2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
October 14-18, 2024. Abu Dhabi, UAE

D3G: Learning Multi-robot Coordination from Demonstrations

Yizhi Zhou, Wanxin Jin and Xuan Wang

***Abstract*— This paper develops a new Distributed approach for solving the inverse problem of a Differentiable Dynamic Game (D3G), which enables robots to learn multi-robot coordination from given demonstrations. We formulate multi-robot coordination as the Nash equilibrium of a parameterized dynamic game, where the behavior of each robot is dictated by an objective function that also depends on the behavior of its neighboring robots. The coordination thus can be adapted by tuning the parameters of the objective and the local dynamics of each robot. The proposed algorithm enables each robot to automatically tune such parameters in a distributed and coordinated fashion — only using the data of its neighbors without global information. Its key novelty is the development of a distributed solver for a diff-KKT condition that can enhance scalability and reduce the computational load for gradient computation. We test the proposed algorithm in simulation with heterogeneous robots given different task configurations. The results demonstrate its effectiveness and generalizability for learning multi-robot coordination from demonstrations.**

I. Introduction

The control and coordination of large-scale multi-robot systems are challenging due to the need for sequential, coordinated decisions. Dynamic game theory provides a framework to model interactions among robots based on local observations and coupled objectives [2], [3]. Designing these objective functions is complex and often relies on heuristic methods. Alternatively, demonstrating desired behaviors is more intuitive, leading to the interest in inverse dynamic game (IDG) approaches to learn objectives from demonstrations [4]. While methods exist for single-robot cases, such as imitation learning and differentiable optimal control [5], scalable solutions for multi-robot systems are limited due to the high dimensionality. We propose a new Distributed Differentiable Dynamic Game (D3G) framework for solving IDG, enabling each robot to learn its objective function in a distributed manner using only local data. The core of our approach is a distributed solver that utilizes the differentiability of the KKT condition (diff-KKT) to improve scalability and reduce computational complexity. A conceptual diagram of D3G with a motivating example is in Fig. 1.

Learning from demonstrations can be formulated as a problem of inverse optimal control (IOC), also known as *Inverse reinforcement learning*, seeking to learn an objective function of a decision-making agent from expert demonstrations [6]. One type of method for solving IOC directly minimizes the residual of the optimality (KKT) conditions by assuming that the demonstration is optimal and fulfills these conditions [7]. Another common approach is built upon a bi-level structure, containing a forward loop and an inverse loop. The forward loop solves a standard optimal control problem with the current objective estimate. Available methods for this include dynamic programming [8], trajectory optimizations [9], and reinforcement learning [10]. The inverse loop updates the objective estimate such that a trajectory of the forward loop matches the demonstrations by minimizing certain losses. Different methods for IOC vary in how to accommodate the forward and inverse loops [11]–[13] , and also how to define loss functions, such as least square [11], [14], maximum margin [12], maximum entropy [13].

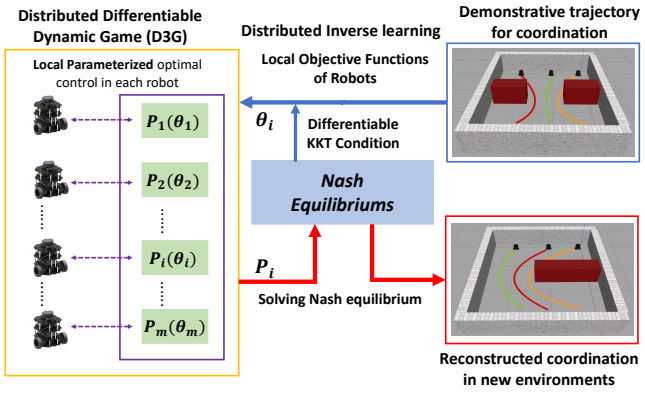


Fig. 1: Each robot possesses a local optimal control $\mathbf{P}_i$, which together constitutes a dynamic game. The Nash equilibrium of the game reconstructs robot coordination. Problem of interest: Distributed inverse learning (blue) of parameterized objective functions from demonstration for robot coordination. The learned objective is generalizable (red) to new environments.

Dynamic game generalizes optimal control to a multi-robot setup, where each robot's objective functions depend on its own action and the actions of other robots over time. Addressing such sequential decision-making processes often involves treating agents' entire state and action trajectories as variables [15], [16]. The set of robots' planned trajectories, when no one can improve its objective function by changing its behaviors, constitutes the solution to the game called open-loop Nash equilibrium [17]. Common approaches to obtaining a Nash equilibrium include: designing an algorithm whose dynamics asymptotically converge to the desired Nash equilibrium [18], [19]. To satisfy dynamics constraints, [20] introduced a projection operator that restricts the gradient flow to a feasible set, ensuring compliance with an agent's local constraints. An alternative approach is to compute Nash equilibrium directly from its holding conditions [21], which can be done by generalizing the Pontryagin's Maximum Principle/KKT condition [22] to a game theoretical setup.

Work supported by Army Research Office (W911NF-22-2-0242) and NSF (2332210). George Mason University. X. Wang and Y. Zhou are with the Department of Electrical and Computer Engineering, George Mason University. Wanxin Jin is with the School for Engineering of Matter, Transport, and Energy, Arizona State University. Extra experiments and complete proofs of this work appear in [1]. Point of contact: xwang64@gmu.edu.

Analogous to IOC, dynamic games also have their *inverse problem*, i.e., given robots collective trajectories satisfying a Nash equilibrium, how to inversely learn the objective functions the robots aim to optimize [4]. Existing works for solving inverse games have three main categories. The first category aims to solve the inverse game by applying derivative-free filter-based approaches built upon Bayesian inference [23], [24], which however has high sample complexity and requires exact observations of state. The second category solves the inverse game by equilibrium-constrained maximum-likelihood estimation (MLE), which uses the optimality conditions of the open-loop Nash equilibrium, to formulate a constrained optimization problem [25]. This type of method can explicitly handle noisy data and partial observations. The third category follows the minimization of residual methods [26], which seek to minimize the residual of the first-order necessary conditions of an open-loop Nash equilibrium. These works are further extended in [27], [28] to involve state and input constraints.

The approach proposed in this paper is similar to the ones in the last category [26], [28]. However, we note that existing methods for solving inverse dynamic games rely on a centralized process, where the forward loop and inverse loop are solved using the global information of all robots. Consequently, the computation and communication complexity grows exponentially with the number of robots and planning horizons. While there exist distributed approaches for solving the forward problem [15], [18], [29], the scalability challenge remains for addressing the inverse problem.

Statement of Contributions: We study the problem of learning multi-robot coordination from demonstration by formulating it as a differentiable dynamic game. Each robot in the game satisfies its dynamics and optimizes a coupling objective function. Both the dynamics and objective of each robot are unknown and learnable. We propose a D3G framework to inversely solve the dynamic game by minimizing the mismatch between the predicted multi-robot trajectories of the dynamic game and the given demonstrations. The learning update of D3G is based on local gradient descent. This allows a fully distributed algorithm design, where each robot uses the diff-KKT condition to compute its parameter update, by only using the data of its neighbors without global information. The effectiveness and scalability of D3G are verified using two types of robots given four different task configurations.

II. Preliminaries and Problem Formulation

A. Parametric Dynamic Game for Multi-robot Coordination

Consider a system of m robots. Suppose each robot solves its own optimal control problem $\mathbf{P}_i(\theta_i)$ parameterized by a vector $\theta_i \in \mathbb{R}^{r_i}$ as follows:

$$\begin{aligned} \min_{\mathbf{u}_i} \quad & \mathcal{J}_i(\theta_i) = \sum_{t=0}^{T-1} c_i^t(x_i^t, u_i^t, x_{\mathcal{N}_i}^t, \theta_i) + h_i(x_i^T, x_{\mathcal{N}_i}^T, \theta_i), \\ \text{s.t.} \quad & x_i^{t+1} = f_i(x_i^t, u_i^t, \theta_i) \quad \text{given} \quad x_i^0. \end{aligned} \tag{$\mathbf{P}_i(\theta_i)$}$$

Here, for robot i, $x_i^t \in \mathbb{R}^{n_i}$ and $u_i^t \in \mathbb{R}^{m_i}$ are the robot's state and control input at each time step $t \in \{0, 1, 2, \cdots, T\}$; $\mathbf{x}_i = \{x_i^0, \cdots, x_i^T\}$ and $\mathbf{u}_i = \{u_i^0, \cdots, u_i^{T-1}\}$; $f_i(\cdot) \in \mathbb{R}^{n_i}$ is the robot dynamics; $\mathcal{J}_i(\cdot) \in \mathbb{R}$ is the local control objective function with $c_i^t(\cdot) \in \mathbb{R}$ and $h_i(\cdot) \in \mathbb{R}$ denoting the running and final costs, respectively. To characterize the fact that connected robots have coordinated behaviors, their objective functions are set to be coupled, i.e., $\mathcal{J}_i(\cdot)$ depends not only on the state/input of robot i, but also on that of its neighbors, denoted by $x_{\mathcal{N}_i}^t = \{x_j^t \mid j \in \mathcal{N}_i\}$, with $\mathcal{N}_i$ being the neighbor set of robot i. The neighborhoods of robots define the communication topology $\mathbb{G}$ across the whole system, whose vertices are associated with the robots. We assume $\mathbb{G}$ is undirected. Further define $\boldsymbol{\xi}_i = \{\mathbf{x}_i, \mathbf{u}_i\}$, which represents the full trajectory of robot i for all time steps.

Since each robot only makes local observations, the collection of optimal control problems $\mathbf{P}_i(\theta_i)$ across all robots forms a *general-sum* dynamic game $\mathbf{P}(\Theta)$ parameterized by $\Theta = \operatorname{col}\{\theta_1, \cdots, \theta_m\} \in \mathbb{R}^{\sum_i^m r_i}$. Given the objective functions $\mathcal{J}_i(\theta_i)$ to be mutually coupled, the 'forward' (v.s. inverse) problem of the game $\mathbf{P}(\Theta)$ is to obtain a set of state-input-trajectories $\boldsymbol{\xi}_i^\star(\Theta) = \{\mathbf{x}_i^\star(\Theta), \mathbf{u}_i^\star(\Theta)\}$ for all $i \in \{1, \cdots, m\}$, called *open-loop Nash Equilibrium (N.E.)*[1], satisfying:

$$\begin{aligned} & \mathcal{J}_i(\boldsymbol{\xi}_i^\star(\Theta),\ \boldsymbol{\xi}_{\mathcal{N}_i}^\star(\Theta),\ \theta_i) \le \mathcal{J}_i(\boldsymbol{\xi}_i,\ \boldsymbol{\xi}_{\mathcal{N}_i}^\star(\Theta),\ \theta_i) \\ & \quad \text{s.t.} \qquad \boldsymbol{\xi}_i \in \Xi_i(\theta_i). \end{aligned} \tag{N.E.}$$

where $\Xi_i(\theta_i)$ is the set of all feasible trajectories of robot i satisfying its initial condition and system dynamics. $\Xi_i(\theta_i)$ is a function of θ_i because the dynamics $f_i(\cdot)$ is parameterized by θ_i. We use the (N.E.) of $\mathbf{P}(\Theta)$ to characterize distributed multi-robot coordination, where each robot determines its trajectory $\boldsymbol{\xi}_i^\star(\Theta)$ based on the local information of its neighboring robots. $\boldsymbol{\xi}_i^\star(\Theta)$ is a function of tunable Θ.

B. Problem Formulation

While lots of effort has been given to solve the 'forward' problem of $\mathbf{P}(\Theta)$, i.e., calculating its (N.E.) given robots' objective functions, this work focuses on the 'inverse' problem: *Which objective functions (the parameters for Θ) can reconstruct desired multi-robot coordination strategies that are aligned with given demonstrations.*

To this end, we first introduce the following assumption.

Assumption 1: Both $\mathcal{J}_i(\cdot)$ and $f_i(\cdot)$ are twice differentiable. Given other variables being fixed, the cost function $\mathcal{J}_i(\cdot)$ is strictly convex on x_i and u_i. The feasible trajectory set $\Xi_i(\theta_i)$ is convex and bounded.

Assumption 1 ensures the existence and uniqueness of a pure (N.E.) for $\mathbf{P}(\Theta)$ [30, Theorem 4.3]. It imposes some mild conditions on $f_i(\cdot)$ and $\mathcal{J}_i(\cdot)$, which are common in the existing literature for game-theoretic studies of multi-robot systems [15], [18], [29] . These conditions generally hold for physical models of simple mobile robots and regular cost functions such as distance to the goal. In the case that $\Xi_i(\theta_i)$ is unbounded, the existence and uniqueness can still be guaranteed [30, Corollary 4.2] if we further assume $\mathcal{J}_i(\cdot) \to \infty$ as $|x_i|$ or $|u_i| \to \infty$. This holds for most cost functions.

[1] In this paper, we refer to N.E. as an open-loop Nash equilibrium, in contrast to the feedback Nash equilibrium [30, Chapter 3].

Problem of interest: Given the demonstrations of robot trajectories $\{\boldsymbol{\xi}_1^d, \boldsymbol{\xi}_2^d, \cdots, \boldsymbol{\xi}_m^d\}$, $d \in \{1, \cdots, D\}$, that are associated with the (N.E.) of a game $\mathbf{P}(\Theta)$, with unknown Θ. Suppose each robot i locally knows $\mathbf{P}_i(\cdot)$ and $\boldsymbol{\xi}_i^d$. We aim to develop a fully distributed algorithm over $\mathbb{G}$ such that all robots jointly learn the parameter $\Theta^\star = \text{col}\,\{\theta_1^\star, \cdots, \theta_m^\star\}$ by minimizing the following loss function

$$\min_{\Theta = \text{col}\,\{\theta_1, \cdots, \theta_m\}} \sum_{i=1}^{m} \mathcal{L}_i(\boldsymbol{\xi}_i^\star(\Theta), \boldsymbol{\xi}_i^d). \tag{1}$$

The loss function in each robot is defined as

$$\mathcal{L}_i(\boldsymbol{\xi}_i^\star(\Theta), \boldsymbol{\xi}_i^d) = \sum_{d=1}^{D} \|\boldsymbol{\xi}_i^\star(\Theta) - \boldsymbol{\xi}_i^d\|_2^2 \tag{2}$$

By minimizing (1), we learn a proper $\Theta^\star$, i.e., $\theta_i^\star$ for each robot, to best mimic/reproduce the demonstrations (from experts) using the (N.E.) of the parameterized game. In the above definition of the loss (2), we consider the robot's trajectories at each time instant to be equally important, but other definitions of the loss [11]–[13] are also applicable.

III. Inverse Learning for Distributed Differential Dynamic Game

A. Method Overview

To solve the formulated problem, we develop a fully distributed learning paradigm, where each robot updates its own $\theta_i^\star$ for $\mathbf{P}_i(\theta_i)$ using only its local data and neighboring communication. We are enlightened by local gradient descent to propose the following algorithm,

$$\theta_i^{k+1} = \theta_i^k - \eta^k \left.\frac{d\mathcal{L}_i(\boldsymbol{\xi}_i^\star(\Theta), \boldsymbol{\xi}_i^d)}{d\theta_i}\right|_{\theta_i^k} \tag{3}$$

where η^k is the learning rate. Compared with the global full gradient, local gradient descent requires stricter step sizes to ensure algorithm stability; however, it achieves significant computational tractability. Similar techniques are used in many machine learning methods, such as actor-critic methods, where the actor and critic models are updated in a decoupled manner [31]. In addition, recall that the global and local loss functions defined in (2) and (1) are both non-negative. If the demonstrations and the generated trajectories can match perfectly, $\sum_{i-1}^m \mathcal{L}_i$ and $\mathcal{L}_i$ share the same minimizer at 0. The effectiveness of 'local gradients' will be further justified by our experiments.

The implementation of update (3) is summarized in Algorithm 1, and it relies on the following chain rule to compute the gradient.

$$\left.\frac{d\mathcal{L}_i(\boldsymbol{\xi}_i^\star(\Theta), \boldsymbol{\xi}_i^d)}{d\theta_i}\right|_{\theta_i^k} = \left.\frac{\partial\mathcal{L}_i(\boldsymbol{\xi}_i^\star(\Theta), \boldsymbol{\xi}_i^d)}{\partial\boldsymbol{\xi}_i^\star(\Theta)}\right|_{\boldsymbol{\xi}_i^\star(\Theta^k)} \cdot \left.\frac{\partial\boldsymbol{\xi}_i^\star(\Theta)}{\partial\theta_i}\right|_{\theta_i^k}. \tag{4}$$

For the first term of the chain rule, the derivative $\frac{\partial\mathcal{L}_i}{\partial\boldsymbol{\xi}_i^\star(\Theta)}$ is readily accessible because the function $\mathcal{L}_i(\boldsymbol{\xi}_i^\star(\Theta), \boldsymbol{\xi}_i^d)$ is explicitly defined. Its evaluation point $\boldsymbol{\xi}_i^\star(\Theta^k)$ relies on solving the forward problem of the game to obtain its (N.E.) with current parameter Θ^k. In this paper, we achieve this by employing an existing distributed Nash equilibrium-seeking algorithm proposed in [18]. More details of the implementation could be found in [1].

Algorithm 1: Inverse Learning for Distributed Differential Dynamic Game, the local update for robot i..

Input Demonstrations of trajectory $\boldsymbol{\xi}_i^d$.
Initialize a random guess for $\theta_i^{k=0}$.
for $k = 0, 1, 2, \cdots$ **do**
  Compute $\frac{\partial\mathcal{L}_i(\boldsymbol{\xi}_i^\star(\Theta), \boldsymbol{\xi}_i^d)}{\partial\boldsymbol{\xi}_i^\star(\Theta)}$ based on definition (2).
  Solving the forward problem of the dynamic game to obtain $\boldsymbol{\xi}_i^\star(\Theta^k)$. (cf. Algorithm **??**, Appendix.)
  Solving a diff-KKT condition to obtain $\frac{\partial\boldsymbol{\xi}_i^\star(\Theta)}{\partial\theta_i}$.
  Compute $\left.\frac{d\mathcal{L}_i(\boldsymbol{\xi}_i^\star(\Theta), \boldsymbol{\xi}_i^d)}{d\theta_i}\right|_{\theta_i^k}$ using (4).
  Update: $\theta_i^{k+1} = \theta_i^k - \eta^k \left.\frac{d\mathcal{L}_i(\boldsymbol{\xi}_i^\star(\Theta), \boldsymbol{\xi}_i^d)}{d\theta_i}\right|_{\theta_i^k}$.
end
Output θ_i

The major obstacle arises from the second term of the chain rule, where $\partial\boldsymbol{\xi}_i^\star(\Theta)/\partial\theta_i$ characterizes the change in the robot's (N.E.) trajectories corresponding to the change from its local parameter. Given a general optimal control system, its solution trajectory $\boldsymbol{\xi}_i^\star(\Theta)$ does not admit an analytical form. Thus, one possible way to compute $\partial\boldsymbol{\xi}_i^\star(\Theta)/\partial\theta_i$ is by numerical approximation [32]. However, the feasibility of this approach is extremely challenging, due to the large number of robots and the complexity of their trajectories considered in this paper. Motivated by these, we next present a new *distributed* method to compute $\partial\boldsymbol{\xi}_i^\star(\Theta)/\partial\theta_i$, whose idea is based on differentiating the KKT condition [8] of the (N.E.) with respect to the parameter Θ [11]. This yields a new representation of the derivative that can significantly reduce its computation burden, and the computation can be performed in a *distributed* fashion.

B. A Fully Distributed Solver for Diff-KKT

In this subsection, we introduce a distributed and efficient approach to compute the $\partial\boldsymbol{\xi}_i^\star(\Theta)/\partial\theta_i$ in (4). First, given x_i^0, define a compact form for robot i's dynamics constraints

$$\mathbf{F}_i(\mathbf{x}_i, \mathbf{u}_i, \theta_i) = \begin{bmatrix} x_i^1 - f_i(x_i^0, u_i^0, \theta_i) \\ x_i^2 - f_i(x_i^1, u_i^1, \theta_i) \\ \vdots \\ x_i^T - f_i(x_i^{T-1}, u_i^{T-1}, \theta_i) \end{bmatrix} = \mathbf{0}. \tag{5}$$

The (N.E.) of a game is the collection of the optimal trajectories of the robots' local optimal control problems. Thus, define augmented functions

$$\mathbf{H}_i = \mathcal{J}_i(\mathbf{x}_i, \mathbf{x}_{\mathcal{N}_i}, \mathbf{u}_i, \theta_i) + \boldsymbol{\lambda}_i^\top \mathbf{F}_i(\mathbf{x}_i, \mathbf{u}_i, \theta_i), \tag{6}$$

with $\boldsymbol{\lambda}_i = \{\lambda_1, \cdots, \lambda_m\}$ being the co-states of the dynamics constraints. Then for any Θ, the trajectory $\boldsymbol{\xi}_i^\star(\Theta) = \{\mathbf{x}_i^\star(\Theta), \mathbf{u}_i^\star(\Theta)\}$ must satisfy a distributed discrete-time

KKT [33] condition, which reads: $\forall i \in \{1, \cdots, m\}$,

$$\frac{\partial \mathbf{H}_i}{\partial \mathbf{x}_i} = \frac{\partial \mathcal{J}_i}{\partial \mathbf{x}_i} + \boldsymbol{\lambda}_i^\top \frac{\partial \mathbf{F}_i}{\partial \mathbf{x}_i} = \mathbf{0} \tag{7a}$$

$$\frac{\partial \mathbf{H}_i}{\partial \mathbf{u}_i} = \frac{\partial \mathcal{J}_i}{\partial \mathbf{u}_i} + \boldsymbol{\lambda}_i^\top \frac{\partial \mathbf{F}_i}{\partial \mathbf{u}_i} = \mathbf{0} \tag{7b}$$

$$\frac{\partial \mathbf{H}_i}{\partial \boldsymbol{\lambda}_i} = \mathbf{F}_i = \mathbf{0} \tag{7c}$$

Now, to obtain the ${}^{\partial \boldsymbol{\xi}_i^\star(\Theta)}/_{\partial \theta_i}$, our idea is to differentiate equation (7) with respect to Θ. This will provide us with a neat and easy-to-solve equation set that directly takes ${}^{\partial \boldsymbol{\xi}_i^\star(\Theta)}/_{\partial \theta_i}$ as variables. To visualize this, define

$$\boldsymbol{X}_i = \frac{\partial \mathbf{x}_i^\star(\Theta)}{\partial \Theta}, \ \boldsymbol{U}_i = \frac{\partial \mathbf{u}_i^\star(\Theta)}{\partial \Theta}, \ \boldsymbol{\Lambda}_i = \frac{\partial \boldsymbol{\lambda}_i^\star(\Theta)}{\partial \Theta}. \tag{8}$$

Since all variables in (8) are functions of Θ, differentiating (7) with respect to Θ yields the following **Diff-KKT**:

$$M_i^\alpha \boldsymbol{X}_i + N_i^\alpha \boldsymbol{U}_i + \sum_{j \in \mathcal{N}_i} Q_{ij}^\alpha \boldsymbol{X}_j + S_i^\alpha \boldsymbol{\Lambda}_i + C_i^\alpha = \mathbf{0} \tag{9a}$$

$$M_i^\beta \boldsymbol{X}_i + N_i^\beta \boldsymbol{U}_i + \sum_{j \in \mathcal{N}_i} Q_{ij}^\beta \boldsymbol{X}_j + S_i^\beta \boldsymbol{\Lambda}_i + C_i^\beta = \mathbf{0} \tag{9b}$$

$$M_i^\gamma \boldsymbol{X}_i + N_i^\gamma \boldsymbol{U}_i + C_i^\gamma = \mathbf{0} \tag{9c}$$

with the application of the chain rule on the derivatives of $\mathbf{x}_i^\star(\Theta)$ and $\mathbf{u}_i^\star(\Theta)$ and $\boldsymbol{\lambda}_i^\star(\Theta)$ with respect to Θ:

$$\begin{aligned} M_i^\alpha &= \frac{\partial^2 \mathbf{H}_i}{\partial \mathbf{x}_i^{\star 2}}, \ N_i^\alpha = \frac{\partial^2 \mathbf{H}_i}{\partial \mathbf{x}_i^\star \partial \mathbf{u}_i^\star}, \ Q_{ij}^\alpha = \frac{\partial^2 \mathbf{H}_i}{\partial \mathbf{x}_i^\star \partial \mathbf{x}_j^\star} \\ S_i^\alpha &= \frac{\partial^2 \mathbf{H}_i}{\partial \mathbf{x}_i^\star \partial \boldsymbol{\lambda}_i^\star}, \ C_i^\alpha = \frac{\partial^2 \mathbf{H}_i}{\partial \mathbf{x}_i^\star \partial \theta_i^k} \end{aligned} \tag{10a}$$

$$\begin{aligned} M_i^\beta &= \frac{\partial^2 \mathbf{H}_i}{\partial \mathbf{u}_i^\star \partial \mathbf{x}_i^\star}, \ N_i^\beta = \frac{\partial^2 \mathbf{H}_i}{\partial \mathbf{u}_i^{\star 2}}, \ Q_{ij}^\beta = \frac{\partial^2 \mathbf{H}_i}{\partial \mathbf{u}_i^\star \partial \mathbf{x}_j^\star} \\ S_i^\beta &= \frac{\partial^2 \mathbf{H}_i}{\partial \mathbf{u}_i^\star \partial \boldsymbol{\lambda}_i^\star}, \ C_i^\beta = \frac{\partial^2 \mathbf{H}_i}{\partial \mathbf{u}_i^\star \partial \theta_i^k} \end{aligned} \tag{10b}$$

$$\begin{aligned} M_i^\gamma &= \frac{\partial^2 \mathbf{H}_i}{\partial \boldsymbol{\lambda}_i^\star \partial \mathbf{x}_i^\star}, \ N_i^\gamma = \frac{\partial^2 \mathbf{H}_i}{\partial \boldsymbol{\lambda}_i^\star \partial \mathbf{u}_i^\star}, \ C_i^\gamma = \frac{\partial^2 \mathbf{H}_i}{\partial \boldsymbol{\lambda}_i^\star \partial \theta_i^k} \end{aligned} \tag{10c}$$

where we use $\frac{\partial^2 \mathbf{H}_i}{\partial \sigma_i^\star \partial \mu_i^\star}$ to denote the seconder-order derivative of $\mathbf{H}_i(\cdot)$ evaluated at $\{\sigma_i^\star(\Theta), \mu_i^\star(\Theta)\}$. All equations in (10) are simple numerical matrices and are readily computable from (7), because $\mathbf{H}_i(\cdot)$ is explicitly defined and $\{\mathbf{x}_i^\star(\Theta), \mathbf{u}_i^\star(\Theta)\}, \boldsymbol{\lambda}_i^\star(\Theta)$ are obtained from forward Nash seeking algorithm given the current Θ. To remark the effectiveness of reformulation, given Assumption 1, results in [33, Sec. 5.9.2] implies the existence and uniqueness of solution to (7); results in [5, Theorem 1] implies the uniqueness of $\mathbf{X}_i$ and $\mathbf{U}_i$ in (9).

Distributed Diff-KKT Solver: Solving (8) from (9) gives us the gradient $\frac{\partial \boldsymbol{\xi}_i^\star(\Theta)}{\partial \theta_i}$ for each robot. However, solving the equation in a centralized manner is not scalable as the robot number grows. To address this, we notice that the coupled terms, i.e., $Q_{ij}, \boldsymbol{X}_j$, in (9) only exist among connected neighbors $j \in \mathcal{N}_i$. This motivates us to develop a fully distributed solver to compute the gradient. To that end, we rewrite all variables and matrices into a compact linear equation form.

Algorithm 2: Distributed Solver for Diff-KKT, the local update for robot i.

Input $\boldsymbol{\xi}_i^\star(\Theta^k)$, θ_i^k.
Compute $\boldsymbol{\lambda}_i^\star(\Theta^k)$ using equations (7) with $\boldsymbol{\xi}_i^\star(\Theta) = \{\mathbf{x}_i^\star(\Theta^k), \mathbf{u}_i^\star(\Theta^k)\}$.
Compute matrices $\boldsymbol{A}_{i,i}$ and $\boldsymbol{A}_{i,j}$, $j \in \mathcal{N}_i$ by (10) and (12).
Acquire matrices $\boldsymbol{A}_{\ell,i}$, $\ell \in \mathcal{N}_i$ from each neighbor ℓ of robot i. Assign $\boldsymbol{A}_{\ell,i} = \mathbf{0}$ for $\ell \notin \mathcal{N}_i$.
Compose matrices $\boldsymbol{\Psi}_i$, $\widehat{\boldsymbol{C}}_i$ by their definitions.
Initialize $\tau = 0$, $\delta \in \mathbb{R}_+$, and $\boldsymbol{Y}_i^{\tau=0}$, $\boldsymbol{Z}_i^{\tau=0}$ as random matrices with proper sizes.
while $\max_i(|\boldsymbol{Y}_i^{\tau+1} - \boldsymbol{Y}_i^{\tau}|) \geq \epsilon_{\boldsymbol{Y}}$ **do**
  Exchange states $\boldsymbol{Z}_i^\tau$ among neighboring robots.
  State update:

$$\begin{aligned} \boldsymbol{v}_i^\tau &= \boldsymbol{\Psi}_i \boldsymbol{Y}_i^\tau - \widehat{\boldsymbol{C}}_i - \sum_{\ell \in \mathcal{N}_i} (\boldsymbol{Z}_i^\tau - \boldsymbol{Z}_\ell^\tau). \\ \boldsymbol{Y}_i^{\tau+1} &= \boldsymbol{Y}_i^\tau - \delta \boldsymbol{\Psi}_i^\top \boldsymbol{v}_i^\tau \\ \boldsymbol{Z}_i^{\tau+1} &= \boldsymbol{Z}_i^\tau + \delta \boldsymbol{v}_i^\tau \end{aligned}$$

end
Obtain $\boldsymbol{X}_i$, $\boldsymbol{U}_i$ from $\boldsymbol{Y}_i^\tau$ based on (12).
Output $\left.\frac{\partial \boldsymbol{\xi}_i^\star(\Theta)}{\partial \theta_i}\right|_{\theta_i^k}$ from $\{\boldsymbol{X}_i, \boldsymbol{U}_i\}$ based on (8).

$$\boldsymbol{A}_{i,i} \boldsymbol{Y}_i + \sum_{j \in \mathcal{N}_i} (\boldsymbol{A}_{i,j} \boldsymbol{Y}_j) + \overline{C}_i = \mathbf{0}. \tag{11}$$

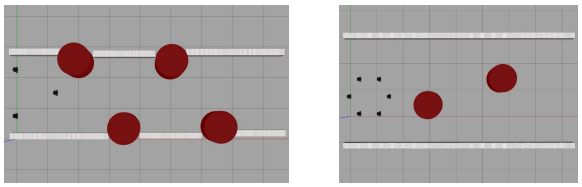
where for all i and $j \in \mathcal{N}_i$,

$$\begin{aligned} \boldsymbol{A}_{i,i} &= \begin{bmatrix} M_i^\alpha & N_i^\alpha & S_i^\alpha \\ M_i^\beta & N_i^\beta & S_i^\beta \\ M_i^\gamma & N_i^\gamma & S_i^\gamma \end{bmatrix}, \quad \boldsymbol{Y}_i = \begin{bmatrix} \boldsymbol{X}_i \\ \boldsymbol{U}_i \\ \boldsymbol{\Lambda}_i \end{bmatrix} \\ \boldsymbol{A}_{i,j} &= \begin{bmatrix} Q_{i,j}^\alpha & \mathbf{0} & \mathbf{0} \\ Q_{i,j}^\beta & \mathbf{0} & \mathbf{0} \\ Q_{i,j}^\gamma & \mathbf{0} & \mathbf{0} \end{bmatrix}, \quad \overline{C}_i = \begin{bmatrix} C_i^\alpha \\ C_i^\beta \\ C_i^\gamma \end{bmatrix} \end{aligned} \tag{12}$$

where $\boldsymbol{Y}_i$ is the local unknown of robot i, $\boldsymbol{A}_{i,i}$ and $\boldsymbol{A}_{i,j}$ are known matrices, and $\boldsymbol{Y}_j$, $j \in \mathcal{N}_i$ is the coupled unknown from i's neighbors. Since each robot in the network possesses an equation in the form of (11), to compute a set of $\boldsymbol{Y}_i$, $i \in \{1, \cdots, m\}$ satisfying all these equation, we essentially need to solve the following compact equation set

$$\sum_{i=1}^{m} (\boldsymbol{\Psi}_i \boldsymbol{Y}_i + \widehat{\boldsymbol{C}}_i) = 0 \tag{13}$$

where $\boldsymbol{\Psi}_i = \left[\boldsymbol{A}_{1,i}^\top, ..., \boldsymbol{A}_{m,i}^\top\right]^\top$, $\widehat{\boldsymbol{C}}_i = \left[\mathbf{0}, ..., \overline{C}_i^\top, ..., \mathbf{0}\right]^\top$. The matrix is a zero matrix if undefined. In $\widehat{\boldsymbol{C}}_i$, the matrix $\overline{C}_i^\top$ is located at the ith block. By stacking the matrices $\boldsymbol{A}_{\ell,i}$ and $\overline{C}_i^\top$, each row block of (13) is associated with one (11) for $i \in \{1, \cdots, m\}$. Further note that network $\mathbb{G}$ is undirected, i.e., $i \in \mathcal{N}_\ell$ yields $\ell \in \mathcal{N}_i$, thus, robot i has access to $\boldsymbol{\Psi}_i$ based on its local communication with its neighbors.



Scenario c) Scenario d)

Fig. 2: Gazebo environment for scenarios c) and d).

Now, suppose each robot i knows $\boldsymbol{\Psi}_i$ and $\widehat{\boldsymbol{C}}_i$, we introduce Algorithm 2 for the robots to efficiently solve its $\boldsymbol{Y}_i$.

Algorithm 2 is fully distributed, in the sense that the computation of each robot only relies on its own state and the states of its neighbors. The convergence of the algorithm is characterized by the following result:

Lemma 3.1: (Validity of Algorithm 2): Suppose the network $\mathbb{G}$ is undetected and connected, suppose equation set (13) has a unique solution, by Algorithm 2, if the positive step-size δ is sufficiently small, the state $\boldsymbol{Y}_i^\tau$ of robot i will converge asymptotically to a state $\boldsymbol{Y}_i^\star$, where the set of $\{\boldsymbol{Y}_i^\star, i = 1 \cdots, m\}$ forms a solution to (13).

The correctness of Lemma 3.1 follows our previous works for solving coupled linear constraints using distributed network flows [34] with a complete proof in [1].

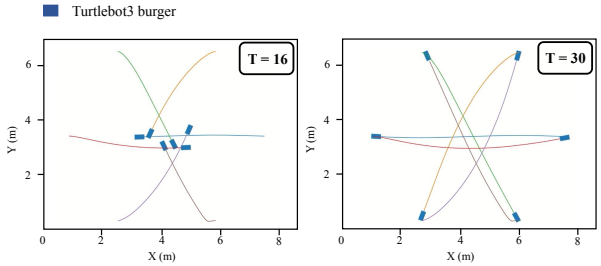
IV. Experiments

This section presents simulation experiments to validate the effectiveness, scalability, and generalizability of the proposed D3G approach for multi-robot coordination. The system includes two types of robots: TurtleBot3 Burger and Waffle. We consider heterogeneous settings, where each robot has different dynamics, such as different radii, weights, and velocity/angular ranges. Four scenarios are used: (a) fixed swapping in open ground, (b) formation initialization using the environment in the introductory Fig. 1, (c) cooperative payload transportation, and (d) formation maintenance using the environments in Fig. 2. Simulations are done in Gazebo via ROS. Robots can communicate with each other, but all computations are performed locally.

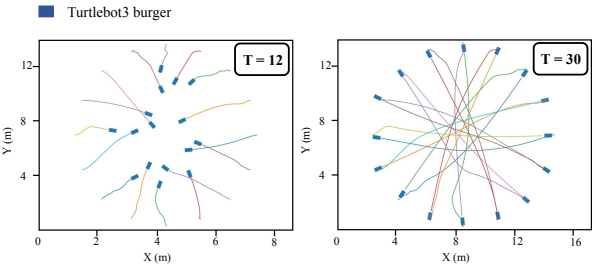
Parameterization of objective functions: Fcuntion $\mathcal{J}_i$ is parameterized by considering a linear combination of the following cost terms with unknown weights: *formation maintenance*, which defines the positional relationship of neighboring robots in terms of their relative positions, distances, or velocities; *risk/obstacle avoidance*, which employs a reciprocal function to repel robots from given risk areas; *collision avoidance*, which utilizes a reciprocal function to prevent robots from colliding with each other; and *waypoint following*, which provides sparse navigation cues for navigating complex environments. We note that these functions satisfy Assumption 1.

Experiment Settings in Each Scenario: We invite humans to create several sets of trajectories (incorporating human-induced random noise to optimal coordination trajectories computed from N.E. of a game with parameter $\Theta^\star$) to serve as the expert demonstration data. Using Algorithm 1, we learn $\theta_i^\star$ for each robot from those demonstrations. Additionally, for each scenario, we test the generalizability of the learned objective functions by applying them in a new environment where the robots can still generate appropriate coordinated behaviors. Details of simulation setups and results are as follows:

Scenario a): We solve a multi-robot fixed swapping task. As shown in Fig. 3-a, in the demonstrations, six robots are initialized around a circle-like formation. Each robot navigates to the diagonally opposite goal position on the other side of the circle. Throughout the process, they must dynamically adjust their positions to move without colliding. We test the generalization of the learned objective function with an increased number of robots, and the task is accomplished very well. Fig.3-b shows an example with **sixteen** robots.



(a) Inverse game solving: learning the fixed swapping task from demonstrations. (Right: the reproduced motion)



(b) Generalization of the learned objective with 16 robots.

Fig. 3: Learning fixed swapping tasks with sixteen robots.

Scalability of Distributed Solver: Using different numbers of robots in scenario (a), we compare the computational scalability of the proposed algorithm with the GT-IRL [26] and IKKT [7] methods. The comparison result is presented in Fig.4. Here, D3G is evaluated based on the per-iteration time of Algorithm 1, which requires the convergence of Algorithm 2 for the inverse pass and Algorithm 3 for the forward pass. Since both algorithms are gradient-based and are sensitive to initial values, we use the result of the last iteration in Algorithm 1 as the initial values for the new iteration. The stopping criteria are chosen such that the variables do not change 1% of their initial values (around hundreds of iterations). For GT-IRL, its forward pass employs a similar but centralized gradient-based method to solve a dynamic game, and the inverse pass uses a centralized linear equation solver. The IKKT method uses a constraint optimization formulation, which is solved iteratively without a forward/inverse structure. From Fig. 4 and the trend of the data, we observe that as the number of robots increases, D3G outperforms both GT-IRL and IKKT in terms of computation time. The inverse pass of D3G outperforms GT-IRL. For D3G, the local computation of each robot is not significantly affected by the system size

as the others, thanks to the distributed nature of the algorithm. The increase in time is mainly because Algorithms 2 and 3 require more iterations to converge. In contrast, for centralized algorithms, the computation time grows quickly due to the increase in the number of variables and constraints.

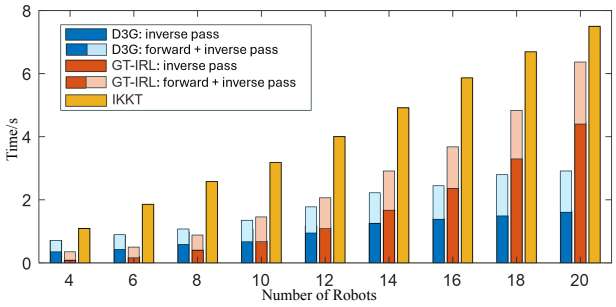
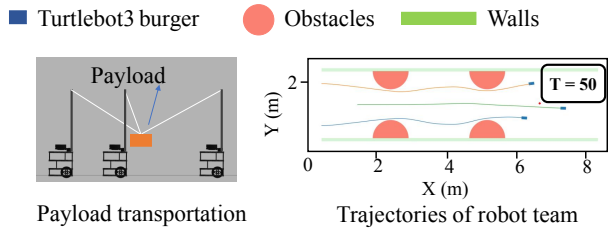


Fig. 4: Comparison of computation time with GT-IRL and IKKT.

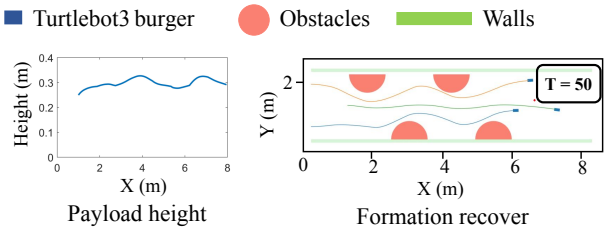
Scenario b): As shown in the introductory example in Fig. 1, three robots start from initial positions at 0 speed to initialize a linear formation at the goal position, maintaining distances of 0.8m and velocities of 0.2m/s. There exists a wide obstacle that robots have to avoid. From the demonstrations, the robots learn to adjust their formation to a 'compact' shape when moving through the narrow space, then recover and form the desired formation at target positions. To test the generalization of the learned objective functions, we solve the learned game but change the obstacle's opening position from the middle to the side. The robots can still generate proper coordination to initialize the formation.

Scenario c): As shown in Fig. 5, three turtlebots start from different initial positions and cooperatively transport a slung payload. We assume each robot is attached to the payload with a length tether visualized in Fig. 5a. The payload has to maintain clearance from the ground. In addition, to stabilize the payload and prevent excessively large forces between the robots and the payload, the robot team will learn to maintain an equilateral triangle-like form, and keep the payload in its centroid. For simplicity, we ignore the dynamics of the payload but only consider the equilibrium point as its location. By learning robots' local objective functions, the reconstructed trajectories are shown in the right plot of Fig. 5a. We then test the generalization of the model in a new environment. In Fig. 5b, the placement of obstacles requires more sophisticated robot maneuvers. The height of the payload is still well maintained, and the robot team keeps the payload in its centroid as much as possible for stable moving.

Scenario d): As shown in Fig. 6, six heterogeneous turtlebot3 robots, including three burgers and three waffles, maintain a desired (circle-like) formation while navigating through complex environments with obstacles. Robots learn to balance between local objective functions including collision avoidance and formation maintenance. The reconstructed trajectories in Fig. 6a show the robot's capability to leverage the shape of the obstacle to minimize the formation degradation. We test the generalization of the learned game in 6b in a new environment with **eight** robots. The robots generate smooth trajectories and formation transitions. Furthermore,



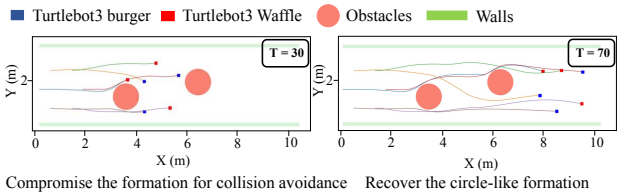
(a) Inverse game solving: learning payload transportation from demonstrations. (Right: the reproduced motion at convergence)



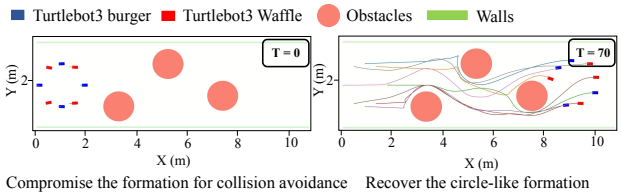
(b) Generalization of the learned objective in a new environment.

Fig. 5: Learning payload transportation with three robots.

we observe two robots change their orders (T=0: different types of robots are separated v.s. T=70: two blue/red robots become adjacent) to reduce the formation degradation.



(a) Inverse game solving: learning formation maintenance with six heterogeneous robots from demonstrations, figures show reproduced motion at convergence.

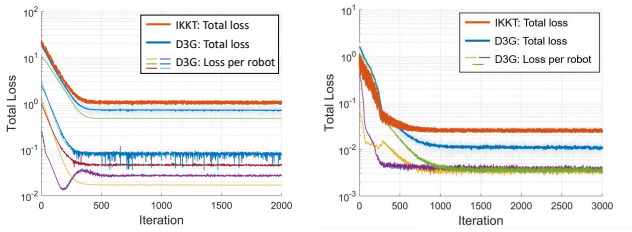


(b) Generalization of the learned objective with eight heterogeneous robots in a new environment.

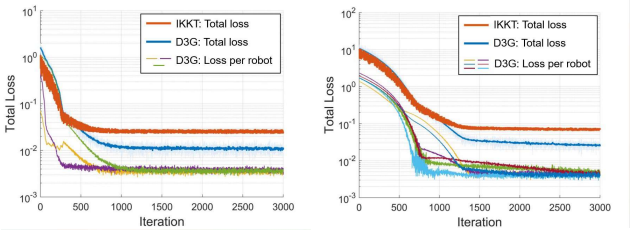
Fig. 6: Learning formation control with heterogeneous robots.

Comparison of Learning loss:

We compare the convergence of the proposed method with the centralized IKKT method [7]. The GT-IRL [26] is not included since it is also based on the diff-KKT condition, leading to a similar convergence property as D3G in terms of learning loss. The results of all scenarios are shown in Fig. 7, where the y-axis represents the learning loss $\mathcal{L}_i$ for each robot, or the total learning loss for the whole system. In all scenarios, the total learning loss converges, and the parameter values will converge to those of the demonstrations. Apart from the advantage in computation scalability demonstrated previously in Fig. 4, the proposed D3G, which is fully distributed, demonstrates a comparable, and in some cases, better convergence speed than the centralized IKKT.



(a) Learning loss of scenario **a**. (b) Learning loss of scenario **b**.



(c) Learning loss of scenario **c**. (d) Learning loss of scenario **d**.

Fig. 7: Total loss $\sum_{i=1}^{m} \mathcal{L}_i$ (with trajectory mismatch defined in (2)) of each scenario.

V. Conclusion and Future Work

We have developed a new approach for inverse learning of a *Distributed Differentiable Dynamic Game* (D3G), which aims to efficiently learn multi-robot coordination from demonstrations using robots' local information exchange. We represented multi-robot coordination as the Nash equilibrium of a parameterized dynamic game. The goal was to learn the parameters of the game so that it can reconstruct desired multi-robot coordination. To this end, we developed a distributed inverse dynamic game algorithm with a solver for the diff-KKT condition that allows robots to cooperatively learn parameters for their dynamics and objective functions. We have shown the effectiveness of the proposed algorithm through analysis and high-fidelity Gazebo simulations and compared it with existing methods. For future works, we plan to further develop the inverse problem of D3G into a reinforcement learning paradigm. Instead of based on demonstrations, robots will learn coordination strategies through self-explorations.

References

[1] Y. Zhou, W. Jin, and X. Wang, "D3g: Learning multi-robot coordination from demonstrations," *arXiv preprint arXiv:2207.08892*, 2022.

[2] K. Zhang, Z. Yang, and T. Başar, "Multi-agent reinforcement learning: A selective overview of theories and algorithms," *Handbook of Reinforcement Learning and Control*, pp. 321–384, 2021.

[3] X. Wang, S. Mou, and B. D. Anderson, "Consensus-based distributed optimization enhanced by integral feedback," *IEEE Transactions on Automatic Control*, vol. 68, no. 3, pp. 1894–1901, 2023.

[4] X. Lin, S. C. Adams, and P. A. Beling, "Multi-agent inverse reinforcement learning for certain general-sum stochastic games," *Journal of Artificial Intelligence Research*, vol. 66, pp. 473–502, 2019.

[5] W. Jin, S. Mou, and G. J. Pappas, "Safe pontryagin differentiable programming," *Advances in Neural Information Processing Systems*, vol. 34, pp. 16 034–16 050, 2021.

[6] N. Ab Azar, A. Shahmansoorian, and M. Davoudi, "From inverse optimal control to inverse reinforcement learning: A historical review," *Annual Reviews in Control*, vol. 50, pp. 119–138, 2020.

[7] P. Englert, N. A. Vien, and M. Toussaint, "Inverse kkt: Learning cost functions of manipulation tasks from demonstrations," *The International Journal of Robotics Research*, vol. 36, no. 13-14, pp. 1474–1488, 2017.

[8] D. Bertsekas, *Dynamic programming and optimal control: Volume I*. Athena scientific, 2012, vol. 1.

[9] M. Ratiu and M. A. Prichici, "Industrial robot trajectory optimization-a review," in *MATEC web of conferences*, vol. 126. EDP Sciences, 2017, p. 02005.

[10] J. Eschmann, "Reward function design in reinforcement learning," *Reinforcement Learning Algorithms: Analysis and Applications*, pp. 25–33, 2021.

[11] W. Jin, Z. Wang, Z. Yang, and S. Mou, "Pontryagin differentiable programming: An end-to-end learning and control framework," *Advances in Neural Information Processing Systems*, vol. 33, pp. 7979–7992, 2020.

[12] N. D. Ratliff, J. A. Bagnell, and M. A. Zinkevich, "Maximum margin planning," in *Proceedings of the 23rd international conference on Machine learning*, 2006, pp. 729–736.

[13] B. D. Ziebart, A. L. Maas, J. A. Bagnell, and A. K. Dey, "Maximum entropy inverse reinforcement learning." in *Aaai*, vol. 8. Chicago, IL, USA, 2008, pp. 1433–1438.

[14] X. Wang, J. Zhou, S. Mou, and M. J. Corless, "A distributed algorithm for least squares solutions," *IEEE Transactions on Automatic Control*, vol. 64, no. 10, pp. 4217–4222, 2019.

[15] M. Ye and G. Hu, "Distributed nash equilibrium seeking by a consensus based approach," *IEEE Transactions on Automatic Control*, vol. 62, no. 9, pp. 4811–4818, 2017.

[16] X. Wang, J. Hudack, and S. Mou, "Distributed algorithm with resilience for multi-agent task allocation," in *2021 4th IEEE International Conference on Industrial Cyber-Physical Systems (ICPS)*. IEEE, 2021, pp. 112–117.

[17] D. Fudenberg and J. Tirole, *Game theory*. MIT press, 1991.

[18] T. Tatarenko and A. Nedić, "Geometric convergence of distributed gradient play in games with unconstrained action sets," *IFAC-PapersOnLine*, vol. 53, no. 2, pp. 3367–3372, 2020.

[19] A. Bressan and W. Shen, "Small bv solutions of hyperbolic noncooperative differential games," *SIAM journal on control and optimization*, vol. 43, no. 1, pp. 194–215, 2004.

[20] F. Salehisadaghiani and L. Pavel, "Distributed nash equilibrium seeking: A gossip-based algorithm," *Automatica*, vol. 72, pp. 209–216, 2016.

[21] A. Bressan, "Noncooperative differential games. a tutorial," *Department of Mathematics, Penn State University*, p. 81, 2010.

[22] S. M. LaValle, *Planning algorithms*. Cambridge university press, 2006.

[23] S. Le Cleac'h, M. Schwager, and Z. Manchester, "Lucidgames: Online unscented inverse dynamic games for adaptive trajectory prediction and planning," *IEEE Robotics and Automation Letters*, vol. 6, no. 3, pp. 5485–5492, 2021.

[24] L. Peters, "Accommodating intention uncertainty in general-sum games for human-robot interaction," *Master's thesis, Hamburg University of Technology*, 2020.

[25] L. Peters, D. Fridovich-Keil, V. Rubies-Royo, C. Tomlin, and C. Stachniss, "Inferring objectives in continuous dynamic games from noise-corrupted partial state observations," 07 2021.

[26] K. Cao and L. Xie, "Game-theoretic inverse reinforcement learning: A differential pontryagin's maximum principle approach," *IEEE Transactions on Neural Networks and Learning Systems*, 2022.

[27] D. Fridovich-Keil, E. Ratner, L. Peters, A. D. Dragan, and C. J. Tomlin, "Efficient iterative linear-quadratic approximations for nonlinear multi-player general-sum differential games," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, 2020, pp. 1475–1481.

[28] X. Liu, L. Peters, and J. Alonso-Mora, "Learning to play trajectory games against opponents with unknown objectives," *IEEE Robotics and Automation Letters*, vol. 8, no. 7, pp. 4139–4146, 2023.

[29] Y. Zou, B. Huang, Z. Meng, and W. Ren, "Continuous-time distributed nash equilibrium seeking algorithms for non-cooperative constrained games," *Automatica*, vol. 127, p. 109535, 2021.

[30] T. Başar and G. J. Olsder, *Dynamic noncooperative game theory*. SIAM, 1998.

[31] L. Zheng, T. Fiez, Z. Alumbaugh, B. Chasnov, and L. J. Ratliff, "Stackelberg actor-critic: Game-theoretic reinforcement learning algorithms," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 36, no. 8, 2022, pp. 9217–9224.

[32] J. P. Perdew, K. Burke, and M. Ernzerhof, "Generalized gradient approximation made simple," *Physical review letters*, vol. 77, no. 18, p. 3865, 1996.

[33] S. P. Boyd and L. Vandenberghe, *Convex optimization*. Cambridge university press, 2004.

[34] X. Wang, S. Mou, and B. D. Anderson, "Scalable, distributed algorithms for solving linear equations via double-layered networks," *IEEE Transactions on Automatic Control*, vol. 65, no. 3, pp. 1132–1143, 2019.