Learning Hyperplanes for Multi-Robot Collision Avoidance in Space

Fernando Palafox[1], Yue Yu[2], and David Fridovich-Keil[1]

***Abstract*—A core challenge of multi-robot interactions is collision avoidance among robots with potentially conflicting objectives. We propose a game-theoretic method for collision avoidance based on rotating hyperplane constraints. These constraints ensure collision avoidance by defining separating hyperplanes that rotate around a keep-out zone centered on certain robots. Since it is challenging to select the parameters that define a hyperplane without introducing infeasibilities, we propose to learn them from an expert trajectory i.e., one collected by recording human operators. To do so, we solve for the parameters whose corresponding equilibrium trajectory best matches the expert trajectory. We validate our method by learning hyperplane parameters from noisy expert trajectories and demonstrate the generalizability of the learned parameters to scenarios with more robots and previously unseen initial conditions.**

I. Introduction

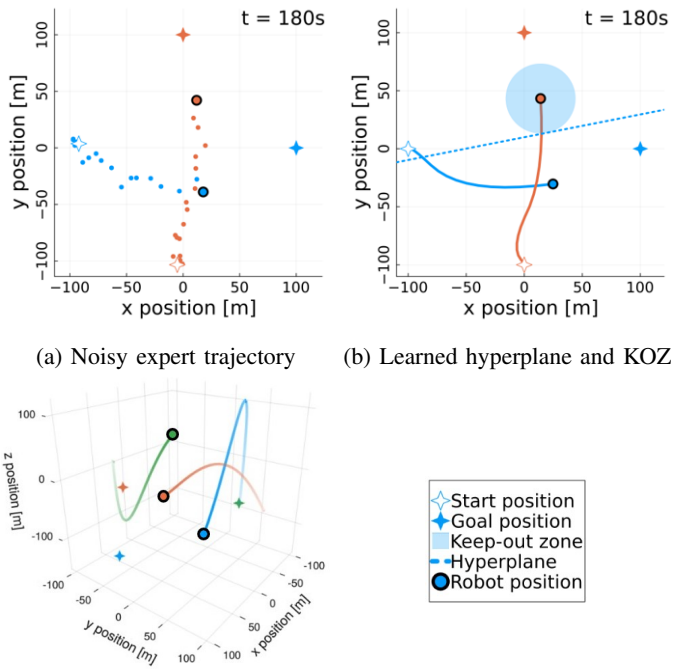
Space robotics play an important role in advancing important human endeavors, such as exploration, asteroid mining, telecommunication systems, scientific research, and establishing permanent human settlements on other planets. These robots can relieve humans from undertaking tasks in remote, inhospitable, and potentially hazardous environments, such as outer space or the surfaces of celestial bodies.

As space gets more crowded, we must manage the challenge of safe multi-robot interactions. On-orbit construction and/or maintenance of spacecraft, space stations, or scientific instruments will likely require close proximity interactions among many robots. It is crucial that we prevent collisions between these robots, particularly in situations where they may have conflicting objectives, as in construction or defense applications.

In much of the existing literature on robot collision avoidance, obstacles remain static or move in predetermined patterns that do not depend upon the robot's decisions. Existing methods typically bound the minimum distance between the robot and any obstacle and/or minimize a performance index that penalizes the robot for getting too close to any obstacle.

In the case of multi-robot collision avoidance, the problem is sometimes framed as a collaborative task where all robots have the common goal of avoiding each other. However, these methods do not model interactions for robots with conflicting objectives. A concrete example of such a scenario is a complex construction task in space, where robots seek different goal positions while operating close to each other.

Multi-robot interactions can be modeled using game theory, which provides an expressive mathematical framework to describe the behavior of several robots seeking to optimize individual objectives that depend upon one another's actions. *Dynamic* gamaes further account for strategic decision-making over time, and model the effects of sequential decisions via a dynamical system. Equilibria of these games yield state and control input trajectories that respect problem constraints and trade-off robots' individual objectives.



(a) Noisy expert trajectory (b) Learned hyperplane and KOZ

(c) Collision-free trajectories

Fig. 1: Our method uses noisy expert trajectories (Figure 1a) to learn the rotation rate and keep-out zone radius of a rotating hyperplane constraint (Figure 1b). We then use the same parameters to generate collision-free trajectories for a three-dimensional scenario with different initial conditions and more robots (Figure 1c).

We propose a *dynamic* game-theoretic method for multi-robot collision avoidance based on a set of rotating hyperplane constraints that define a separating hyperplane between an ego robot and an obstacle robot. This hyperplane rotates around a circular keep-out zone (KOZ) centered on the obstacle robot as seen in Figure 2 which the ego robot must avoid. Arbitrarily selecting hyperplane parameters may introduce infeasiblities to the game such as a rotation rate that requires robots to move faster than their actuators allow, or keep-out zone (KOZ) radii larger than the distance between robots at their initial positions. Therefore, we learn the parameters from expert trajectories that already avoid collisions, which could, e.g., be collected by recording human operators. With our method, we obtain collision-free trajectories by solving for the equilibrium

Authors are with [1]The Department of Aerospace Engineering and Engineering Mechanics and [2]The Oden Institute for Computational Engineering & Sciences at The University of Texas at Austin. Correspondence to fernandopalafox@utexas.edu.

This work was supported by the National Science Foundation under Grant No. 2211548, and by the Army Research Laboratory under Cooperative Agreement Number W911NF-23-2-0011.

trajectories of a game with learned hyperplane constraints between robots.

We evaluate the performance of our method in a simulated collision-avoidance scenario with robot dynamics modeled by the discrete-time Hill-Clohessy-Wiltshire equations for relative orbital motion [1]. We validate the learning capabilities of our method by learning hyperplane parameters from noisy expert data with known ground truth parameter values, and we show the generalizability of inferred parameters by applying them to different initial conditions and a scenario with more robots.

II. Related Work

In the domain of single-robot obstacle avoidance, Di Cairano, Park, and Kolmanovsky [2] formulated collision-avoidance constraints in terms of half-plane constraints to facilitate safe docking maneuvers. Building on this, Weiss, Baldwin, Erwin, *et al.* [3] defined a rotating hyperplane, effectively separating obstacles from spacecraft trajectories. In this work, we extend their ideas to the multi-robot case.

Existing game-theoretic approaches for collision avoidance include [4] where the authors presented a framework based on differential feedback games to address multi-agent collision avoidance. Their work integrates barrier terms proportional to proximity and solves for ϵ-Nash equilibria. Erdoğan, Innocenti, and Pollini [5] explored a game-theoretic strategy for formation flying and obstacle avoidance, decomposing complex avoidance tasks into smaller solvable components. Our work is different because it ensures collision avoidance via constraints in each player's trajectory.

In this work, we solve for unknown parameters in nonlinear constraints of an observed game-theoretic interaction (i.e., we solve the "inverse" game). Relevant literature includes approaches for inverse optimal control such as [6] where they solved for the objective and constraints of a manipulation task using Bellman's principle of optimality. Their work focuses on a linearly parameterized objective and selects constraints from a set of candidate constraints formed by the convex hull of all observed data points. Another example is [7] where they detailed two approaches for learning the parameters of an objective function based on maximum-likelihood estimation. Solving for unknown parameters in the inequality constraints was approached in Papadimitriou and Li [8] by employing an alternating optimization method that minimizes the KKT residual objective. Similarly, Chan and Kaw [9] solved for the parameters of linearly parameterized inequality constraints with two duality-based approaches: the first one minimizes the duality gap and the second one seeks to satisfy strong duality. Note that all of these approaches focus on solving for parameters in the objective and constraints of a single expert human (or robot), whereas our work extends to the case of a multi-robot interaction with nonlinear inequality constraints.

Existing approaches for solving an inverse game include [10], where the authors presented a method based on maximum-likelihood estimation to obtain unknown parameters in linearly parameterized objectives using noisy observations. Their work does not include inequality constraints but allows for collision-avoiding behavior via appropriate objective functions. Finally, Liu, Peters, and Alonso-Mora [11] presented a method for trajectory planning against opponents with unknown objectives and constraints by formulating the forward game as a differentiable parametric mixed complementarity problem and then using the gradient information to solve for the unknown parameters. Their work provides the foundation for the learning algorithm we present in this paper.

III. Problem Formulation

In this section, we describe the specifics of the multi-robot interaction, how it can be described as a game, and the Nash equilibrium solution concept.

A. Robot dynamics

We focus on the case of collision avoidance for robotic spacecraft operating in close proximity to each other. We model each robot's motion as a discrete-time, time-invariant dynamical system where the equations of motion are given by the discretized Hill-Clohessy-Wilshire equations for relative orbital motion [1]. In the following definitions, an overdot denotes a time-derivative, superscripts and subscripts denote indexing by robot and time, respectively. The position of robot i at time t is given by $\mathbf{p}_t^i = \begin{bmatrix} x_t^i & y_t^i & z_t^i \end{bmatrix}^\top \in \mathbb{R}^3$ and thrust forces in each direction are given by $F_x, F_y, F_z \in \mathbb{R}$. We define the set of all possible states and controls as $\mathcal{X} \subset \mathbb{R}^6$ and $\mathcal{U} \subset \mathbb{R}^3$, respectively. Then, for robot i at time t, state and controls are $\mathbf{x}_t^i \in \mathcal{X}$ and $\mathbf{u}_t^i \in \mathcal{U}$ where

$$\mathbf{x}_t^i := \begin{bmatrix} \mathbf{p}_t^{i\top} & \dot{\mathbf{p}}_t^{i\top} \end{bmatrix}^\top \tag{1a}$$

$$\mathbf{u}_t^i := \begin{bmatrix} F_{x,t}^i & F_{y,t}^i & F_{z,t}^i \end{bmatrix}^\top. \tag{1b}$$

We model discrete time as $t \in [T] = \{1, \dots, T\}$ and define the set of all robots as $[N] = \{1, \dots, N\}$. We use the following indexing shorthand throughout the paper:

$$\begin{aligned} \mathbf{x}_t &:= \begin{bmatrix} \mathbf{x}_t^{1\top} & \dots & \mathbf{x}_t^{N\top} \end{bmatrix}^\top \\ \mathbf{x} &:= \begin{bmatrix} \mathbf{x}_1^\top & \dots & \mathbf{x}_T^\top \end{bmatrix}^\top \\ \mathbf{x}^i &:= \begin{bmatrix} \mathbf{x}_1^{i\top} & \dots & \mathbf{x}_T^{i\top} \end{bmatrix}^\top \\ \mathbf{x}^{-i} &:= \begin{bmatrix} \mathbf{x}^{1\top} & \dots & \mathbf{x}^{(i-1)\top} & \mathbf{x}^{(i+1)\top} & \dots & \mathbf{x}^{N\top} \end{bmatrix}^\top \end{aligned}$$

We describe state dynamics with the time-invariant difference equation

$$\mathbf{x}_{t+1}^i = A\mathbf{x}_t^i + B\mathbf{u}_t^i, \tag{3}$$

where $A \in \mathbb{R}^{6\times 6}$ and $B \in \mathbb{R}^{3\times 3}$ are given in Appendix A. By defining $G_t^i := \mathbf{x}_{t+1}^i - (A\mathbf{x}_t^i + B\mathbf{u}_t^i)$, the equality constraints encoding dynamic feasibility for robot i are

$$G^i := \begin{bmatrix} G_1^{i\top} & \dots & G_{T-1}^{i\top} \end{bmatrix}^\top = \mathbf{0}. \tag{4}$$

Note that G^i is implicitly parameterized by the initial state $\mathbf{x}_1$.

We model finite thrusting capabilities by limiting thrust magnitude to $u_{\max} \in \mathbb{R}^+$ in any direction. Mathematically, this is given by the constraints

$$u_{\max}\mathbf{1}_3 + \mathbf{u} \geq 0 \tag{5a}$$

$$u_{\max}\mathbf{1}_3 - \mathbf{u} \geq 0 \tag{5b}$$

where $\mathbf{1}_3$ denotes a 3×1 vector of ones, and $\geq$ is evaluated element-wise.

B. Objective

The objective of every robot is to minimize the distance between its position at final time T and its goal position while also minimizing control effort. Mathematically, we define it as a scalar function $J^i : \mathbb{R}^{6NT} \times \mathbb{R}^{3NT} \to \mathbb{R}$,

$$J^i(\mathbf{x}, \mathbf{u}) := \left\|\mathbf{p}_T^i - \mathbf{p}_{\text{goal}}^i\right\|_2^2 + \xi^i \left\|\mathbf{u}^i\right\|_2^2, \tag{6}$$

where $\mathbf{p}_{\text{goal}}^i \in \mathbb{R}^3$ is robot i's goal position, and $\xi^i \in \mathbb{R}^+$ scales the weight of the second term relative to the weight of the first term.

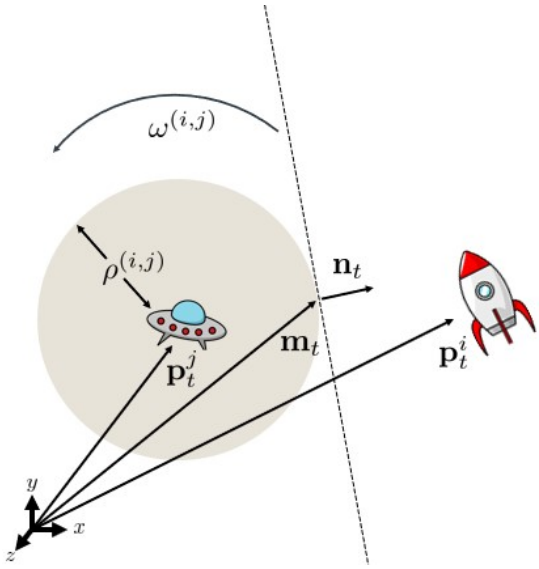


Fig. 2: For constraint $H^{(i,j)}$, the hyperplane (dotted line) rotates at a rate $\omega^{(i,j)}$ around the shaded KOZ with radius $\rho^{(i,j)}$.

C. Rotating hyperplane constraints

In the interest of computational efficiency, we follow [3] and formulate collision avoidance in terms of an affine hyperplane constraint that changes with time. We define Ω as the set of robot pairs for which a hyperplane constraint exists in the game. For example, $\Omega = \{(1,2),(1,3),(2,3)\}$ in a game with hyperplanes separating robots 1 and 2, 1 and 3, and 2 and 3. We denote Ω^i as the set of pairs that include robot i. For example, in the previous game $\Omega^2 = \{(1,2),(2,3)\}$. Robot pair (i,j) avoids collisions by defining a separating hyperplane that rotates at an angular velocity $\omega^{(i,j)} \in \mathbb{R}$ around a circular KOZ of radius $\rho^{(i,j)} \in \mathbb{R}^+$ centered on robot j, as illustrated in Figure 2. Mathematically, the constraint is defined as

$$H_t^{(i,j)} \geq 0, \forall t \in \{2,\dots,T\} \tag{7a}$$

$$\text{where } H_t^{(i,j)} := \mathbf{n}_t^\top (\mathbf{p}_t^i - \mathbf{m}_t) \tag{7b}$$

$$\mathbf{n}_t := \begin{bmatrix} \cos(\alpha^{(i,j)} + \omega^{(i,j)}(t-1)) \\ \sin(\alpha^{(i,j)} + \omega^{(i,j)}(t-1)) \end{bmatrix} \tag{7c}$$

$$\mathbf{m}_t := \mathbf{p}_t^i + \rho^{(i,j)} \mathbf{n}_t. \tag{7d}$$

$\alpha^{(i,j)} \in \mathbb{R}$ is the angle between the x-axis and $\mathbf{p}_t^i - \mathbf{p}_t^j$. Note that this formulation only depends on the planar components of the state. This limits the range of possible trajectories since robots' planar positions cannot overlap even if out-of-plane positions are at a safe distance. This trade-off can be justified by the reduced complexity of the constraint formulation and by noting that, in practice, this formulation is sufficient for collision-free trajectories in common space maneuvers e.g., docking [2].

In general, this interaction is non-cooperative because robots' objectives are arbitrary and can conflict. However, we assume that all agents are responsible for avoiding collisions, and therefore, robot i's complete set of constraints contains constraints for robots it wishes to avoid (e.g. $H^{(i,j)}$, a hyperplane centered on robot j that robot i wishes to avoid), as well as any constraints for robots trying to avoid robot i itself (e.g. $H^{(k,i)}$, a hyperplane centered on robot i that robot k wishes to avoid). Therefore, robot i's hyperplane constraints are expressed as

$$H^{(i,j)} := \begin{bmatrix} H_2^{(i,j)} & \dots & H_T^{(i,j)} \end{bmatrix}^\top \tag{8a}$$

$$\mathbf{H}^i := \{H^\omega \mid \omega \in \Omega^i\} \geq 0 \tag{8b}$$

where the set-builder notation in (8b) is abused to return a vector $\mathbf{H}^i \in \mathbb{R}^{(T-1)|\Omega|}$ and $\geq$ is evaluated element-wise.

D. Collision avoidance game

We model the multi-robot interaction with hyperplane constraints and thrust limits as an open-loop and infinite game $\Gamma(\theta) := (\mathbf{x}_1, \{J^i\}, \{\mathbf{G}^i\}, \{\mathbf{H}^i\}, u_{\max}, \theta, [N], [T])$, where $\theta := \{\omega, \rho, \xi\}$ contains the hyperplane parameters and cost weights for all robots. Each robot is only given the initial conditions $\mathbf{x}_1$ and then solves the coupled optimization problems

$$\mathbf{x}^{i*}, \mathbf{u}^{i*} \in \arg\min_{\mathbf{x}^i, \mathbf{u}^i} J^i(\mathbf{x}, \mathbf{u}) \tag{9a}$$

$$\text{subject to } \mathbf{G}^i = \mathbf{0} \tag{9b}$$

$$\mathbf{H}^i \geq \mathbf{0} \tag{9c}$$

$$u_{\max}\mathbf{1}_3 + \mathbf{u} \geq 0 \tag{9d}$$

$$u_{\max}\mathbf{1}_3 - \mathbf{u} \geq 0. \tag{9e}$$

Note that this problem is implicitly parameterized by θ, which enters both J^i and $\mathbf{H}^i$.

We seek generalized Nash equilibrium (GNE) solutions [12] in which no robot has a unilateral incentive to change its current strategy. This concept captures the non-cooperative nature of the interaction between rational robots with potentially conflicting objectives and is defined as follows:

Definition 1 (Generalized Nash equilibrium)**.** *A set of strategies $\mathbf{u}^* := \{\mathbf{u}^{1*}, \dots, \mathbf{u}^{N*}\}$ is a generalized Nash equilibrium of the game $\Gamma(\theta)$ if it satisfies*

$$J^i(\mathbf{x}^*, \mathbf{u}^*) \leq J^i(\mathbf{x}, \{\mathbf{u}^i, \mathbf{u}^{-i*}\}), \forall i \in [N] \tag{10}$$

for any feasible deviation $(\mathbf{x}, \mathbf{u})$ with $\mathbf{u}^{-i*}$ denoting all but robot i's controls.

In general, computing Nash equilibria is computationally intractable [13], so we solve for trajectories $(\mathbf{x}, \mathbf{u})$ that locally satisfy the equilibrium conditions (10) to first order. Assuming linear independence of the constraints, solutions for (9) must satisfy the intersection of every player's Karush–Kuhn–Tucker

(KKT) conditions [12]. To express these, we introduce Lagrange multipliers $\mathbf{w}, \mathbf{v}, \lambda_{\text{hi}}, \lambda_{\text{lo}}$ and write robot i's Lagrangian as

$$\begin{aligned}\mathcal{L}^i = \; & J^i - \mathbf{w}^{i\top}\mathbf{G}^i - \mathbf{v}^{i\top}\mathbf{H}^i \\ & - \lambda_{\text{hi}}^{i\top}(u_{\max}\mathbf{1}_3 + \mathbf{u}^i) - \lambda_{\text{lo}}^{i\top}(u_{\max}\mathbf{1}_3 - \mathbf{u}^i)\end{aligned} \tag{11}$$

where $\mathbf{v}^i = \{\mathbf{v}^\omega \mid \omega \in \Omega^i\}$. Then, the KKT conditions are

$$\forall i \in [N] \begin{cases} \nabla_{\mathbf{x}^i}\mathcal{L}^i = 0 \\ \nabla_{\mathbf{u}^i}\mathcal{L}^i = 0 \\ \mathbf{G}^i = 0 \\ 0 \le \mathbf{H}^i \perp \mathbf{v}^i \ge 0 \\ 0 \le u_{\max}\mathbf{1}_3 + \mathbf{u} \perp \lambda_{\text{hi}} \ge 0 \\ 0 \le u_{\max}\mathbf{1}_3 - \mathbf{u} \perp \lambda_{\text{lo}} \ge 0. \end{cases} \tag{12}$$

Robots share multipliers associated with hyperplane constraints $\mathbf{H}^i$ to encode shared responsibility for constraint feasibility. For example, $(i, j) \in \Omega$ implies that $H^{(i,j)} \in \mathbf{H}^i \cap \mathbf{H}^j$ so $\mathbf{v}^i$ and $\mathbf{v}^j$ will share some elements.

It can be shown that solving (12) is equivalent to solving a mixed complementarity problem (MCP) [12], parameterized by θ and denoted as $\text{MCP}(\theta)$. The solution for $\text{MCP}(\theta)$ is a vector $z^* \in \mathbb{R}^n$ that can be expressed in terms of the decision variables in (12) as

$$z^* = [\mathbf{x}^{*\top}, \mathbf{u}^{*\top}, \mathbf{w}^{*\top}, \mathbf{v}^{*\top}, \lambda_{\text{hi}}^{*\top}, \lambda_{\text{lo}}^{*\top}]^\top. \tag{13}$$

This equivalence is very convenient because, as shown in [11], it is possible to differentiate z^* with respect to θ. This fact will be fundamental in the development of the hyperplane parameter learning algorithm, as discussed in the next section.

IV. Learning Hyperplane Parameters

Selecting hyperplane parameters without introducing infeasibilities into the game can be challenging, particularly due to the thrust limit constraints (9d) and (9e). In games with two robots, selecting parameters can be done through judicious trial and error. However, the increasingly complicated geometry rules out this approach for games with more robots. Instead, we propose to learn these parameters from expert trajectories known to satisfy all constraints, such as the trajectory of two small satellites controlled by expert human operators.

Concretely, we begin with an observed state trajectory $\tilde{\mathbf{x}}$, assumed to represent the equilibrium behavior of a parametric collision-avoidance game $\Gamma(\theta)$ (where all or only some of the elements of θ are unknown), and then solve an "inverse" game to recover the unknown parameters. The solution to the inverse game is the set of constraint parameters for which the resulting equilibrium best matches the observed trajectories, i.e.

$$\min_{\theta, \mathbf{x}} \ell(\mathbf{x}, \tilde{\mathbf{x}}) \text{ subject to } \mathbf{x} \in \mathbb{X} \tag{14}$$

where $\ell : \mathbb{R}^{6NT} \times \mathbb{R}^{6NT} \to \mathbb{R}, \ell(\mathbf{x}, \tilde{\mathbf{x}}) := \|\mathbf{x} - \tilde{\mathbf{x}}\|_2^2$, and

$$\begin{aligned}\mathbb{X} = \{\mathbf{x} \in \mathbb{R}^{6NT} \mid \exists \mathbf{u}, \mathbf{w}, \mathbf{v}, \lambda_{\text{hi}}, \lambda_{\text{lo}}, \text{ such that} \\ z = [\mathbf{x}^\top, \mathbf{u}^\top, \mathbf{w}^\top, \mathbf{v}^\top, \lambda_{\text{hi}}^\top, \lambda_{\text{lo}}^\top]^\top \text{ solves } \text{MCP}(\theta)\}\end{aligned} \tag{15}$$

We follow the approach in [11] and solve for the hyperplane parameters by descending the gradient of ℓ with respect to θ as shown in Algorithm 1. We found that scaling the gradient with an Adam optimizer [14] greatly improved numerical stability.

Algorithm 1: Learn hyperplane parameters

Input: Initial θ, learning rate α, decay rates β_1, β_2, convergence tolerance $\epsilon > 0$.
while $\|\nabla_\theta \ell\| \ge \epsilon$ **do**
  $(z^*, \nabla_\theta z^*) \leftarrow$ solve $\text{MCP}(\theta)$
  $\nabla_\theta \ell \leftarrow \nabla_{\mathbf{x}} \ell^\top \nabla_{z^*} \mathbf{x} \nabla_\theta z^*$
  $\theta \leftarrow \theta - \text{Adam}(\nabla_\theta \ell; \alpha, \beta_1, \beta_2)$
end
return θ

In the case of rotating hyperplane constraints, gradient descent yields a loss-optimal set of hyperplane parameters without requiring that all hyperplane constraints be active. If a constraint is never active, the gradient of the loss with respect to the constraint parameters will be zero (even if the parameters are not optimal), rendering gradient descent useless. We avoid this issue because a set of hyperplane parameters affects $T - 1$ constraints as defined in (7a). Although most of these constraints are not active, at least one of them must be if a collision is avoided because of the hyperplane constraints. Therefore, the gradient of the loss function with respect to the hyperplane parameters will only be zero at optimal points or in the case where the hyperplane constraint is never active at any point in the game. In practice, the latter case was very rare and never occurred in the Monte Carlo experiments presented in Section V. Therefore, we can use gradient descent to converge on a set of loss-optimal parameters.

We solve (14) with a Julia implementation of Algorithm 1 that is publicly available.[1] We express the MCP using ParametricMCPs.jl [15], a mathematical programming layer that compiles a differentiable MCP parameterized by a runtime vector. Within ParametricMCPs.jl, the MCP is solved via the PATH solver [16].

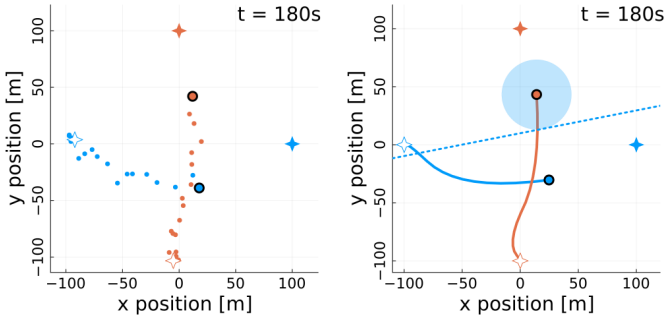
V. Experimental Results

In this section, we present experimental results to demonstrate the performance of our method. First, we validate learning capabilities by presenting the performance of our method when learning from noise-corrupted expert trajectories. Then, we demonstrate the generalizability of the learned parameters by presenting a six-robot collision-free trajectory using the parameters learned from a two-robot expert trajectory. Finally, we characterize the sensitivity of the learned parameters to perturbations in the initial velocity of the robots.

A. Learning from noisy expert trajectories

We validate the performance of Algorithm 1 by presenting the difference between ground truth hyperplane parameters and parameters learned from a noisy expert trajectory. For the expert trajectory in this experiment, we use an equilibrium state trajectory for a two-dimensional, two-robot game with initial conditions and ground truth parameters given in Table I

[1] github.com/CLeARoboticsLab/InverseHyperplanes.jl



(a) Noisy trajectory ($\sigma = 5$) (b) Learned hyperplane

Fig. 3: Using noisy expert trajectories, our method learns hyperplane parameters and recovers the corresponding equilibrium trajectory. Legend in Figure 1.

of Appendix B. Two-dimensional dynamics are trivially implemented by noting that the out-of-plane position dynamics in (3) are decoupled from the planar position.

We generate 400 noise-corrupted trajectories by adding isotropic additive white Gaussian noise to the expert state trajectory at 20 levels of standard deviation σ from 0.0 to 20.0 and 20 samples for each level. Figure 3a shows a snapshot of a typical noisy state trajectory, and Figure 3b shows the learned hyperplane and corresponding equilibrium trajectory.

We show the accuracy of the learned hyperplane parameters by presenting a comparison between ground-truth and learned parameters in Figure 4 as a function of noise level. Learned rotation rates are very close to the ground truth even when the trajectory is corrupted by extreme noise levels. The learned KOZ radius is not as accurate as noise increases, but this is likely because the solution is not as sensitive to changes in radius ρ as it is to changes in rotation rate ω. That is, changes in ω require a larger change in the trajectory than changes in ρ, and large changes to the controls may be infeasible due to actuator limits (Equations (9d) and (9e)).

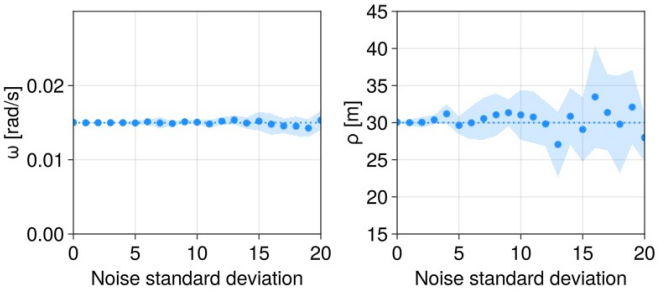


Fig. 4: Learned hyperplane parameters vs. noise level. A dotted line is the ground truth value, dots are median inferred values, and ribbons show the interquartile range.

We show that the learned parameters (14) correspond to an equilibrium trajectory that closely matches ground truth by computing a measure of its difference from the noise-free expert trajectory. Concretely, we report the reconstruction error D:

$$D(\theta_{\text{truth}}, \theta_{\text{learned}}) = \frac{1}{NT} \sum_{i \in [N]} \sum_{t \in [T]} \left\| \mathbf{p}^i_{\text{truth},t} - \mathbf{p}^i_{\text{learned},t} \right\|_2^2 \tag{16}$$

where $\mathbf{p}^i_{\text{truth,t}}$ is robot i's position at time t as given by the noise-free expert data with ground truth parameters θ_{truth}, and $\mathbf{p}^i_{\text{learned},t}$ is robot i's position at time t as given by a Nash equilibrium of a game parameterized by the learned parameters θ_{learned}. Figure 5 shows the median value of D for all trials as well the associated interquartile range. As expected, the error increases as noise increases. However, at reasonable noise values ($\sigma < 5$), reconstruction error is relatively low, particularly when considering the scale of the problem (200m between initial and goal positions).

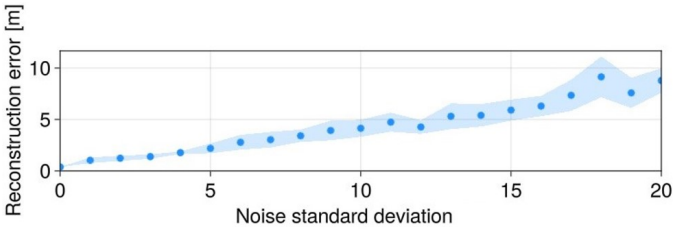


Fig. 5: Reconstruction error (16) vs. noise level. Dots are median error values, and ribbons show the interquartile range.

B. *Generalizability of learned parameters*

Hyperplane parameters from expert trajectories generalize to scenarios with more robots. For example, one can use the parameters learned from a two-robot expert trajectory to generate a collision-free trajectory for a six-robot game, as shown in Figure 6. Our method is easily extended to games with more robots by simply appending the KKT conditions for each additional robot to (12) and defining an appropriate set of robot pairs Ω.

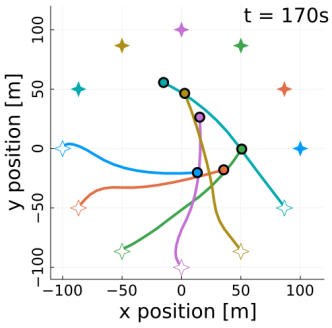


Fig. 6: Collision-free trajectory for a six-robot game. Hyperplanes and KOZs are not shown for clarity. We adjust initial and goal positions, add additional pairs to Ω, and otherwise reuse the parameters in Appendix B.

Learned parameters can also be used to generate collision-free trajectories for scenarios with previously unseen initial conditions. Anecdotally, trajectory generation from learned parameters is robust to changes in initial position but sensitive to changes in initial velocities. To quantify this sensitivity,

we perturb the initial velocity with noise at varying levels and then attempt to construct a collision-free trajectory. In Figure 7 we report trajectory generation convergence success as a function of velocity noise level given a nominal velocity. In this experiment, the expert trajectory has an initial velocity of zero.

Results show that learned parameters readily generalize to scenarios with more robots as long as the initial velocity is close to the initial velocity of the expert trajectories. This may not be an issue under the assumption that conditions encountered in practice will be similar to those in the expert trajectories. We also note that the parameters are more sensitive to changes in initial velocity as we include more robots. This is likely because the more robots we include, the more crowded the space becomes, and it naturally becomes harder to optimize their trajectory.

Finally, since the hyperplanes only constrain planar positions and the dynamics of the out-of-plane positions are decoupled from the planar dynamics, collision-free trajectory generation can be trivially extended to three dimensions. We present an example trajectory in Figure 1c.

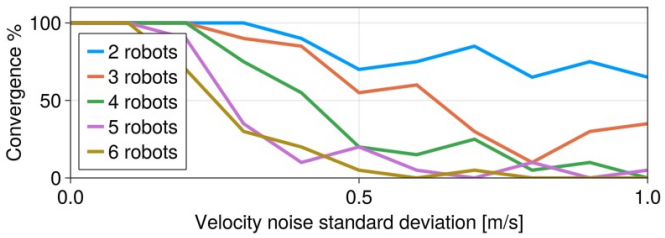


Fig. 7

Fig. 8: Success of collision-free trajectory generation as a function of velocity noise level. The initial velocity of the expert trajectory is zero and we ran 20 trials at each noise level.

VI. Conclusion

We propose a game-theoretic method for multi-robot collision avoidance in space robotic applications. Our method is different from existing techniques as it extends collision avoidance with rotating hyperplanes to a multi-robot setting, using parameters learned from expert trajectories. Unlike previous inverse game solvers, we explicitly solve for parameters in the inequality constraints.

We conduct extensive numerical simulations to validate learning capabilities from noise-corrupted expert trajectories and demonstrate the generalizability of the learned parameters to games with different initial conditions and/or more robots. Future areas of work include applying this method to other dynamical systems and using real expert trajectories like tracked on-orbit maneuvers.

References

[1] C. Jewison and D. W. Miller, "Probabilistic trajectory optimization under uncertain path constraints for close proximity operations," *Journal of Guidance, Control, and Dynamics*, vol. 41, no. 9, pp. 1843–1858, 2018. DOI: 10.2514/1.G003152. eprint: https://doi.org/10.2514/1.G003152. [Online]. Available: https://doi.org/10.2514/1.G003152.

[2] S. Di Cairano, H. Park, and I. Kolmanovsky, "Model predictive control approach for guidance of spacecraft rendezvous and proximity maneuvering," en, *International Journal of Robust and Nonlinear Control*, vol. 22, no. 12, pp. 1398–1427, 2012, ISSN: 1099-1239. DOI: 10.1002/rnc.2827.

[3] A. Weiss, M. Baldwin, R. S. Erwin, and I. Kolmanovsky, "Model predictive control for spacecraft rendezvous and docking: Strategies for handling constraints and case studies," *IEEE Transactions on Control Systems Technology*, vol. 23, no. 4, pp. 1638–1647, Jul. 2015, ISSN: 1558-0865. DOI: 10.1109/TCST.2014.2379639.

[4] T. Mylvaganam, M. Sassano, and A. Astolfi, "A differential game approach to multi-agent collision avoidance," *IEEE Transactions on Automatic Control*, vol. 62, no. 8, pp. 4229–4235, Sep. 2017, ISSN: 1558-2523. DOI: 10.1109/TAC.2017.2680602.

[5] M. E. Erdoğan, M. Innocenti, and L. Pollini, "Obstacle avoidance for a game theoretically controlled formation of unmanned vehicles," *IFAC Proceedings Volumes*, vol. 44, no. 1, pp. 6023–6028, 2011, 18th IFAC World Congress, ISSN: 1474-6670. DOI: https://doi.org/10.3182/20110828-6-IT-1002.03043. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1474667016445696.

[6] M. Menner, P. Worsnop, and M. N. Zeilinger, "Constrained inverse optimal control with application to a human manipulation task," *IEEE Transactions on Control Systems Technology*, vol. 29, no. 2, pp. 826–834, 2019.

[7] M. Menner and M. N. Zeilinger, "Maximum likelihood methods for inverse learning of optimal controllers**this work was supported by the swiss national science foundation under grant no. pp00p2157601 / 1.," *IFAC-PapersOnLine*, vol. 53, no. 2, pp. 5266–5272, 2020, 21st IFAC World Congress, ISSN: 2405-8963. DOI: https://doi.org/10.1016/j.ifacol.2020.12.1206. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2405896320316025.

[8] D. Papadimitriou and J. Li, *Constraint inference in control tasks from expert demonstrations via inverse optimization*, 2023. arXiv: 2304.03367 [cs.RO].

[9] T. C. Chan and N. Kaw, "Inverse optimization for the recovery of constraint parameters," *European Journal of Operational Research*, vol. 282, no. 2, pp. 415–427, 2020, ISSN: 0377-2217. DOI: https://doi.org/10.1016/j.ejor.2019.09.027. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0377221719307830.

[10] L. Peters, V. Rubies-Royo, C. J. Tomlin, L. Ferranti, J. Alonso-Mora, C. Stachniss, and D. Fridovich-Keil, "Online and offline learning of player objectives from partial observations in dynamic games," en, *The International Journal of Robotics Research*, p. 02 783 649 231 182 453, Jun. 2023, ISSN: 0278-3649. DOI: 10.1177/02783649231182453.

[11] X. Liu, L. Peters, and J. Alonso-Mora, "Learning to play trajectory games against opponents with unknown objectives," no. arXiv:2211.13779, Mar. 2023, arXiv:2211.13779 [cs]. [Online]. Available: http://arxiv.org/abs/2211.13779.

[12] *Finite-Dimensional Variational Inequalities and Complementarity Problems* (Springer Series in Operations Research and Financial Engineering), en. New York, NY: Springer, 2004, ISBN: 978-0-387-95580-3. DOI: 10.1007/b97543. [Online]. Available: http://link.springer.com/10.1007/b97543.

[13] C. Daskalakis, P. W. Goldberg, and C. H. Papadimitriou, "The complexity of computing a nash equilibrium," *SIAM Journal on Computing*, vol. 39, no. 1, pp. 195–259, 2009. DOI: 10.

1137/070699652. eprint: https://doi.org/10.1137/070699652. [Online]. Available: https://doi.org/10.1137/070699652.

[14] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," no. arXiv:1412.6980, Jan. 2017, arXiv:1412.6980 [cs]. DOI: 10.48550/arXiv.1412.6980. [Online]. Available: http://arxiv.org/abs/1412.6980.

[15] L. Peters, *Parametricmcps*, Julia, Sep. 2023. [Online]. Available: https://github.com/lassepe/ParametricMCPs.jl.

[16] S. P. Dirkse and M. C. Ferris, "The path solver: A nommonotone stabilization scheme for mixed complementarity problems," *Optimization Methods and Software*, vol. 5, no. 2, pp. 123–156, Jan. 1995, ISSN: 1055-6788. DOI: 10.1080/10556789508805606.

[17] C. Zagaris, H. Park, J. Virgili-Llop, R. Zappulla, M. Romano, and I. Kolmanovsky, "Model predictive control of spacecraft relative motion with convexified keep-out-zone constraints," *Journal of Guidance, Control, and Dynamics*, vol. 41, no. 9, pp. 2054–2062, Sep. 2018, ISSN: 0731-5090. DOI: 10.2514/1.G003549.

APPENDIX A
ROBOT DYNAMICS

Here we present the complete robot dynamics (3) as a linear-time invariant system of equations representing the discrete-time Hill-Clohessy-Wiltshire equations for relative orbital motion as defined in Jewison and Miller [1]. This system describes the robot's state as a function of applied controls and state at the previous time step.

$$\mathbf{x}_{t+1}^i = A\mathbf{x}_t^i + B\mathbf{u}_t^i$$

$$A = \begin{bmatrix} 4-3\cos(n\Delta t) & 0 & 0 & \dots \\ 6(\sin(n\Delta t)-n\Delta t) & 1 & 0 & \\ 0 & 0 & \cos(n\Delta t) & \\ 3n\sin(n\Delta t) & 0 & 0 & \\ -6n(1-\cos(n\Delta t)) & 0 & 0 & \\ 0 & 0 & -n\sin(n\Delta t) & \end{bmatrix}$$

$$\dots \begin{bmatrix} \frac{1}{n}\sin(1-n\Delta t) & \frac{2}{n}(1-\cos(n\Delta t)) & 0 \\ -\frac{2}{n}(1-\cos(n\Delta t)) & \frac{1}{n}(4\sin(n\Delta t)-3n\Delta t) & 0 \\ 0 & 0 & \frac{1}{n}\sin(n\Delta t) \\ \cos(n\Delta t) & 2\sin(n\Delta t) & 0 \\ -2\sin(n\Delta t) & 4\cos(n\Delta t)-3 & 0 \\ 0 & 0 & \cos(n\Delta t) \end{bmatrix}$$

$$B = \frac{1}{m}\begin{bmatrix} \frac{1}{n}\sin(n\Delta t) & \frac{2}{n}(1-\cos(n\Delta t)) & \dots \\ -\frac{2}{n^2}(n\Delta t-\sin(n\Delta t)) & \frac{4}{n^2}(1-\cos(n\Delta t))-\frac{3}{2}\Delta t^2 & \\ 0 & 0 & \\ \frac{1}{n}\sin(n\Delta t) & \frac{2}{n}(1-\cos(n\Delta t) & \\ -\frac{2}{n}(1-\cos(n\Delta t)) & \frac{4}{n}\sin(n\Delta t)-3\Delta t & \\ 0 & 0 & \end{bmatrix}$$

$$\dots \begin{bmatrix} 0 \\ 0 \\ \frac{1}{n^2}(1-\cos(n\Delta t)) \\ 0 \\ 0 \\ \frac{1}{n}\sin(n\Delta t) \end{bmatrix}$$

where Δt is the discretization interval and m is the satellite mass. $n \approx \sqrt{G/a}$ is the angular speed required for a body to complete one orbit (mean motion) and is defined in terms of the universal constant of gravitation G and the orbit semi-major axis a. We assume a circular orbit around Earth, in this case, $a = (\text{orbital altitude} + \text{ Earth's radius})$.

APPENDIX B
SIMULATION PARAMETERS

In Table I, we include relevant simulation parameters for the two-dimensional, two-robot inverse game solved in Section V. In the six-robot trajectory in Figure 6, we adjust initial and goal positions, add additional pairs to Ω, and reuse parameters otherwise.

	Parameter	Value	Units
Game	Players	2	
	Initial state	$[0\ 100\ 0\ 0\ -100\ 0\ 0\ 0]^\top$	m
	Goal positions	$[0\ -100]^\top, [100\ 0]^\top$	m
	Orbital altitude	400	km
	Satellite mass	100	kg
	$u_{\max}$	1.0	N
	ξ^i	0.0001	
	Ω	$\{(1,2)\}$	
	Δt	5.0	s
	Total time	220	s
Ground truth	$\omega^{(1,2)}$	0.015	rad/s
	$\rho^{(1,2)}$	30.0	m
Inverse game	Initial guess $\omega^{(1,2)}$	0.008	rad/s
	Initial guess $\rho^{(1,2)}$	10.0	m
	Num. of grad. steps	30	
Experiments	σ range	$0:5:40$	m
	Trials	20	

TABLE I: Simulation parameters for hyperplane learning using noisy expert trajectories. The initial guess for the hyperplane rotation rate has empirically been shown to work well in a single-robot scenario [17].