Potential iLQR: A Potential-Minimizing Controller for Planning Multi-Agent Interactive Trajectories

Talha Kavuncu
Aerospace Engineering
UIUC
kavuncu2@illinois.edu

Ayberk Yaraneri
Aerospace Engineering
UIUC
ayberky2@illinois.edu

Negar Mehr
Aerospace Engineering
UIUC
negar@illinois.edu

Abstract—Many robotic applications involve interactions between multiple agents where an agent's decisions affect the behavior of other agents. Such behaviors can be captured by the equilibria of differential games which provide an expressive framework for modeling the agents' mutual influence. However, finding the equilibria of differential games is in general challenging as it involves solving a set of coupled optimal control problems. In this work, we propose to leverage the special structure of multi-agent interactions to generate interactive trajectories by simply solving a single optimal control problem, namely, the optimal control problem associated with minimizing the potential function of the differential game. Our key insight is that for a certain class of multi-agent interactions, the underlying differential game is indeed a potential differential game for which equilibria can be found by solving a single optimal control problem. We introduce such an optimal control problem and build on single-agent trajectory optimization methods to develop a computationally tractable and scalable algorithm for planning multi-agent interactive trajectories. We will demonstrate the performance of our algorithm in simulation and show that our algorithm outperforms the state-of-the-art game solvers. To further show the real-time capabilities of our algorithm, we will demonstrate the application of our proposed algorithm in a set of experiments involving interactive trajectories for two quadcopters.

I. INTRODUCTION

Many robotic applications involve interactions between multiple agents. For instance, two quadcopters may need to interact and implicitly coordinate to successfully navigate in a shared three-dimensional space (Fig. 1). An autonomous car may need to interact with human-driven cars or pedestrians to navigate an intersection. Planning in such interactive settings is in general challenging due to the feedback interactions between the agents. An agent's state and action will affect the state and actions of the other agents, i.e., agents are coupled by their intentions and actions. In this work, we demonstrate that by leveraging the structure that is inherent in such interactive settings, we can resolve these couplings, and agents can plan interactive trajectories by solving an optimal control problem.

It has been shown that feedback interactions in interactive settings can be captured by differential games [1, 2]. Every agent is a utility maximizer seeking to maximize their own utility over a horizon of time while an agent's utility can depend on the state and actions of all the agents. In such settings, the mutual influence of the agents as well as the outcome of the interaction is best represented by an equilib-
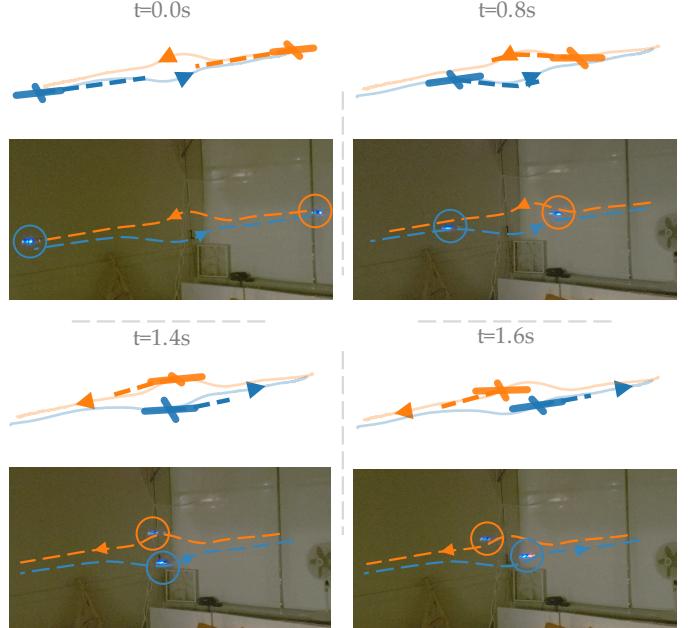

Fig. 1. Demonstration of our interactive trajectory planning algorithm in an experiment involving two quadcopters. The two quadcopters start at roughly the same altitude and switch their positions such that the starting position of one becomes the goal position of the other. The quadcopters exhibit intuitive interactive trajectories and change altitudes for avoiding collisions with each other.

rium of the underlying differential game. However, finding the equilibrium strategies of such games is in general challenging as it involves solving a set of coupled optimal control problems. Consequently, most differential games do not have analytical solutions, and even their numerical solutions are not scalable [3, 4].

In this work, we propose to leverage the special structure of multi-agent interactions to generate interactive trajectories by simply solving a *single* optimal control problem. Our key insight is that for a certain class of multi-agent interactions, the underlying differential game is indeed a potential differential game. Potential games are a class of games for which a Nash equilibrium always exists, and the Nash equilibrium can be found by solving a single optimization problem [5]. Thus,

we can employ a standard single-agent trajectory optimization method such as iLQR [6] for planning multi-agent interactive trajectories.

We will prove that a class of multi-agent interactions, namely, interactions where the mutual couplings between the agents are symmetric, are indeed potential differential games. For such games, we introduce the optimal control problem associated with minimizing the potential of the game whose solution is the equilibrium trajectory of the original interaction game. Using this result, we develop a computationally tractable algorithm for interactive trajectory planning. We will compare the performance of our algorithm with the state-of-the-art in interactive trajectory planning both in terms of the quality of the trajectories as well as the computational tractability. To further show the real-time capabilities of our algorithm, we will demonstrate the application of our framework in a set of experiments involving interactive trajectories for two quadcopters.

II. RELATED WORK

A. Interactive Trajectory Planning

The common approach to interactive trajectory planning is for a robot to make predictions of the future trajectories of other agents and plan reactively [7, 8, 9, 10, 11]. Planning reactively will make the agents decoupled and simplify the control problem. Nevertheless, throughout this decoupling, agents lose the capability to affect each other. To capture this, it has been shown that interactive settings can be modeled by equilibria of differential games [1, 2, 12]. Several methods have been proposed for finding equilibria in interaction games. In [1], a Stackelberg equilibrium was considered where one agent is the leader, and the other agent is the follower. To find Nash equilibria of the interaction game, in [13], a hierarchical decomposition of the underlying game into strategic and tactical games was proposed. Iterative best response algorithms were developed for capturing interactions in racing problems and autonomous driving settings [14, 15, 16]. In [17], iterative dynamic programming in Gaussian belief space was used to solve for equilibria of a game-theoretic continuous POMDP.

B. Approximate Solutions to Differential Games

To find equilibria of general differential games, sequential linear-quadratic methods were proposed for two-player zero-sum differential games [18, 19]. To enable scalable interactive trajectory planning for a broad class of differential games, recently, a local iterative algorithm was proposed in [20] where the analytic solution to the Linear Quadratic games [4] was exploited for approximating the equilibria of general-sum differential games. This algorithm builds upon the iterative linear-quadratic regulator (iLQR) [6], and at every iteration, solves for the LQ game that results from linearizing the system dynamics and finding a quadratic approximation of the agents' cost functions. A similar iterative method was proposed in [21] for planning interactive trajectories in the presence of uncertainties where equilibria of risk-sensitive dynamic games were sought. In [22], a solver was developed for interactive trajectory planning in the presence of general nonlinear state and input constraints.

C. Potential Games

The literature on potential games is mostly focused on static games. A class of static potential games with pure Nash equilibria was identified in [23]. Later, potential games were also introduced in [5]. Because of the appealing properties of potential games, potential games have had applications in various control and resource allocation problems [24, 25, 26, 27, 28, 29]. We argue that potential games, in the form of potential differential games, can be further utilized for trajectory planning in multi-agent settings.

III. PROBLEM FORMULATION

We assume that we have N agents. For each agent i, $1 \leq i \leq N$, the vector $u_i(t) \in \mathbb{R}^{m_i}$ represents the control input of agent i at time t, where m_i is the dimension of the control input of agent i. Similarly, we let $x_i(t) \in \mathbb{R}^{n_i}$ denote the state of agent i at time t, where n_i is the dimension of the state space of agent i. We let $x(t) = (x_1(t), \cdots, x_N(t))$ denote the concatenated vector of all agents' states at time t, and n be the dimension of the state vector $x(t)$. We further use $u(t) = (u_1(t), \cdots, u_N(t))$ to denote the vector of all agents' control inputs at time t. The overall system dynamics are

$$\dot{x}(t) = f(x(t), u(t), t). \tag{1}$$

We consider open-loop control inputs, i.e., control inputs that are only a function of the system's initial state x_0 and time t:

$$u_i(t) = u_i(x_0, t). \tag{2}$$

Although the open-loop assumption may seem restrictive, in many practical applications, open-loop trajectory planning algorithms are applied in a receding-horizon fashion to approximate closed-loop feedback policies[1]. For each agent i, we let the set of Borel measurable functions $U_i := \{u_i | u_i : T \to \mathbb{R}^{m_i}\}$ represent the open-loop strategy space of player i that maps time to the player i's control input. Moreover, we let $U_{-i} := U_1 \times \cdots \times U_{i-1} \times U_{i+1} \times \cdots \times U_N$ represent the open-loop strategy space of all agents except agent i. We write (u_i, u^*_{-i}) to denote the vector

$$(u_1^*, \cdots, u_{i-1}^*, u_i, u_{i+1}^*, \cdots, u_N^*) \in U, \tag{3}$$

where U is the strategy space of all agents.

We assume that each agent i minimizes a cost function $J_i(\cdot)$, where J_i depends on the initial state x_0 and the agents' control signals $u_1, \cdots, u_N$ through the following

$$J_i(x_0, u_1, \cdots, u_N) = \int_0^T L_i(x(t), u(t), t)dt + S_i(x(T)), \tag{4}$$

[1]We acknowledge that in general, receding horizon application of open-loop equilibrium strategies may not be enough for finding close-loop equilibrium policies due to the difference between the information structure of open-loop and closed-loop policies. However, for many interactive trajectory planning settings, this is a valid approximation.

where T is a finite time horizon, and L_i and S_i are the running and terminal costs of agent i respectively.

In a compact form, we describe our differential game by

$$\Gamma_{x_0}^T := (N, \{U_i\}_{i=1}^N, \{J_i\}_{i=1}^N, f), \tag{5}$$

where x_0 is the initial state of the system. In a multi-agent setting, since each agent i seeks to optimize their own cost $J_i(.)$, the outcome of interaction is best represented via a notion of equilibrium. Among the various notions of equilibria, we look for Nash equilibria which characterize the interaction outcome in non-cooperative multi-agent settings.

Definition 1. *Given a differential game $\Gamma_{x_0}^T$, a control signal $u^* = (u_1^*, \cdots, u_N^*)$ is an open-loop Nash equilibrium if for every agent i, we have*

$$J_i(x_0, u^*(\cdot)) \leq J_i(x_0, u_i(\cdot), u_{-i}^*(\cdot)). \tag{6}$$

Intuitively, at Nash equilibrium, no agent has any incentive for unilateral deviation from $u_i^*(\cdot)$, i.e., when the control inputs of all the other agents $u_{-i}^*(\cdot)$ are fixed, as (6) suggests, agent i will not benefit from changing its equilibrium control signal. In general, finding Nash strategies u^* satisfying (6) is challenging since it involves solving N coupled optimal control problems.

IV. POTENTIAL DIFFERENTIAL GAMES

In this section, we introduce potential differential games, and then in the next section, we discuss how we can leverage potential differential games for tractable interactive trajectory planning in multi-agent settings. While finding Nash equilibria is in general challenging, there exists a class of differential games called potential differential games to which we can associate an optimal control problem (OCP) whose solutions are open-loop Nash equilibria for the original game $\Gamma_{x_0}^T$ [29].

Definition 2. *(cf. [29]) A differential game $\Gamma_{x_0}^T$ is a potential differential game if there exists an optimal control problem whose solutions are Nash equilibria of the game $\Gamma_{x_0}^T$.*

This problem reduction allows us to benefit from the existing planning methods for solving single-player optimal control problems to calculate Nash equilibria in interactive settings. In interactive trajectory planning, agents' dynamics are normally decoupled, and each agent's state update is governed by its own control inputs and its own dynamics. The coupling between the agents occurs due to the coupling between the agents' cost functions. For example, in navigation problems, the coupling between the agents arises from the inter-agent collision avoidance costs. We leverage this property, and in the rest of this paper, assume that for each agent i, we have

$$\dot{x}_i(t) = f_i(x_i(t), u_i(t), t), \tag{7}$$

where f_i is the dynamics of the i_{th} agent.

Under decoupled dynamics (7), it has been shown in [29] that a differential game is a potential differential game if the following holds.

Theorem 1. *For a differential game $\Gamma_{x_0}^T = (N, \{U_i\}_{i=1}^N, \{J_i\}_{i=1}^N, \{f_i\}_{i=1}^N)$, if for each agent i, the running and terminal costs have the following structure*

$$L_i(x(t), u(t), t) = p(x(t), u(t), t) + c_i(x_{-i}(t), u_{-i}(t), t), \tag{8}$$

and

$$S_i(x(T)) = \bar{s}(x(T)) + s_i(x_{-i}(T)), \tag{9}$$

then, the open-loop control input $u^* = (u_1^*, \cdots, u_N^*)$ that minimizes the following optimal control problem

$$\min_{u(\cdot)} \quad \int_0^T p(x(t), u(t), t)dt + \bar{s}(x(T)) \\ \text{s.t.} \quad \dot{x}_i(t) = f_i(x_i(t), u_i(t), t), \tag{10}$$

is an open-loop Nash equilibrium of the differential game $\Gamma_{x_0}^T$, i.e., $\Gamma_{x_0}^T$ is a potential differential game.

Proof: See Appendix A. ∎

Note that Theorem 1 requires the running cost of every agent i to be composed of a potential function p and a term c_i which has no dependence on the state and action of agent i. The potential function p may depend on the entire state and input vector x and u, but it is not agent-specific as it does not have any dependence on the agent's index i. On the other hand, the agent-specific term c_i must depend only on the states and actions of all agents except agent i. In the next section, we will show how this decomposition can be achieved when the coupling between agents occurs due to collision avoidance cost terms. It is important to mention that in general, there are less restrictive conditions under which a differential game is a potential game, but we have only included the conditions that are relevant to interactive trajectory planning. Interested reader is referred to [29] for further details.

Note that while a solution to (10) is always a Nash equilibrium of the original game $\Gamma_{x_0}^T$, there may exist other equilibria for the game $\Gamma_{x_0}^T$ which do not necessarily optimize (10). In other words, solving optimal control (10) always provides a set of equilibria which is a subset of all equilibria of the game. Nevertheless, if a game is potential game, we are guaranteed that an equilibrium exists.

V. INTERACTIVE TRAJECTORY PLANNING

In this section, we discuss how Theorem 1 and the special structure of agents' cost functions can be leveraged for interactive trajectory planning. In multi-agent settings such as navigation, the cost function of each agent i is typically composed of two types of cost terms: (i) Cost terms which are only dependent on the state and action of agent i itself such as input and state tracking costs, and (ii) cost terms that capture the mutual couplings between the pairs of agents, such as collision avoidance costs, which are dependent on the state of the agent i as well as other agents. For examples of these cost structures, see [1, 20, 21, 13]. For instance, the agent-specific tracking cost can be composed of a running cost C_i^{tr}

and a terminal cost $C_{i,T}^{tr}$ in the following form

$$
\begin{aligned}
C_i^{tr}(x_i, u_i) =& (x_i - x_i^{\text{ref}})^\intercal Q_i(x_i - x_i^{\text{ref}}) + \\
& (u - u_i^{\text{ref}})^\intercal R_i(u - u_i^{\text{ref}}),
\end{aligned} \tag{11}
$$

$$
C_{i,T}^{tr}(x_i, u_i) = (x_i(T) - x_i^{\text{ref}}(T))^\intercal Q_i(x_i(T) - x_i^{\text{ref}}(T)),
$$

where Q_i and R_i are weight matrices for penalizing state and control deviations from a reference trajectory $(x_i^{\text{ref}}, u_i^{\text{ref}})$. In a navigation setting, x_i^{ref} represents the goal state of agent i, and u_i^{ref} is the zero input signal for agent i to minimize its control effort. Note that in general, the matrices Q_i and R_i can be time-variant.

In addition to tracking cots, each agent i has coupling cost terms too that create mutual impact between the agents such as avoiding collisions with other agents. For each agent i, we let the collision avoidance cost term C_i^a be composed of pairwise collision avoidance costs C_{ij}^a for all $j \neq i$. For each $j \neq i$, C_{ij}^a penalizes agent i for colliding with or getting close to agent j. We assume that pairwise collision avoidance terms C_{ij}^a have the following structure

$$
C_{ij}^a(x_i, x_j) = \alpha_{ij}\left(d(x_i, x_j)\right) \tag{12}
$$

where α_{ij} is a function of the distance d between the agents. Note that d depends only on the states of agent i and j. Hence, for agent i, the running and terminal costs become

$$
L_i(x, u_i) = C_i^{tr}(x_i, u_i) + \sum_{j \neq i}^{N} C_{ij}^a(x_i, x_j) \tag{13}
$$

and

$$
S_i(x_i(T)) = C_{i,T}^{tr}(x_i(T)). \tag{14}
$$

The above cost structures can be more general. For example, C_i^{tr} can encode any other objective that agent i cares about such as minimum time to reach, minimum fuel, etc. Similarly, the inter-agent coupling terms can be more complicated than collision avoidance costs. But for simplicity, in the presentation of the paper, we specifically consider tracking and collision avoidance costs. Our key insight is that if the inter-agent collision avoidance costs (12) are symmetric for any two agents i and j, i.e., $C_{ij}^a(x_i, x_j) = C_{ji}^a(x_j, x_i)$ for all pair of agents; then, the game $\Gamma_{x_0}^T$ is a potential differential game. In other words, if any two agents penalize for collisions with each other similarly, the game is a potential game. Note that this does not imply that all agents penalize collisions similarly. We only need any two agents to penalize for getting close to each other similarly. In other words, the agents' sensitivity to collisions with each other should be symmetric.

We first show this through an example. Consider a differential game with three agents (see Fig. 2), and cost structure (13) and (14). We show that the game is potential if inter-agent cost terms are symmetric. We let p and $\bar{s}$ in Theorem 1 be

$$
\begin{aligned}
p(x, u) =& C_1^{tr}(x_1, u_1) + C_2^{tr}(x_2, u_2) + C_3^{tr}(x_3, u_3) \\
& + C_{12}^a(x_1, x_2) + C_{13}^a(x_1, x_3) + C_{23}^a(x_2, x_3),
\end{aligned} \tag{15}
$$

$$
\bar{s}(x, T) = C_{1,T}^{tr}(x_1(T)) + C_{2,T}^{tr}(x_2(T)) + C_{3,T}^{tr}(x_3(T)). \tag{16}
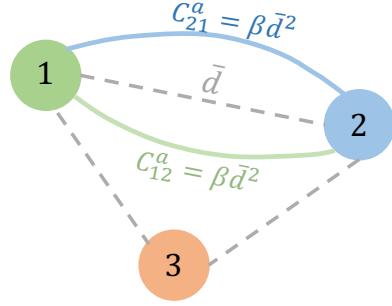$$

Fig. 2. Agent 1 penalizes for collision with agent 2 the same way that agent 2 penalizes for collision with agent 1. For example, if collision avoidance costs are in the form of the quadratic of the distance between the two agents, we must have $C_{12}^a = C_{21}^a = \beta \bar{d}^2$ where $\bar{d}$ is the distance between agents 1 and 2 and β is the weight for avoiding collisions. Agents 1 and 2 are symmetric in how they penalize for collisions. If collision avoidance costs are symmetric for all pair of agents, then the underlying game is a potential differential game.

For each agent i, we define the term c_i in (8) to be

$$
\begin{aligned}
c_1(x_2, x_3, u_2, u_3) =& \\
& -C_2^{tr}(x_2, u_2) - C_3^{tr}(x_3, u_3) - C_{23}^a(x_2, x_3), \\
c_2(x_1, x_3, u_1, u_3) =& \\
& -C_1^{tr}(x_1, u_1) - C_3^{tr}(x_3, u_3) - C_{13}^a(x_1, x_3), \\
c_3(x_1, x_2, u_1, u_2) =& \\
& -C_1^{tr}(x_1, u_1) - C_2^{tr}(x_2, u_2) - C_{12}^a(x_1, x_2).
\end{aligned} \tag{17}
$$

Likewise, for each agent i, we define the s_i term in (9) to be:

$$
\begin{aligned}
s_1(x_2(T), x_3(T)) &= -C_{2,T}^{tr}(x_2(T)) - C_{3,T}^{tr}(x_3(T)), \\
s_2(x_1(T), x_3(T)) &= -C_{1,T}^{tr}(x_1(T)) - C_{3,T}^{tr}(x_3(T)), \\
s_3(x_1(T), x_2(T)) &= -C_{1,T}^{tr}(x_1(T)) - C_{2,T}^{tr}(x_2(T)).
\end{aligned} \tag{18}
$$

The potential function (15) consists of the sum of the running costs of all agents and the sum of all pair-wise collision avoidance costs of agents with unordered pairs of distinct indices $\{i, j\}$. Similarly, $\bar{s}$ is the sum of the terminal costs of all agents. It is easy to show that if $C_{ij}^a(x_i, x_j) = C_{ji}^a(x_j, x_i)$ for any two agents i and j, we have:

$$
L_i(x(t), u(t), t) = p(x, u, t) + c_i(x_{-i}, u_{-i}, t), \quad 1 \leq i \leq 3, \tag{19}
$$

$$
S_i(x(T)) = \bar{s}(x(T)) + s_i(x_{-i}(T)), \quad 1 \leq i \leq 3. \tag{20}
$$

Thus, using Theorem 1, our game is a potential game. We can generalize this intuition through the following theorem:

Theorem 2. *Under dynamics (7), and cost structures (13) and (14), if for any two agents i and j, we have*

$$
C_{ij}^a(x_i, x_j) = C_{ji}^a(x_j, x_i), \tag{21}
$$

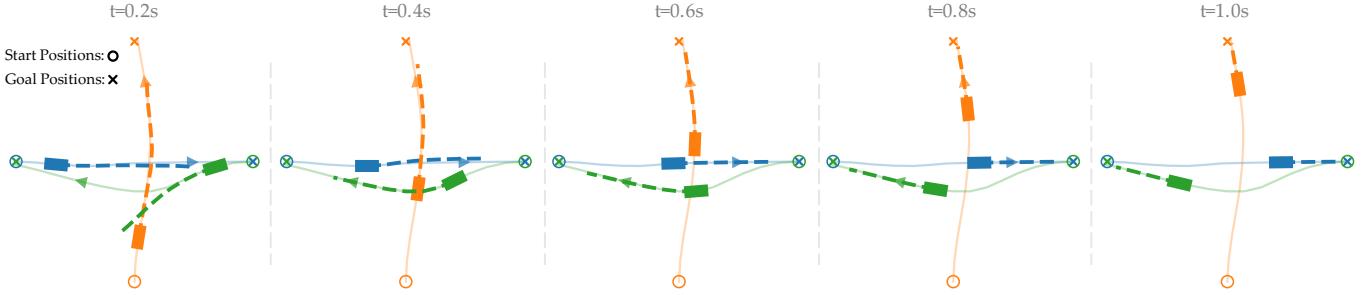*the underlying differential game $\Gamma_{x_0}^T$ is a potential differential*

Fig. 3. The snapshots of the trajectories found by our algorithm for an intersection scenario. Agents move from their start positions (denoted by circles) and move towards their goal positions (denoted by cross signs). The agents manage to successfully avoid collisions and resolve conflicts when they get close to each other. Dashed lines represent the planned trajectories over the receding horizon while the solid lines represent the final resulting trajectories

game with the following potential functions p and $\bar{s}$

$$p(x, u, t) = \sum_{i=1}^{N} C_i^{tr}(x_i, u_i) + \sum_{1 \leq i < j} C_{ij}^a(x_i, x_j),$$
$$\bar{s} = \sum_i C_{i,T}^{tr}(x_i), \tag{22}$$

where the second summation in p is over all unordered pairs of distinct agents' indices.

Proof: We prove this by showing that under assumption (21), cost structures (13) and (14) satisfy the conditions (8) and (9) in Theorem 1. Let p, and $\bar{s}$ be defined as introduced in (22). For each agent i, we define the agent-specific term c_i in (8) to be

$$c_i(x_{-i}, u_{-i}, t) = -\sum_{j \neq i} C_j^{tr}(x_j, u_j) - \sum_{\substack{1 \leq j < k \\ j \neq i,\, k \neq i}} C_{jk}^a(x_j, x_k),$$
$$\tag{23}$$

$$s_i(x_{-i}(T)) = -\sum_{j \neq i} C_{j,T}^{tr}(x_j), \tag{24}$$

where the first term in c_i is the negative of the sum of the tracking costs of all agents except for agent i, and the second term is the sum of all pairwise inter-agent costs for the unordered pairs of agents $\{j \neq i, k \neq i\}$. Similarly, s_i is the summation over all agents' terminal costs except for the agent i's terminal cost. Note that in the above, c_i and s_i are only a function of the states and inputs of agents other than agent i. Using the potential function (22) and the cost functions (23) and (24), it is easy to verify that for each agent i, we have

$$L_i(x, u_i) = C_i^{tr}(x_i, u_i) + \sum_{j \neq i}^{N} C_{ij}^a(x_i, x_j) \tag{25}$$

$$= p(x, u, t) + c_i(x_{-i}, u_{-i}, t), \tag{26}$$

and further

$$S_i(x(T)) = \bar{s}(x(T)) + s_i(x_{-i}(T)). \tag{27}$$

Hence, using Theorem 1, our game is a potential differential game. ∎

Using Theorem 2, under assumption (21), for finding equilibria of the game $\Gamma_{x_0}^T$, instead of solving a set of coupled optimal control problems (1), we can simply solve the optimal control problem (10). Consequently, we propose to solve the following optimal control problem in a receding horizon fashion:

$$\min_{u(\cdot)} \quad \int_0^T p(x(t), u(t), t)dt + \bar{s}(x(T))$$
$$\text{s.t.} \quad \dot{x}(t) = f(x(t), u(t), t), \tag{28}$$
$$x(0) = x_0.$$

where the potential functions p and $\bar{s}$ are as defined in (22). The significance of this result is that now we can use standard single-agent trajectory optimization algorithms to solve (28). For robots with general nonlinear dynamics, we can utilize any nonlinear trajectory optimization algorithm. We use iterative Linear Quadratic Regulator (iLQR) derived in [6, 30] for solving (28). We choose iLQR because of its success across different robotic applications [31, 32, 33].

Algorithm 1 Potential iLQR

1: **Inputs**
2: system dynamics (7), potential functions (22)
3: **Initialization**
4: initialize the control input using $u_i(.) = 0, 1 \leq i \leq N$
5: forward simulate (7) to obtain nominal trajectories $\eta = \{\bar{x}(t), \bar{u}_1(t), \cdots, \bar{u}_N(t)\}_{t \in [0,T]}$
6: **while** not converged **do**
7:     linear approximation of (7) around η
8:     quadratic approximation of (22) around η
9:     solve the backward recursion through Ricatti equation and obtain new control policies.
10:     forward simulate the controls and obtain the new nominal trajectories η.
11: **return** control input $u_i(.)$ for every agent i.

We start with initializing our controller. Then, starting

from an initial condition, we integrate the system dynamics (7) forward in time to obtain a nominal trajectory $\eta = \{\bar{x}(t), \bar{u}_1(t), \cdots, \bar{u}_n(t)\}_{t \in [0,T]}$. We linearize the the system dynamics (1) around η and further compute a quadratic approximation of the potential function in (28) around η. We use Ricatti equation to solve the resulting approximate Linear Quadratic Regulator problem to obtain a new nominal trajectory and repeat this process until convergence. The outline of our interactive trajectory planning algorithm is summarized in Algorithm 1.

In the next two sections, we demonstrate the success of our approach in generating intuitive interactive trajectories in both simulations and experiments.

VI. SIMULATION STUDIES

In this section, we demonstrate the performance of our algorithm in a planar navigation setting involving three agents at an intersection where each agent wants to reach its goal while avoiding collisions with other agents (See Fig. 3).

For each agent i, we use the following unicycle dynamics to model our vehicle dynamics:

$$\dot{p}_{x,i} = v_i \cos \theta_i, \quad \dot{p}_{y,i} = v_i \sin \theta_i,$$
$$\dot{\theta}_i = \omega_i, \qquad \dot{v}_i = a_i, \qquad (29)$$

where $p_{i,x}$ and $p_{i,y}$ are the x and y coordinates of the position of agent i in the 2D plane, v_i is the forward velocity for agent i, and θ_i is the heading of vehicle i. For each agent i, the state vector is $x_i = [p_{x,i}, p_{x,i}, \theta_i, v_i]$, and the input vector is $u_i = [\omega_i, a_i]$. We assume that each agent has a tracking cost C_i^{tr} in the form of (11) where we let the Q and R matrices be diagonal matrices with different weights on each diagonal entry, and each diagonal entry in Q and R matrices acts as a scaling weight for penalizing the corresponding state or control input. Following [20], we choose the inter-agent costs C_{ij}^a to be the following for any two agents i and j:

$$C_{ij}^a(x_i, x_j) = \begin{cases} (d_{ij} - d_{\text{prox}})^2 & \text{if } d_{ij} < d_{\text{prox}} \\ 0 & \text{else} \end{cases} \qquad (30)$$

where d_{ij} is the distance between vehicles i and j, and d_{prox} is the threshold distance above which no collision cost is incurred. We keep d_{prox} to be 2.4 meters in our simulations.

We run Algorithm 1 in a receding horizon fashion. Although we presented our algorithm in continuous time, we solved the backward and forward recursions of Algorithm 1 in discrete time for our implementation. We set our discrete time intervals to be 0.1 second. We further let the planning horizon for the receding horizon controller be 1 second. Fig. 3 illustrates the trajectories found by our planning algorithm.

We further compare the performance of our algorithm with the game solver iLQGames [20] by running a Monte Carlo study for the intersection problem. We consider 1000 random initial conditions and evaluate the solution times for both algorithms. We solve for trajectories with a 5-second prediction horizon. Fig. 4 shows the histograms of the results of our Monte Carlo study. For 1000 samples, the average convergence
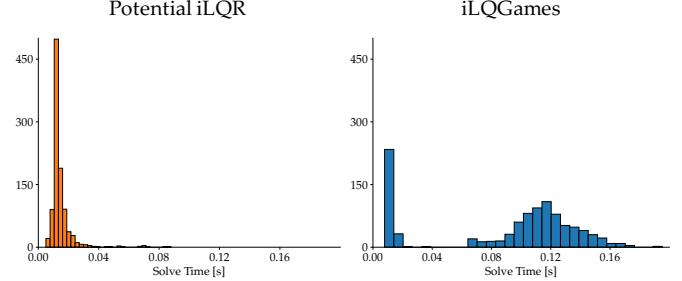

Fig. 4. Histogram data of our Monte Carlo study with 1000 random initial conditions. Potential-iLQR yields an average solution time of 14 ms with a standard deviation of 8 ms. For iLQGames, the average solution time is 89 ms with a standard deviation of 50 ms. Our approach is more than 6 times faster than iLQGames.

time of our algorithm is 14 ms with a standard deviation of 8 ms, whereas iLQGames has the average convergence time of 89 ms with a standard deviation of 50 ms. We also expect our algorithm to be more scalable in the number of agents. Since we solve a single optimal control problem using iLQR, we inherit its $\mathcal{O}(n^3)$ complexity that only scales with the dimension of the state space whereas the iLQGames has $\mathcal{O}(N^3 n^3)$ complexity that scales with both the state-space dimension and the number of agents.

VII. EXPERIMENTS

To demonstrate the real-time capabilities of our framework, we set up an experiment in hardware on two Crazyflie 2.0 quadcopters within the Robot Operating System (ROS) framework [34]. To obtain the state information, we use a Vicon motion capture system. To send waypoint commands to the quadcopters within the ROS environment, we use the Crazyswarm repository [35].

We use a 6D kinematic quadrotor model for quadcopters (see [36] for further information). Our state state vector is

$$x_i = [p_{x,i}, p_{y,i}, p_{z,i}, \phi_i, \theta_i, \psi_i],$$

where $p_{x,i}, p_{y,i}, p_{z,i}$ represent the position of the quadcopter body frame origin and ϕ_i, θ_i, ψ_i represent the roll, pitch, and yaw angles describing the orientation of the body frame of the quadcopter with respect to the inertial frame. For each quadcopter, the control inputs are

$$u_i = [v_{x,i}^b, v_{y,i}^b, v_{z,i}^b, p_i, q_i, r_i],$$

where $v_{x,i}^b$, $v_{y,i}^b$, and $v_{z,i}^b$ are the translational velocities expressed in the body frame. Similarly, p_i, q_i, and r_i represent the components of the angular velocities expressed in the body frame.

We run experiments with 2 different scenarios. In the first scenario, the two quadcopters start at the same height and switch their positions such that the starting position of one quadcopter becomes the goal position of the other (see Fig. 1). We set the xyz coordinate of the start position of one of the quadcopters to be $(0, 1, 2)$ while the start position
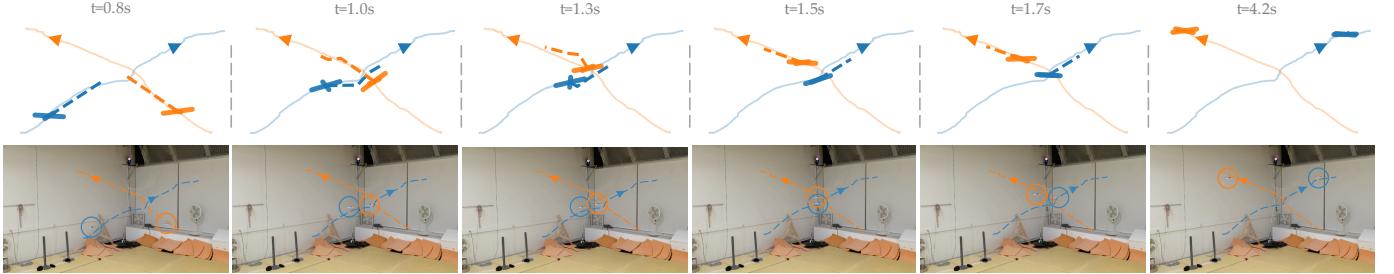
Fig. 5. The snapshots of the interactive trajectories found by our algorithm for two quadcopters. The start and goal positions of the quadcopters are set such that they need to avoid collisions while traversing their trajectories in 3D. Our algorithm generates intuitive trajectories where quadcopters manage to change altitudes for avoiding collisions. The plots in the first row were generated using the real-time data collected during the experiment. The second row includes the matching video frames of the same experiment. Each column represents the same time instance.

of the other quadcopter is $(1.5, 0, 2)$. To reach their goal positions, quadcopters need to avoid collisions and coordinate their motion. In the second scenario, the start position of the quadcopters is similar to the first scenario, but their goal positions are in different altitudes. The quadcopters need to traverse three-dimensional trajectories to reach their goals (see Fig. 5). In this experiment, one quadcopter starts at $(0, 1, 1.5)$ and reaches $(1.5, 0, 2.5)$ while the other quadcopter starts at $(1.5, 0, 1.5)$ and reaches $(0, 1, 2.5)$. We let our time step be 0.2 second and set the horizon length to be 1s for our receding horizon planner. With this setting, we are able to generate waypoints with 20 Hz update rate.

Fig. 1 demonstrates the trajectories of the quadcopters in the first scenario, and Fig. 5 shows the trajectories in the second scenario. As the figures illustrate, the quadcopters successfully avoid collisions and exhibit intuitive trajectories in three-dimensional space. In particular, as Fig. 1 and Fig. 5 show, the quadcopters change altitudes to avoid collisions with each other. These are indeed an extension of the planar interactive trajectories to the three-dimensional space where the agents can also change altitudes to avoid collisions. Note that both quadcopters maintain their nominal trajectories initially in both scenarios. However, as they get closer than the distance d_{prox}, the coupling between the cost functions becomes effective and the algorithm generates collision-free trajectories.

VIII. CONCLUSION

Summary. We showed that interactive trajectories can be found by solving a single optimal control problem instead of solving a set of coupled optimal control problems. In particular, for a class of multi-agent settings where agents have symmetric mutual cost couplings, we proved that the differential game underlying the interaction is a potential differential game whose equilibrium can be found by solving a single optimal control problem. We further showcased the applicability of our framework on a set of simulations and experiments in hardware.

Limitations and Future Work. The idea of reducing equilibria finding to solving a single-agent optimal control problem is achieved under symmetric pairwise couplings. However, it

is unclear whether interaction remains a potential game under general asymmetric pairwise couplings. We plan to investigate this further. We believe that for the asymmetric settings, the current algorithm still provides a very fast warm-starting method which provides a feasible relevant initial trajectory. We expect this to address one of the challenges in interactive trajectory planning using game solvers which are sensitive to proper initialization of trajectories.

Our proposed algorithm is a centralized planning algorithm. Ideally, in a game-theoretic setting such as interactions, we want the agents to be the decision makers. Thus, ultimately, we would like to achieve decentralized interactive planning. We expect that our proposed reduction to a single-agent optimal control problem enables us to address this problem by investigating the applications of decentralized control algorithms for minimizing the potential function of the interaction game.

APPENDIX A
PROOF OF THEOREM 1

This theorem was originally proved in [29]. For completeness, we are including our proof here. Let u^* be the solution to (10) and x^* be the corresponding optimal trajectory of the system state. Fix an agent i. Let $u_i \neq u_i^*$ be an open-loop control signal for agent i. Let x be the state trajectory that corresponds to the control signals (u_i, u_{-i}^*) in the original differential game. Since the state-input trajectory x^* and u^* are optimal for (10), we have

$$\int_0^T p\left(x^*(t), u^*(t), t\right) dt + \bar{s}\left(x^*(T)\right)$$
$$\leq \int_0^T p\left(x(t), \left(u_i(t), u_{-i}^*(t)\right), t\right) dt + \bar{s}\left(x(T)\right). \tag{31}$$

If we add

$$\int_0^T c_i\left(x_{-i}(t), u_{-i}^*(t), t\right) dt + s_i(x_{-i}(T)), \tag{32}$$

to both sides, we have

$$J_i^*(x_0, u^*) \leq J_i(x_0, u_i, u_{-i}^*). \tag{33}$$

It is important to note that because the dynamics (7) are decoupled, x_{-i} in (32) depends only on u_{-i}^*. Once u_{-i}^* is

fixed, (32) becomes a constant term added to both sides. Equation (33) holds for any agent i which is the definition of (6) which proves our theorem.

REFERENCES

[1] D. Sadigh, S. Sastry, S. A. Seshia, and A. D. Dragan, "Planning for autonomous cars that leverage effects on human actions." in *Robotics: Science and Systems*, vol. 2. Ann Arbor, MI, USA, 2016.

[2] D. Sadigh, S. S. Sastry, S. A. Seshia, and A. Dragan, "Information gathering actions over human internal state," in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2016, pp. 66–73.

[3] A. Fabrikant, C. Papadimitriou, and K. Talwar, "The complexity of pure nash equilibria," in *Proceedings of the thirty-sixth annual ACM symposium on Theory of computing*, 2004, pp. 604–612.

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

[5] D. Monderer and L. S. Shapley, "Potential games," *Games and economic behavior*, vol. 14, no. 1, pp. 124–143, 1996.

[6] W. Li and E. Todorov, "Iterative linear quadratic regulator design for nonlinear biological movement systems." in *ICINCO (1)*. Citeseer, 2004, pp. 222–229.

[7] B. D. Ziebart, N. Ratliff, G. Gallagher, C. Mertz, K. Peterson, J. A. Bagnell, M. Hebert, A. K. Dey, and S. Srinivasa, "Planning-based prediction for pedestrians," in *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2009, pp. 3931–3936.

[8] E. Schmerling, K. Leung, W. Vollprecht, and M. Pavone, "Multimodal probabilistic model-based planning for human-robot interaction," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2018, pp. 3399–3406.

[9] H. Nishimura, B. Ivanovic, A. Gaidon, M. Pavone, and M. Schwager, "Risk-sensitive sequential action control with multi-modal human trajectory forecasting for safe crowd-robot interaction," *arXiv preprint arXiv:2009.05702*, 2020.

[10] M. P. Vitus and C. J. Tomlin, "A probabilistic approach to planning and control in autonomous urban driving," in *52nd IEEE Conference on Decision and Control*. IEEE, 2013, pp. 2459–2464.

[11] H. Bai, S. Cai, N. Ye, D. Hsu, and W. S. Lee, "Intention-aware online pomdp planning for autonomous driving in a crowd," in *2015 ieee international conference on robotics and automation (icra)*. IEEE, 2015, pp. 454–460.

[12] A. Turnwald, D. Althoff, D. Wollherr, and M. Buss, "Understanding human avoidance behavior: interaction-aware decision making based on game theory," *International journal of social robotics*, vol. 8, no. 2, pp. 331–351, 2016.

[13] J. F. Fisac, E. Bronstein, E. Stefansson, D. Sadigh, S. S. Sastry, and A. D. Dragan, "Hierarchical game-theoretic planning for autonomous vehicles," in *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 9590–9596.

[14] W. Schwarting, A. Pierson, J. Alonso-Mora, S. Karaman, and D. Rus, "Social behavior for autonomous vehicles," *Proceedings of the National Academy of Sciences*, vol. 116, no. 50, pp. 24 972–24 978, 2019.

[15] M. Wang, Z. Wang, J. Talbot, J. C. Gerdes, and M. Schwager, "Game theoretic planning for self-driving cars in competitive scenarios." in *Robotics: Science and Systems*, 2019.

[16] R. Spica, E. Cristofalo, Z. Wang, E. Montijano, and M. Schwager, "A real-time game theoretic planner for autonomous two-player drone racing," *IEEE Transactions on Robotics*, vol. 36, no. 5, pp. 1389–1403, 2020.

[17] W. Schwarting, A. Pierson, S. Karaman, and D. Rus, "Stochastic dynamic games in belief space," *IEEE Transactions on Robotics*, 2021.

[18] H. Mukai, A. Tanikawa, I. Tunay, I. Katz, H. Schättler, P. Rinaldi, I. Ozcan, G. Wang, L. Yang, Y. Sawada *et al.*, "Sequential linear quadratic method for differential games," in *Proc. 2nd DARPA-JFACC Symposium on Advances in Enterprise Control*. Citeseer, 2000, pp. 159–168.

[19] A. Tanikawa, H. Mukai, and M. Xu, "Local convergence of the sequential quadratic method for differential games," *Transactions of the Institute of Systems, Control and Information Engineers*, vol. 25, no. 12, pp. 349–357, 2012.

[20] 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)*. IEEE, 2020, pp. 1475–1481.

[21] M. Wang, N. Mehr, A. Gaidon, and M. Schwager, "Game-theoretic planning for risk-aware interactive agents," 2020.

[22] S. L. Cleac'h, M. Schwager, and Z. Manchester, "Algames: A fast solver for constrained dynamic games," *arXiv preprint arXiv:1910.09713*, 2019.

[23] R. W. Rosenthal, "A class of games possessing pure-strategy nash equilibria," *International Journal of Game Theory*, vol. 2, no. 1, pp. 65–67, 1973.

[24] S. Zazo, J. Zazo, and M. Sánchez-Fernández, "A control theoretic approach to solve a constrained uplink power dynamic game," in *2014 22nd European Signal Processing Conference (EUSIPCO)*. IEEE, 2014, pp. 401–405.

[25] S. Zazo, S. Valcarcel, S. Matilde, J. Zazo *et al.*, "A new framework for solving dynamic scheduling games," in *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2015, pp. 2071–2075.

[26] G. Arslan, J. R. Marden, and J. S. Shamma, "Autonomous vehicle-target assignment: A game-theoretical formulation," 2007.

[27] J. R. Marden, G. Arslan, and J. S. Shamma, "Cooperative control and potential games," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 39, no. 6, pp. 1393–1407, 2009.

[28] U. O. Candogan, I. Menache, A. Ozdaglar, and P. A. Parrilo, "Near-optimal power control in wireless networks: A potential game approach," in *2010 Proceedings IEEE INFOCOM*. IEEE, 2010, pp. 1–9.

[29] A. Fonseca-Morales and O. Hernández-Lerma, "Potential differential games," *Dynamic Games and Applications*, vol. 8, no. 2, pp. 254–279, 2018.

[30] Y. Tassa, T. Erez, and E. Todorov, "Synthesis and stabilization of complex behaviors through online trajectory optimization," in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2012, pp. 4906–4913.

[31] J. Koenemann, A. Del Prete, Y. Tassa, E. Todorov, O. Stasse, M. Bennewitz, and N. Mansard, "Whole-body model-predictive control applied to the hrp-2 humanoid," in *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2015, pp. 3346–3351.

[32] N. Kitaev, I. Mordatch, S. Patil, and P. Abbeel, "Physics-based trajectory optimization for grasping in cluttered environments," in *2015 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2015, pp. 3102–3109.

[33] J. Chen, W. Zhan, and M. Tomizuka, "Constrained iterative lqr for on-road autonomous driving motion planning," in *2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC)*. IEEE, 2017, pp. 1–7.

[34] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, "Ros: an open-source robot operating system," in *ICRA workshop on open source software*, vol. 3, no. 3.2. Kobe, Japan, 2009, p. 5.

[35] J. A. Preiss, W. Honig, G. S. Sukhatme, and N. Ayanian, "Crazyswarm: A large nano-quadcopter swarm," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2017, pp. 3299–3304.

[36] F. Sabatino, "Quadrotor control: modeling, nonlinearcontrol design, and simulation," 2015.