2 \tag{13}$$

with the input being the local observation of each robot, $\boldsymbol{o}_i(t)$, and the target output being the desired control $\boldsymbol{u}_i(t)$. The pairs of input, $\boldsymbol{o}_{n,i}(t)$, and target output, $\boldsymbol{u}_{n,i}(t)$, are provided by the guiding trajectories $\tilde{\boldsymbol{X}}_n$, $\tilde{\boldsymbol{U}}_n$.

We use a feed-forward neural network to approximate the policy function $\pi(\boldsymbol{o}_i|\boldsymbol{\theta})$. The neural network is composed of multiple hidden layers. The parameters $\boldsymbol{\theta}$ include the weights and bias of the neural network. A supervised learning with stochastic gradient descent algorithm is used to fit the policy function to the current guiding trajectory samples $\tilde{\boldsymbol{X}}, \tilde{\boldsymbol{U}}$ by minimizing the loss function (13).

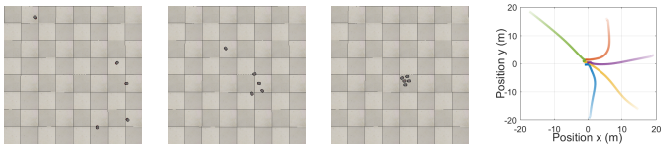
IV. Simulation Results

A. Experiment Setup

1) Robot Simulation: The simulation experiments were conducted in V-REP robot simulator [23]. A 40×40 m^2 square region was created as the simulated environment. The differential drive robot Pioneer P3-DX was selected as the mobile robot. The robot control algorithm was implemented in a MATLAB client program which communicates with V-REP via remote API. The number of robots was set to $N = 5$. The sampling period of the simulation was 0.1 s.

Note that the policy learned using our MRGPS method computes the high-level control $\boldsymbol{u}_i(t)$ in Eq. (1). To control the differential drive robot in the simulator, we applied the coordinate transformation [24] to transform $\boldsymbol{u}_i(t)$ to the wheel speed control of the differential drive robot. This method has been used in our previous work [25], [26].

2) Neural Network Implementation: The policy function is approximated using a fully connected neural network with two hidden layers. Both hidden layers have 32 hidden nodes and are activated by a sigmoid function. The dimension of the input layer and the output layer is 8 and 2, respectively. A linear activation function is applied to the output layer. The selection of the hyperparameters is a result of balancing the policy fitting accuracy and the variance of predicted control outputs. The training and numerical computation of the neural network were implemented using the MATLAB neural network training toolbox.



(a) t=20 s (b) t=30 s (c) t=50 s (d) trajectories

Fig. 2: Snapshots of the online swarm control simulation.

B. Policy Learning

The MRGPS algorithm described in Alg. 1 was carried out to learn the decentralized policy. The parameter ρ was selected as 2. The Lagrangian multipliers $\boldsymbol{\lambda}^x$ and $\boldsymbol{\lambda}^x$ were randomly initialized between $[0, 1]$. In the distributed trajectory optimization algorithm, the initial estimate of the relative position $\boldsymbol{d}_{ij}$ was randomly initialized between $[0, 20]$ in both directions. The step-size for updating the estimate $\boldsymbol{d}_{ij}$ at each iteration k was chosen as $\alpha_k = 0.001$. To verify the sample-efficiency of the proposed method, we trained the policies with $M = 10$ and $M = 100$ samples, respectively, and compare their performance in online test experiments. Each sample contains randomly initialized robot positions and consists of 100 time steps. The training using 10 samples converges with a policy fitting error about 5×10^{-4} after 9 iterations of Alg. 1. The training using 100 samples converges with a policy fitting error about 9×10^{-4} after 12 iterations.

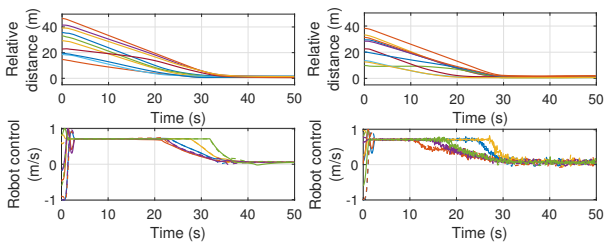
The computational cost of Alg. 1 is evaluated with $M = 100$ on a computer with an Intel® Xeon® E5-1620 (3.5 GHz) CPU and 16 GB RAM. The average time consumed for each portion of the algorithm per iteration is as follows: 131.2 s for trajectory optimization (line 4), 551.3 s for policy optimization (line 5), and 412.2 s for guiding trajectory update (line 6).

C. Testing and Performance Evaluation

1) Online Swarm Control: In the online swarm control experiments, the learned decentralized policy was deployed on each robot. At each time step, the policy computes the robot control from the its local observation only. The robots' positions were randomly initialized in the 40×40 m^2 square region. The duration of each simulation was set to 50 s.

The snapshots of the online swarm control experiment are presented in Fig. 2, which show the robot positions at different time steps. Fig. 2d shows the robot trajectories, whose moving directions are indicated by the increasing degree of opacity. One can see that the robots start from different initial positions and successfully complete the rendezvous task. The time history of the relative distance between any two robots is shown in the upper figure of Fig. 3a. We can see that the relative distance converges after about 30 s. The time history of the robot wheel speed control are shown in the lower figure of Fig. 3a, where the solid and dashed lines denote the right and left wheel speed control, respectively. The robot control signals all converge to 0 after about 40 s.

We also tested the robustness of our method under noisy observations. A Gaussian noise with mean $\mu = 0$ m and standard deviation $\sigma = 0.2$ m was added to each robot's local observation of the neighboring robots' relative positions, $\Delta_{i,ij}$. The results of online swarm control under noisy observations



(a) (b)

Fig. 3: Online swarm control results (a) without and (b) with observation noise.

are shown in Fig. 3b. One can see that the computed control drives the robots to achieve the rendezvous task. The policy is trained without observation noise but is tolerant to moderate observation noise during online execution.

2) Performance Evaluation: We evaluate our method by conducting extensive experiments and compare the performance with the standard supervised learning method adopted in existing work such as [18], [19]. The supervised learning uses the same neural network architecture as the proposed MRGPS algorithm. The trajectory optimizer in Section III-B is used to generate successful trajectories as the training set which includes robot local observations and target control at every time steps of the trajectories. The neural network policy is trained on the training set once, using mini-batch stochastic gradient decent to minimize the mean squared error between the network outputs and the target outputs. Note that the supervised learning only solves the regression problem (line 5 of Alg. 1) once, while our MRGPS iteratively generates reproducible sample trajectories (lines 4 and 6 of Alg. 1) to guide the policy search.

To compare sample-efficiency, we use $M = 10$ and $M = 100$ samples for both methods to train the swarm control policies and compare the online swarm control performance. We present the statistical results of 100 simulation runs and compare the success rate and the convergence time. A simulation run is considered successful if for all robots $i \in N$, the distance to the nearest robot j, $\|\boldsymbol{x}_i - \boldsymbol{x}_j\|$, is less than 0.5 m after convergence within 50 s.

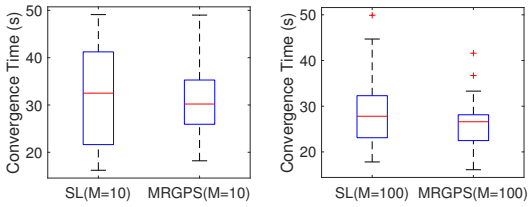
Table I shows the success rate of our proposed method compared with the supervised learning. One can see that our method can achieve 91% success rate with only 10 samples. The success rate increases to 97% with 100 samples. On the contrary, the success rate of the supervised learning is 77% with 10 samples and 90% with 100 samples, respectively. Compared with the supervised learning, our method is more sample-efficient and attains good success rate even with a small number of samples for training. As for the unsuccessful runs of our method, the relative distances between the robots are converging, but do not meet the requirement of "less than 0.5 m after convergence within 50 s" as defined above.

We also analyze the convergence time of the online swarm control using our MRGPS and the supervised learning method. We consider that the swarm control converges at the time when, for all robots, the distance to its nearest robot is less than 0.5 m. Fig. 4 shows the statistical results of convergence

TABLE I: Success rate over 100 test runs.

	Our MRGPS method		Supervised learning	
	$M = 10$	$M = 100$	$M = 10$	$M = 100$
Success rate	91%	97%	77%	90%

* M is the number of samples used to train the policy.



(a) (b)

Fig. 4: Convergence time over successful runs with (a) M=10 and (b) M=100. SL and MRGPS stand for supervised learning and our multi-robot guided policy search, respectively.

time over successful runs, where the central mark in each box is the median, the edges of the boxes are the 25th and 75th percentiles, respectively, the whiskers extend to the max/min, and the cross markers represent outliers. It can be seen that 1) given the same number of training samples M, the control policy trained using our MRGPS method converges faster than that using the supervised learning; 2) For both methods, the more training samples are used, the faster the online control converges.

V. Conclusions and Future Work

In this letter, we proposed a novel multi-robot guided policy search method for learning decentralized control of robotic swarm. The policy learning is guided by the guiding trajectories generated by the distributed trajectory optimization. Meanwhile, the guiding trajectories are updated to be reproducible by the learned policy. Thus, the proposed multi-robot guided policy search alternates between a supervised policy training and a distributed trajectory optimization with an augmented objective for the resulting guiding trajectories to adapt to the current policy. Simulation experiments in a robot simulator were conducted to verify and evaluate our method, and the results demonstrated superior performance and sample-efficiency over existing supervised learning.

The proposed method adopts a distributed trajectory optimization that operates on known system dynamics. In future work, we will investigate the multi-robot guided policy search under approximate or unknown system dynamics. Furthermore, we will investigate end-to-end learning where the learned policy operates on low-level robot perceptions such as raw sensor measurements with limited robot local observation of nearby robots.

References

[1] M. L. Littman, "Markov games as a framework for multi-agent reinforcement learning," in *Machine Learning Proceedings*, pp. 157–163, 1994.

[2] D. S. Bernstein, R. Givan, N. Immerman, and S. Zilberstein, "The complexity of decentralized control of markov decision processes," *Mathematics of Operations Research*, vol. 27, no. 4, pp. 819–840, 2002.

[3] M. Tan, "Multi-agent reinforcement learning: Independent vs. cooperative agents," in *Proceedings of the International Conference on Machine Learning*, pp. 330–337, 1993.

[4] G. Tesauro, "Extending q-learning to general adaptive multi-agent systems," in *Advances in Neural Information Processing Systems*, pp. 871–878, 2004.

[5] K. H. Low, W. K. Leow, and M. H. Ang, "Autonomic mobile sensor network with self-coordinated task allocation and execution," *IEEE Transactions on Systems, Man, and Cybernetics, Part C*, vol. 36, no. 3, pp. 315–327, 2006.

[6] M. Otte, "An emergent group mind across a swarm of robots: Collective cognition and distributed sensing via a shared wireless neural network," *The International Journal of Robotics Research*, vol. 37, no. 9, pp. 1017–1061, 2018.

[7] M. Hüttenrauch, A. Šošić, and G. Neumann, "Guided deep reinforcement learning for swarm systems," *arXiv:1709.06011*, 2017.

[8] J. K. Gupta, M. Egorov, and M. Kochenderfer, "Cooperative multi-agent control using deep reinforcement learning," in *International Conference on Autonomous Agents and Multiagent Systems*, pp. 66–83, 2017.

[9] R. Lowe, Y. Wu, A. Tamar, J. Harb, O. P. Abbeel, and I. Mordatch, "Multi-agent actor-critic for mixed cooperative-competitive environments," in *Advances in Neural Information Processing Systems*, pp. 6379–6390, 2017.

[10] J. N. Foerster, G. Farquhar, T. Afouras, N. Nardelli, and S. Whiteson, "Counterfactual multi-agent policy gradients," in *AAAI Conference on Artificial Intelligence*, pp. 2974–2892, 2018.

[11] S. Li, Y. Wu, X. Cui, H. Dong, F. Fang, and S. Russell, "Robust multi-agent reinforcement learning via minimax deep deterministic policy gradient," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, pp. 4213–4220, 2019.

[12] Y. Wu, E. Mansimov, R. B. Grosse, S. Liao, and J. Ba, "Scalable trust-region method for deep reinforcement learning using kronecker-factored approximation," in *Advances in Neural Information Processing Systems*, pp. 5279–5288, 2017.

[13] S. Levine and V. Koltun, "Guided policy search," in *International Conference on Machine Learning*, pp. 1–9, 2013.

[14] I. Mordatch and E. Todorov, "Combining the benefits of function approximation and trajectory optimization," in *Robotics: Science and Systems*, vol. 4, 2014.

[15] S. Boyd, N. Parikh, E. Chu, B. Peleato, J. Eckstein, *et al.*, "Distributed optimization and statistical learning via the alternating direction method of multipliers," *Foundations and Trends® in Machine Learning*, vol. 3, no. 1, pp. 1–122, 2011.

[16] A. Yahya, A. Li, M. Kalakrishnan, Y. Chebotar, and S. Levine, "Collective robot reinforcement learning with distributed asynchronous guided policy search," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 79–86, 2017.

[17] S. Levine, P. Pastor, A. Krizhevsky, and D. Quillen, "Learning hand-eye coordination for robotic grasping with large-scale data collection," in *International Symposium on Experimental Robotics*, pp. 173–184, 2016.

[18] P. Long, W. Liu, and J. Pan, "Deep-learned collision avoidance policy for distributed multiagent navigation," *IEEE Robotics and Automation Letters*, vol. 2, no. 2, pp. 656–663, 2017.

[19] C. Jiang, Z. Chen, and Y. Guo, "Learning decentralized control policies for multi-robot formation," in *IEEE/ASME International Conference on Advanced Intelligent Mechatronics*, pp. 758–765, 2019.

[20] S. Ross, G. Gordon, and D. Bagnell, "A reduction of imitation learning and structured prediction to no-regret online learning," in *Proceedings of the International Conference on Artificial Intelligence and Statistics*, pp. 627–635, 2011.

[21] R. L. Raffard, C. J. Tomlin, and S. P. Boyd, "Distributed optimization for cooperative agents: Application to formation flight," in *IEEE Conference on Decision and Control*, vol. 3, pp. 2453–2459, 2004.

[22] B. D. Anderson and J. B. Moore, *Optimal control: linear quadratic methods*. Courier Corporation, 2007.

[23] E. Rohmer, S. P. Singh, and M. Freese, "V-REP: A versatile and scalable robot simulation framework," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 1321–1326, 2013.

[24] A. De Luca, G. Oriolo, and M. Vendittelli, "Control of wheeled mobile robots: An experimental overview," in *Ramsete*, pp. 181–226, 2001.

[25] Z. Chen, C. Jiang, and Y. Guo, "Distance-based formation control of a three-robot system," in *Chinese Control and Decision Conference*, pp. 5501–5507, 2019.

[26] C. Jiang, Z. Chen, and Y. Guo, "Multi-robot formation control: a comparison between model-based and learning-based methods," *Journal of Control and Decision*, vol. 7, no. 1, pp. 90–108, 2020.