Learning Two-agent Motion Planning Strategies from Generalized Nash Equilibrium for Model Predictive Control

Hansung Kim[1]   HANSUNG@BERKELEY.EDU
Edward L. Zhu[2]   EDWARD.ZHU@PLUS.AI
Chang Seok Lim[1]   CSHIGH22@BERKELEY.EDU
Francesco Borrelli[1]   FBORRELLI@BERKELEY.EDU
[1]*University of California, Berkeley*
[2]*PlusAI, Inc*

Editors: N. Ozay, L. Balzano, D. Panagou, A. Abate

Abstract

We introduce an Implicit Game-Theoretic MPC (IGT-MPC), a decentralized algorithm for two-agent motion planning that uses a learned value function that predicts the game-theoretic interaction outcomes as the terminal cost-to-go function in a model predictive control (MPC) framework, guiding agents to implicitly account for interactions with other agents and maximize their reward. This approach applies to competitive and cooperative multi-agent motion planning problems which we formulate as constrained dynamic games. Given a constrained dynamic game, we randomly sample initial conditions and solve for the generalized Nash equilibrium (GNE) to generate a dataset of GNE solutions, computing the reward outcome of each game-theoretic interaction from the GNE. The data is used to train a simple neural network to predict the reward outcome, which we use as the terminal cost-to-go function in an MPC scheme. We showcase emerging competitive and coordinated behaviors using IGT-MPC in scenarios such as two-vehicle head-to-head racing and un-signalized intersection navigation. IGT-MPC offers a novel method integrating machine learning and game-theoretic reasoning into model-based decentralized multi-agent motion planning.

Keywords: Multi-agent Motion Planning; Game-theory; Supervised Learning; Model Predictive Control

1. Introduction

Multi-agent motion planning is a fundamental challenge for autonomous systems navigating complex and dynamic environments, especially when an ego agent must adapt in real-time to the actions of other agents. Multi-agent settings are generally categorized as competitive, cooperative, or non-cooperative. In competitive settings, agents pursue conflicting goals, often seeking outcomes that favor themselves at the expense of their opponents. An example, as shown in Fig. 1a, is head-to-head car racing, where competing agents attempt to finish a race ahead of their opponent. In cooperative settings, agents collaborate to achieve shared objectives, such as navigating an un-signalized intersection with connected autonomous vehicles (CAVs), which must coordinate to move safely and efficiently without causing delays or collisions, as shown in Fig. 1b. Lastly, in non-cooperative settings, agents act independently to maximize their own interests without collaboration. While our approach applies to all three multi-agent environment settings, we will demonstrate our approach on the two examples illustrated in Fig. 1. The main source of challenge in multi-agent motion planning is the complex interaction between agents in a shared environment. Robotaxi services, now operational in select U.S. cities, showcase advanced autonomous navigation but occasionally face
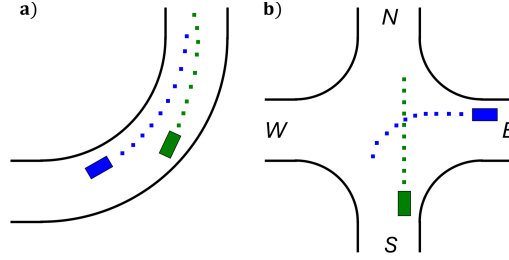
Figure 1: Two-agent interaction scenarios: a) competitive head-to-head racing, and b) cooperative un-signalized two-way intersection navigation, with colored squares showing vehicles' planned trajectories.

the "freezing robot" problem—vehicles freezing to avoid collisions—highlighting challenges in real-world interactions between robots (Bishop, 2024; McIntosh, 2024). Decentralized non-convex algorithms with coupled collision avoidance constraints, as demonstrated in Jankovic (2024), are prone to instability, while purely decentralized strategies without interaction modeling often lead to gridlock. In contrast, interaction-aware decentralized approaches with best-response computations by accounting for other agents' actions effectively avoid these issues.

A common method for interaction-aware motion planning leverages game-theoretic principles to explicitly model multi-agent interactions, accounting for how each agent's actions influence others. As in Zhu and Borrelli (2024); Wang et al. (2018); Williams et al. (2023), the multi-agent motion planning problem is posed as an equilibrium finding problem of a constrained dynamic game, for which the generalized Nash equilibrium (GNE)—a set of strategies where no agent has an incentive to unilaterally deviate from its GNE strategy—is a common solution concept. In Zhu and Borrelli (2024), a novel numerical solver for computing GNE for open-loop dynamic games with state and input constraints is proposed and the solver is demonstrated in autonomous racing examples. However, this method suffers from computational efficiency which limits its use for real-time planning. Similarly, Wang et al. (2018) proposed a game-theoretic formulation for multi-robot racing that approximates the GNE using an iterative best-response method but also faces scalability challenges. The Potential-iLQR approach (Williams et al., 2023) formulates multi-agent interactions as a potential dynamic game (Monderer and Shapley, 1996) and uses distributed optimization to approximate the GNE in a scalable manner. However, the potential game formulation restricts the application of this method to cooperative interactive scenarios and cannot be used to describe games with zero-sum components in their agent objective functions. Furthermore, the game-theoretic motion planning approaches presented in these works require precise knowledge of other agents' dynamics, costs, and decision-making processes (Zhu, 2024), often requiring assumptions about other agents' behavior that may not hold true in real-world scenarios. This limitation motivates our approach to design a feedback policy that uses game-theoretic reasoning as a guiding strategy while adapting to the observed behaviors of other agents by using a learned value function to integrate game-theoretic reasoning into a feedback MPC policy via the terminal cost function.

In this paper, we present an algorithm for learning two-agent motion planning strategies from a dataset of game-theoretic interactions and embedding it in a real-time model predictive control (MPC) scheme to guide the solution of short-horizon optimal control problems. Our approach comprises three main steps:

1. **Game-theoretic Interaction Dataset Generation:** Generate the dataset offline by randomly sampling joint initial conditions and solving constrained dynamic games over agents and storing the GNE solutions.

2. **Reward Outcome Learning:** Train a neural network value function to predict the reward outcomes associated with these interactions. The reward outcome captures the objective of the multi-agent task in a low-dimension.

3. **Implicit Game-Theoretic MPC (IGT-MPC):** Use the learned value function as the terminal cost-to-go function in an MPC policy for online control.

By integrating a learned value function that encodes the strategic outcomes arising from game-theoretic motion planning, GT-MPC enables implicit interaction-aware motion planning without the need for explicit interaction models or computationally intensive game-theoretic equilibrium finding (Zhu and Borrelli, 2024; Wang et al., 2018).

Remark 1 *The current IGT-MPC implementation focuses on two-agent motion planning problems, leaving open the challenges of scaling to larger numbers of agents and generalizing to unseen environments such as varying race track layouts or intersection geometries. While this initial effort does not address these challenges, we acknowledge their importance for enabling practical real-world applications. The primary aim of this work is not to provide an immediate solution for autonomous racing or driving but to demonstrate that game-theoretic strategic outcomes can be effectively encoded in neural networks which can then be integrated into numerical optimization schemes for model-based optimal control, paving the way for research in integration of game-theoretic reasoning and machine learning techniques to be leveraged in numerical optimization schemes for model-based optimal control.*

2. Problem Formulation

Consider a two-agent autonomous vehicle system where the finite-horizon, discrete-time dynamics are described by

$$z_{t+1} = f(z_t, u_t), \tag{1}$$

where $z_t^i \in \mathcal{Z}^i, u_t^i \in \mathcal{U}^i$ are the state and input of agent i at time t and

$$z_t := [z_t^1, z_t^2]^\top \in \mathcal{Z}^1 \times \mathcal{Z}^2 = \mathcal{Z} \subseteq \mathbb{R}^{2n_z}, \tag{2}$$

$$u_t := [u_t^1, u_t^2]^\top \in \mathcal{U}^1 \times \mathcal{U}^2 = \mathcal{U} \subseteq \mathbb{R}^{2n_u}, \tag{3}$$

are the concatenated states and inputs of all agents.

Agent i is subject to state and input constraints defined by $\mathcal{Z}^i = \{z \in \mathbb{R}^{n_z} \mid \underline{z}^i \leq z \leq \overline{z}^i\}$ and $\mathcal{U}^i = \{u \in \mathbb{R}^{n_u} \mid \underline{u}^i \leq u \leq \overline{u}^i\}$, where $\underline{x}$ and $\overline{x}$ are lower and upper bounds on the respective variables which can be derived from the vehicle's actuation limitations, traffic law, or road boundaries. Also, agent i is subject to actuation rate limits denoted by $\Delta\mathcal{U}^i = \{\Delta u \in \mathbb{R}^{n_u} \mid \underline{\Delta u}^i \leq \Delta u \leq \overline{\Delta u}^i\}$, where $\Delta u_k := [u_k^1 - u_{k-1}^1, u_k^2 - u_{k-1}^2]^\top \in \Delta\mathcal{U}^1 \times \Delta\mathcal{U}^2 = \Delta\mathcal{U} \subseteq \mathbb{R}^{2n_u}$.

Each agent i minimizes its own cost function, which is comprised of stage cost $\ell_k^i(z_k, u_k^i)$ and terminal cost $\ell_{N_G}^i(z_{N_G})$ over a finite horizon length of N_G:

$$J^i(\mathbf{z}, \mathbf{u^i}) = \sum_{k=0}^{N_G-1} \ell_k^i(z_k, u_k^i) + \ell_{N_G}^i(z_{N_G}) \tag{4a}$$

$$= J^i(\mathbf{u}^i, \mathbf{u}^{\neg i}, z_0), \tag{4b}$$

where the $\mathbf{u^i} = [u_0^i \ldots, u_{N_G-1}^i]$ denote the input sequences of agent i, and $\mathbf{z} = [z_0, \ldots, z_{N_G}]$ is the sequence of joint states over the prediction horizon N_G. In this work, we use the notation $z_k^{\neg i}$ and $u_k^{\neg i}$ to denote the collection of states and inputs for all but the i-th agent. We derive (4b) by recursively substituting the dynamics (1) into the cost function (4a), which inherently depends on the open-loop input sequences of all agents. The cost is a function of the joint state and input sequence of the i-th agent, capturing dependence on the behavior of other agents. The agents are also subject to additional n_c constraints, $C(\mathbf{u}^1, \mathbf{u}^2, z_0) \leq 0$, which describe the coupling between agents, with its dependence on joint dynamics remaining implicit.

3. Learning Game-theoretic Motion Planning Strategy from GNE

In this section, we describe our approach to learning game-theoretic motion planning strategies from GNE and how to embed it in an MPC policy for motion planning in competitive and cooperative two-autonomous vehicle systems: 1) two-vehicle head-to-head racing on an L-shaped track. 2) two-vehicle navigating through an un-signalized four-way intersection.

3.1. Game-theoretic Interaction Data Generation

First, we define the constrained dynamic game as the tuple:

$$\Gamma = (N_G, \mathcal{Z}, \mathcal{U}, \Delta\mathcal{U}, f, \{J^i\}_{i=1}^2, C). \tag{5}$$

For such a game, a GNE (Facchinei and Kanzow, 2010) is attained when all agents have no incentive to unilaterally change their strategies along feasible directions of the solution space, which means $\mathbf{u}^\star = [\mathbf{u}^{1,\star}, \mathbf{u}^{2,\star}]$ minimizes (4) for both agent 1 and 2. There exists a vast literature on formulating constrained dynamic games for different robotic multi-agent motion planning problems and solving for GNE, such as ALGAMES, iterative Linear Quadratic Regulator (iLQR), Potential-iLQR, and DG-SQP (Le Cleac'h et al., 2022; Fridovich-Keil et al., 2020; Williams et al., 2023; Zhu and Borrelli, 2024). Without loss of generality, we use the DG-SQP solver (Zhu and Borrelli, 2024) to compute a GNE for constrained dynamic games in this work.

We randomly sample joint initial conditions z_0 while other components of the game Γ are fixed and solve for the GNE from that initial condition. Then, the GNE is used to compute the reward outcome of game-theoretic interactions. The reward $R^i \in \mathbb{R}$ quantifies the success in achieving the primary objective of the multi-agent interactive motion planning problem from the perspective of Agent i. While J^i in (4) describes Agent i's cost function and includes stage costs ℓ_k^i, R^i is derived from the terminal costs $\ell_{N_G}^i$, $i = 1, 2$, and thus directly reflects the game outcome.

3.1.1. TWO-VEHICLE HEAD-TO-HEAD RACING

Here the objective is to stay ahead of the opponent. We use the reward

$$R^i = s_{N_G}^i - s_{N_G}^{\neg i}, \quad i = 1, 2, \tag{6}$$

that represents the advantage of vehicle i over vehicle $\neg i$, where $s_{N_G}^i$ and $s_{N_G}^{\neg i}$, are the longitudinal progress of Agent i and Agent $\neg i$ on the race track at time step N_G, respectively. When $i = 1$, we are considering Agent 1 as the ego agent, which means we are predicting the progress advantage that Agent 1 will have over Agent 2. We also construct the reward from the perspective of Agent 2 as the ego agent when $i = 2$. Thus, from a single GNE solution, we extract two data points from the perspective of each agent. Each reward target R is associated with an input feature vector which is the joint initial condition augmented with additional information about the interaction. We note that the input vectors are also problem-dependent, and in the two-vehicle head-to-head racing example, we define

$$\tilde{x}_0^i = [z_0^{\neg i}, z_0^i - z_0^{\neg i}]^\top \in \mathbb{R}^{n_x} \quad i = 1, 2. \tag{7}$$

Here, $z_0^i = [v_0^i, e_{\psi,0}^i, s_0^i, e_{y,0}^i]$, where v_0^i denotes the initial longitudinal velocity, $e_{\psi,0}^i$ is the heading angle error, s_0^i represents the progress along the race line, and $e_{y,0}^i$ is the lateral error of agent i.

3.1.2. COLLABORATIVE NAVIGATION THROUGH AN UNSIGNALIZED INTERSECTION

The main objective of the agents is to collaborate with other agents to ensure all agents pass the intersection safely and promptly. In this example, we propose the reward

$$R^i = \sum_{i=1}^{2} s_{N_G}^i \quad i = 1, 2, \tag{8}$$

which represents the collective progress along corresponding routes at the end of the game horizon N_G. The associated input feature to R^i is

$$\tilde{x}_0^i = [\tilde{z}_0^{\neg i}, \tilde{z}_0^i - \tilde{z}_0^{\neg i}]^\top \in \mathbb{R}^{n_x} \quad i = 1, 2, \tag{9}$$

where $\tilde{z}_0^i = [s_0^i, v_0^i, sc_i]$. Assuming all agents don't deviate from the centerline of the road, we only include the longitudinal states and scenario encoding sc_i, which is a categorical variable, of the two-vehicle intersection to distinguish what type of interaction is taking place in the perspective of agent i. Without it, data points from different interactions (i.e. left turn & left turn vs. left turn & straight) become indistinguishable. For more details about scenario encoding and its implementation, please refer to (Kim et al., 2024, Sec 6). Similarly, we construct the features and reward targets from the perspective of both agents. Note that the reward target for each agent is equal, which reflects the cooperative setting where all agents share the reward. The computed reward $R = \{R^i\}_{i=1}^2$ and feature vector $\tilde{x}_0 = \{\tilde{x}_0^i\}_{i=1}^2$ are added to the dataset as follows

$$\begin{aligned} X &= X \cup \{\tilde{x}_0\} \\ Y &= Y \cup \{R\}, \end{aligned} \tag{10}$$

Remark 2 *Capturing interaction between agents in the generated dataset is key for a successful implementation of the proposed method. In practice, the distribution of training data often differs from the distribution seen during IGT-MPC deployment, a phenomenon known as distribution shift. As covering the entire feature space by exhaustively sampling data points is intractable, a more data-efficient approach leverages strategic initial condition sampling based on any prior knowledge of the deployment data distribution. For example, in a head-to-head racing scenario, when two agents are far enough apart, then there isn't any significant coupling between their decision-making. Thus, sampling from the set of initial conditions where an interaction between two agents is possible will be sufficient in capturing the interactive behavior.*

Remark 3 *When sampling initial conditions, the game solver may occasionally fail to compute the GNE due to infeasibility from collision avoidance constraints under the joint dynamics. Storing these infeasible points in the dataset and associating them with low reward outcomes is crucial, as they represent undesirable states leading to collisions that the model must learn to avoid.*

3.2. Learning the Reward Outcome

After generating the dataset $\mathcal{D} = (X, Y)$, we obtain the game-theoretic value function $V_{GT} : \mathbb{R}^{n_x} \mapsto \mathbb{R}$ that estimates the reward outcome R^i of the game-theoretic interactions between agents at time step N_G from the input features $\tilde{x}_0^i$. We use the following simple neural network architecture with $tanh$ activation function:

$$y_l = tanh(W_l y_{l-1} + b_l), \quad l = 1, ..., L, \tag{11}$$

where W_l, b_l are the parameters of the l-th layer of the network with hidden state dimension h. Additionally, $y_0 = \tilde{x}^i$ and $y_L = \hat{R}^i$, and $tanh$ is a non-linear, differentiable activation function.

The regression model is trained on $\mathcal{D}$ using the mean-squared error loss function defined as $\mathcal{L} = \frac{1}{n_b} \sum_{k=1}^{n_b} \sum_{i=1}^{2} (R_k^i - \hat{R}^i{}_k)^2$ where n_b is the training batch size. We employ the Adam optimizer (Kingma and Ba, 2014) with early stopping to prevent overfitting. Also, the features and target values in $\mathcal{D}$ are normalized during training. From this point forward, the trained neural network value function on a game-theoretic interaction dataset is referred to as V_{GT}. Notably, this relatively simple neural network architecture effectively captures game-theoretic strategies in the GNE for two-agent motion planning in both competitive and cooperative settings, as demonstrated in Sec. 4.

3.3. Implicit Game-theoretic MPC Policy

After training V_{GT} that predicts the game-theoretic reward outcome N_G steps into the future given the input features, we use this predictor as a terminal cost-to-go function for a real-time MPC policy that implicitly emulates the game-theoretic strategies encoded in GNE.

3.3.1. COLLISION AVOIDANCE

While methods like optimization-based collision avoidance constraints (Zhang et al., 2021), which model agents as polytopes, can be used, we opt to represent obstacles as circles and impose distance constraints, maintaining simplicity without compromising effectiveness. Inter-agent collision avoidance constraints are given as $\mathbb{CA} = \{(p_1, p_2) \mid \|p_1 - p_2\|_2 \geq d_{min}\}$, where p_1 and p_2 are global Cartesian position vectors, and d_{min} is the user-defined parameter for minimum allowable distance between any two agents. We denote $P(z^i, z^j)$ as the projection of the state vector pair to Cartesian position vector space. Two agents i and j are considered to be collision-free if and only if $P(z^i, z^j) \in \mathbb{CA}$.

3.3.2. MPC FORMULATION

The MPC policy is a *predict-then-plan* scheme that computes an optimal ego trajectory w.r.t. the open-loop predictions of the environment. We assume the forecasts are provided by a trajectory prediction scheme. In the numerical examples, we generate $\hat{\mathbf{z}}^{\neg i}$ using an oracle that provides the optimal planned trajectories of all agents. For competitive racing scenarios, we corrupt it with Gaussian noise (Zhu, 2024).

Each agent i computes a state-feedback control $u_t^i = \pi_{\text{MPC}}(z_t^i, \hat{\mathbf{z}}^{\neg i}, V)$ by solving the following finite-horizon optimal control problem (FHOCP) towards tracking a state reference $\{z_{r,k}\}_{k=t}^{t+N}$ given forecasts of other agents $\hat{\mathbf{z}}^{\neg i} = \{\hat{z}_{k|t}^{\neg i}\}_{k=t}^{t+N}$ over the N step MPC prediction horizon,

$$\min_{\boldsymbol{u}^i} \left[\sum_{k=t}^{t+N-1} \|z_{k|t}^i - z_{r,k}\|_Q^2 + \|u_{k|t}^i\|_{R_1}^2 + \|\Delta u_{k|t}^i\|_{R_2}^2 \right] + V(\tilde{x}_{t+N}^i) \tag{12a}$$

$$\text{s.t.} \quad z_{k+1|t}^i = f^i(z_{k|t}^i, u_{k|t}^i), \quad \forall k \in \mathbb{I}_t^{t+N-1} \tag{12b}$$

$$z_{k|t}^i \in \mathcal{Z}^i, \quad \forall k \in \mathbb{I}_t^{t+N}, \ u_{k|t}^i \in \mathcal{U}^i, \ \Delta u_{k|t}^i \in \Delta \mathcal{U}^i, \quad \forall k \in \mathbb{I}_t^{t+N-1} \tag{12c}$$

$$P(z_{k|t}^i, \hat{z}_{k|t}^j) \in \mathbb{CA}, \quad \forall j \in \{\neg i\}, \ z_{t|t}^i = z^i(t), \tag{12d}$$

where $z^i(t)$ is the measured state of the i-th agent at time t, and $z_{k|t}^i$ indicates the state z^i at k-th step predicted at time t. f^i is the discrete-time dynamics of agent i, and $\mathbb{I}_{k_1}^{k_2}$ denote the index set $\{k_1, k_1+1, \ldots, k_2\}$. z_r is a reference state trajectory, and $Q \succeq 0$ and $R_1, R_2 \succ 0$ are weighting matrices for the state reference tracking, input, and input rate costs, respectively. The reference state trajectory is the pre-computed raceline or the centerline of a road. The notation $\|x\|_M^2 = x^\top M x$, where M is a matrix with appropriate dimensions.

The cost function (12a) includes a stage cost term $\ell_k^i(\cdot)$ and a terminal cost term $V(\cdot)$. It is important to note that the input to the terminal cost function $\tilde{x}_{t+N}^i$ includes the joint state $z_{t+N|t} = [z_{t+N|t}^i, \hat{z}_{t+N|t}^{\neg i}]$ where $z_{t+N|t}^i$ is a decision variable of the optimization problem. Therefore, if $V = -V_{GT}$, minimizing the learned value function over $z_{t+N|t}^i$ can be interpreted as agent i selecting its terminal state given the forecasts of other agents, $\hat{z}_{t+N|t}^{\neg i}$, to maximize its reward (i.e. maximize advantage over the opponent). The learned value function, V_{GT}, is used in closed-loop with the MPC policy as formalized in Algorithm 1. The FHOCP (12) is constructed using CasADi (Andersson et al., 2018) where V_{GT} is constructed symbolically using the same architecture and trained weights from the learned model in PyTorch (Paszke et al., 2017).

4. Numerical Examples

We demonstrate Algorithm 1 in two-vehicle competitive and cooperative multi-agent settings. The code for obtaining the presented results can be found in the GitHub repository (Kim, 2024a). We use $N = 10$ and a sampling time of $0.1\ s$ for all MPC policies in our numerical examples.

Algorithm 1 Implicit Game-Theoretic MPC (IGT-MPC)

Require: V_{GT}
  Set the task horizon T
  $t \leftarrow 0$
  while $t < T$ **do**
    $\hat{\mathbf{z}}^{\neg i} \leftarrow$ Forecasts of other agents
    $\tilde{x}_{t+N} \leftarrow \hat{\mathbf{z}}^{\neg i}$
    $V \leftarrow V_{GT}$
    $u_t^i \leftarrow \pi_{\text{MPC}}(z^i(t), \hat{\mathbf{z}}^{\neg i}, V)$ $\triangleright$ See Eq. (12)
    Apply u_t^i to system
    $t \leftarrow t + 1$
      end

4.1. Two-vehicle Head-to-Head Racing

In the two-vehicle head-to-head race example, we consider the Frenet-frame dynamic bicycle model given a precomputed race line trajectory for an L-shaped race track (Zhu and Borrelli, 2024). We formulate the autonomous racing problem as a constrained dynamic game and use DG-SQP solver to compute the GNE for such game. For details of formulating the game and solving for GNE, refer to (Zhu, 2024, Chapter 7). We collected approximately 1800 samples with $N_G = 25$, and trained V_{GT} with $h = 48, L = 2$ using PyTorch. To study the impact of the V_{GT}, we consider two different terminal cost-to-go functions,

$$V = \begin{cases} -V_{GT}(z^i, \hat{z}^{\neg i}), \\ V_{MP}(z^i, \hat{z}^{\neg i}) = -(s^i - \hat{s}^{\neg i}), & i = 1, 2, \end{cases} \tag{13}$$

where V_{MP} is a naive maximum progress function based on the opponent's prediction to encourage progress maximization with respect to the opponent but does not have any game-theoretic component. Fig. 2 shows
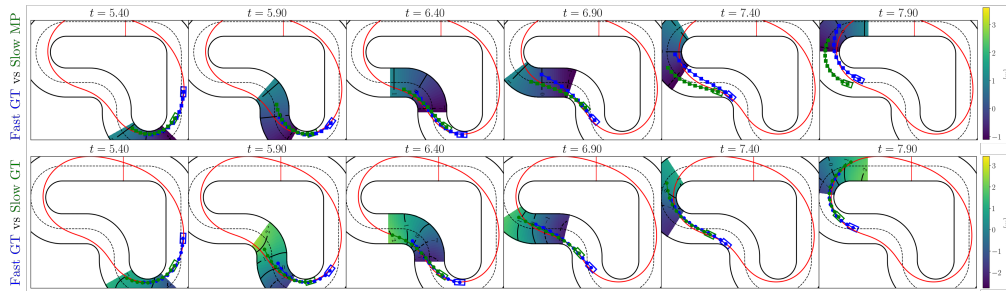

Figure 2: In a simulation experiment, the slower (green) vehicle is unable to defend its position against the faster (blue) vehicle with V_{MP} (Top). The slower vehicle with V_{GT} successfully defends its position against the faster car with V_{GT} (Bottom). The heatmap represents the level curves to visualize the value functions used in this experiment. The red curve is the raceline and the colored squares and circles are the planned trajectories for vehicles with corresponding colors.

snapshots at identical time instances of two simulated races starting from the same initial condition but different configurations to demonstrate a position defense capability of a slower vehicle when guided by V_{GT}. Race configurations are categorized by two settings: *fast* or *slow* and *GT* or *MP* value functions, where *GT* or *MP* denotes the terminal cost-to-go function used by the MPC policy. While both vehicles track pre-computed racelines of the same shape, we force the initially leading green vehicle to be slower than the initially trailing blue vehicle by reducing its velocity profile by 10%. For details on setting up the race, refer to
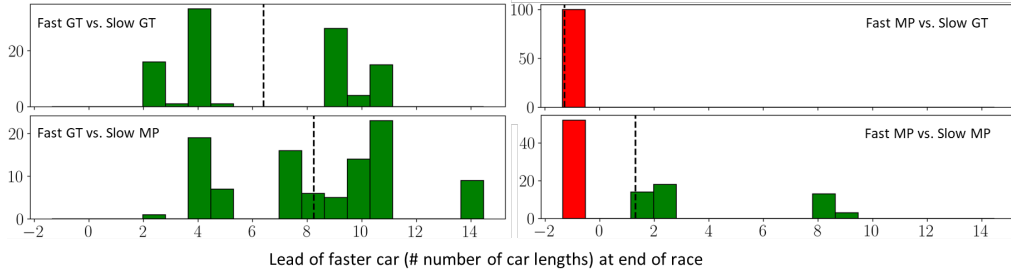
Figure 3: Histograms of the lead (in number of car lengths) of the faster vehicle over the slower vehicle at the end of each simulated race over 100 different initial conditions. Green bars indicate bins where the faster vehicle won the race, while red bars indicate losses. The black dashed lines represent average lead values over all simulation runs. Note that vertical scales vary between histograms.

(Zhu, 2024, Chapter 7). The level curves of V_{MP} (Top) and V_{GT} (Bottom) for the green vehicle are projected onto the track in Fig. 2. While the level curves of V_{MP} are constant over lateral positions, indicating no preference for where on the track the vehicle should end up at the end of the horizon, V_{GT} exhibits more intricate level curves, showing a strategic preference for lateral positions near the pre-computed race line, although the learned value functions had no prior knowledge of it (Zhu, 2024). By $t = 7.90\ s$, the green vehicle guided by V_{MP} is overtaken by the faster opponent, while under V_{GT}, it successfully defends its leading position. More simulation results can be found in the video (Kim, 2024b).

We report Monte Carlo simulation results over 100 initial conditions for various race configurations. The histograms in Fig. 3 show the lead of the faster car, measured in car lengths, at the end of each race. Black dashed lines indicate the means. In the top row, the *Fast GT* vehicle consistently wins, averaging a 6.5 car-length lead over the *Slow GT*, while the *Fast MP* trails the *Slow GT* by 1.4 car lengths. Notably, *Fast MP* loses 100% of the races against the slower but more strategic *Slow GT*. In the bottom row, the faster vehicle races against *Slow MP*. Guided by V_{GT}, it wins all races, averaging 8.1 car lengths of lead. With V_{MP}, the win rate drops to 50%. Overall, V_{GT} outperforms V_{MP} in both overtaking and defending, as seen in the *Fast GT vs Slow MP* and *Fast MP vs Slow GT* cases.

4.2. Two-vehicle Intersection Navigation

In the two-vehicle un-signalized intersection example, we consider the Frenet-frame kinematic bicycle model (Kim and Borrelli, 2023) augmented with global Cartesian coordinates (x, y) for imposing the collision avoidance constraints. For simplicity, we consider homogeneous agents under identical constraint sets in this example. We collected approximately 2000 samples per scenario with $N_G = 200$ and sampling time of $0.1\ s$, and trained V_{GT} with $h = 128, L = 2$ using PyTorch. The implementation details for obtaining the GNE solutions and constructing the feature vectors are detailed in (Kim et al., 2024, Sec 6).

Again, to study the impact of the V_{GT}, we consider two different terminal cost-to-go functions,

$$V = \begin{cases} -V_{GT}(\tilde{z}^i, \hat{\tilde{z}}^{\neg i}), \\ V_{MP}(z^i, \hat{z}^{\neg i}) = -(s^i + \hat{s}^{\neg i}), \quad i = 1, 2, \end{cases} \tag{14}$$

where V_{MP} is a naive maximum progress function that maximizes the progress of agent i without considering the interaction with other agents. The optimization problem (12) is formulated in CasADi and solved using IPOPT. In the cooperative two-vehicle un-signalized intersection example, we assume the vehicles are connected and communicate their route information and planned optimal trajectories, $\hat{\mathbf{z}}^i = \{z_{k|t}^{i,\star}\}_{k=t}^{t+N}$ without delay, allowing $\tilde{x}_{t+N}^i$ to be computed exactly at time t.

We highlight the differences in how V_{GT} and V_{MP} guide the short-horizon MPC solutions through an intersection from the same initial condition. In Fig. 4, we show Vehicles 1 (green) on route *SN* and Vehicle 2 (blue) on *ES* navigating through an un-signalized intersection starting from rest, where both vehicles are controlled using identical MPC policies (12) with the same V_{GT}. However, the input features to V_{GT} are constructed from each vehicle's perspective by concatenating their respective features, as specified in (9).
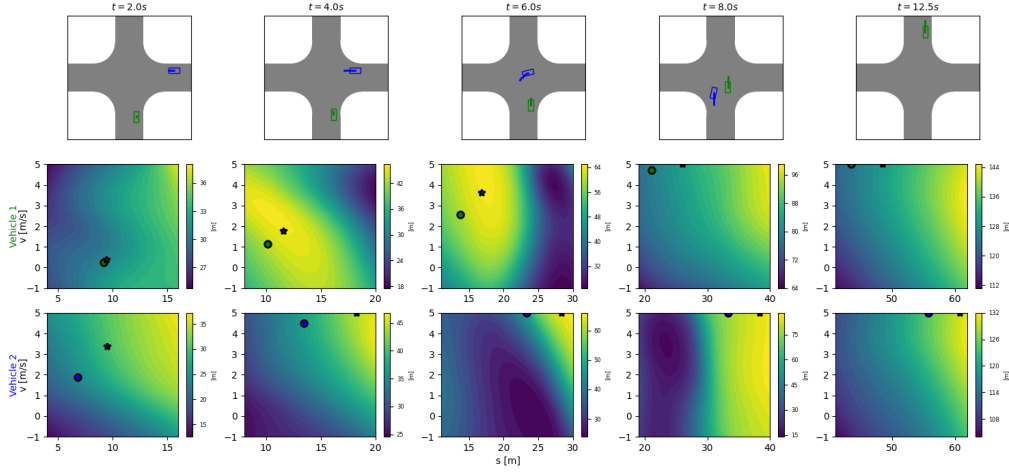
Figure 4: In a simulation experiment, Vehicle 1 (green) on route *SN* and Vehicle 2 (blue) on route *ES* navigate through an intersection using V_{GT} (Top). Colored squares represent each vehicle's planned trajectories. The middle and bottom rows display the contour of V_{GT} as perceived by Vehicle 1 and 2, respectively. Colored circles indicate current states at each time instance, and colored stars denote the planned terminal state at time $t + N$.

Although both vehicles share the same policy and value function, their unique perspectives lead to different inputs and outputs. This enables each to follow the collaborative strategy while exhibiting distinct behaviors tailored to their individual contexts. For instance, at $t = 4$ s, as both vehicles approach the intersection, the level curves of V_{GT} for Vehicle 1 show a preference for the terminal states in $s_{t+N}^1 \in [10, 13]$ m and $v_{t+N}^1 \in [2, 3]$ m/s, whereas Vehicle 2's level curves indicate a preference for terminal states in $s_{t+N}^2 \in [17, 20]$ m and $v_{t+N}^2 \in [4, 5]$ m/s. This illustrates how V_{GT} is guiding the MPC planner for Vehicle 1 to accelerate slowly and yield, while V_{GT} is guiding the Vehicle 2 to accelerate quickly and pass the intersection first. At $t = 6$ s, when Vehicle 2 reaches the middle of the intersection at its maximum speed of 5 m/s, the level curves of V_{GT} for Vehicle 1 show a preference for higher speed and progress, guiding Vehicle 1 to start accelerating. After Vehicle 2 completes its left turn and is no longer in the path of Vehicle 1, V_{GT} guides Vehicle 1 to accelerate to maximum speed, allowing both vehicles to pass the intersection by $t = 12.5$ s.

In Fig. 5, when both MPC planners are guided by V_{MP}, they prioritize maximizing progress without accounting for their actions' impact on other agents, as shown by the linear level sets in the middle and bottom rows. This leads to planner infeasibility due to unavoidable collisions from inertia, triggering the safety controller to apply maximum braking at $t = 6$ s. As a result, the vehicles become stationary and gridlocked, unable to proceed due to coupled collision avoidance constraints. While extending the prediction horizon could mitigate this issue, it increases computational complexity. Even so, the naive *predict-then-plan* approach can still encounter infeasibility and gridlock in this intersection scenario, as demonstrated in the video in Kim (2024b). Furthermore, simulation experiment results demonstrating the effectiveness of V_{GT} in emulating coordinated behaviors in different scenarios are reported in (Kim et al., 2024, Sec 6).

For each scenario $m \in \mathbb{I}_1^8$ as defined in (Kim et al., 2024, Sec 6) route combinations are sampled from each scenario set to obtain the corresponding precomputed reference trajectory, z_r, for the route. Initial $s_0^1, s_0^1 \sim U(0, s_{int}/2)$ are sampled, and $v_0^1, v_0^2 = 0$ m/s, where s_{int} denotes the longitudinal displacement at the intersection. Closed-loop simulations are then evaluated with both vehicles using identical terminal cost-to-go functions in (12) over a task horizon of $T = 150$. The value functions, V_{MP} and V_{GT}, are evaluated over 100 initial conditions for each scenario. Performance metrics include % of feasibility, % of gridlock and the average solve time of (12). The feasibility percentage measures the proportion of simulation runs in which MPC planners remain feasible throughout the simulation. High feasibility % signifies that the terminal cost-to-go function effectively guided vehicles to navigate collaboratively, avoiding safety controller intervention. Conversely, a lack of clear passing order or collaborative strategy can cause vehicles to enter the
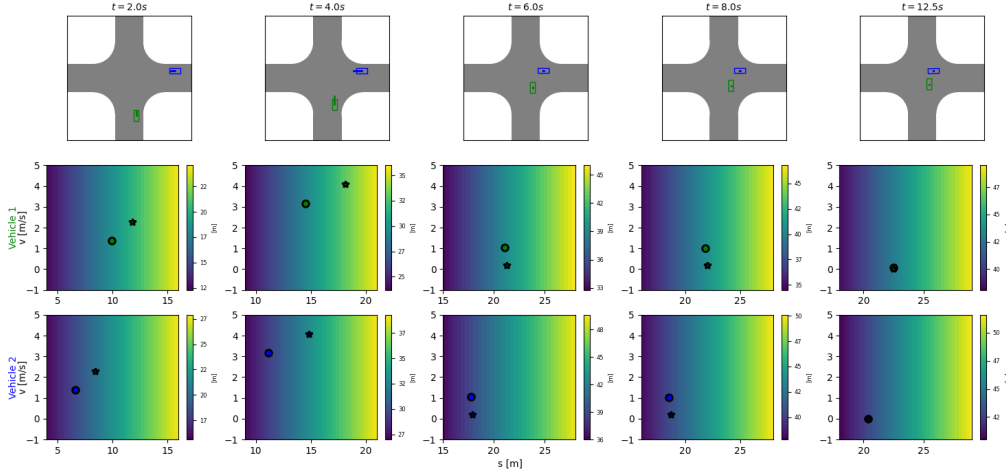
Figure 5: In a simulation experiment, Vehicle 1 (green) on route *SN* and Vehicle 2 (blue) on route *ES* navigate through an intersection using V_{MP} (Top), reaching a gridlock. Colored squares represent each vehicle's planned trajectories. The middle and bottom rows display the contour of V_{MP} as perceived by Vehicle 1 and 2, respectively. Colored circles indicate current states at each time instance, and colored stars denote the planned terminal state at time $t + N$. The initial conditions are identical to that of in Fig. 4

intersection simultaneously, resulting in gridlock. The numerical results in Table 1 highlight the effectiveness of the proposed approach. Guided by V_{GT}, MPC planners achieve at least 97% feasibility across all scenarios, consistently outperforming V_{MP}. Notably, in scenarios 6 and 7, where the naive maximum-progress terminal cost V_{MP} fails completely, V_{GT} achieves 99% feasibility with no gridlocks. Vehicles guided by V_{GT} avoid gridlock by adhering to the passing order encoded in the value function, enabling smooth collaborative navigation. In contrast, V_{MP} often results in gridlocks due to its lack of interaction modeling. While V_{GT} introduces higher computation time, alternative NLP solvers could improve efficiency.

Table 1: Metrics for Two-Vehicle Intersection Navigation Monte Carlo Simulations

Scenario	Feasibility (%)		Gridlock (%)		Avg. Solve Time (s)	
	V_{MP}	V_{GT}	V_{MP}	V_{GT}	V_{MP}	V_{GT}
1	25	**99**	6	**0**	**0.045**	0.061
2	1	**97**	80	**0**	**0.055**	0.089
3	37	**100**	22	**0**	**0.073**	0.10
4	95	**100**	0	**0**	**0.087**	0.091
5	93	**99**	1	**0**	**0.075**	0.078
6	0	**99**	43	**0**	**0.082**	0.10
7	0	**99**	100	**0**	**0.077**	0.098
8	1	**98**	57	**0**	**0.083**	0.084

5. Conclusion

In this work, we presented IGT-MPC, an algorithm that enables decentralized, game-theoretic multi-agent motion planning in competitive and cooperative settings without requiring explicit game-theoretic modeling, which can be computationally expensive. We demonstrated IGT-MPC through two multi-agent examples: a head-to-head race and two-vehicle navigation at an unsignalized intersection. Results highlight the learned value function's effectiveness in guiding MPC to replicate game-theoretic interactions, achieving competitive or coordinated behaviors. Scaling to larger M and diverse environments requires a richer dataset capturing game-theoretic interactions, which becomes increasingly expensive to generate. As interactions grow more complex with additional agents, more sophisticated neural networks are needed to capture data patterns (e.g., multi-agent interactions and environmental context). However, this adds computational cost to gradient evaluations, increasing solve time. Balancing model complexity and efficiency remains a key challenge for future work.

References

Joel A E Andersson, Joris Gillis, Greg Horn, James B Rawlings, and Moritz Diehl. CasADi – A software framework for nonlinear optimization and optimal control. *Mathematical Programming Computation*, 2018.

Richard Bishop. 60 million miles and counting: Robotaxis shift into high gear, 2024. URL https://www.forbes.com/sites/richardbishop1/2024/07/27/60-million-miles-and-counting-robotaxis-shift-into-high-gear/.

Francisco Facchinei and Christian Kanzow. Generalized nash equilibrium problems. *Annals of Operations Research*, 175(1):177–211, Mar 2010. ISSN 1572-9338. doi: 10.1007/s10479-009-0653-x. URL https://doi.org/10.1007/s10479-009-0653-x.

David Fridovich-Keil, Ellis Ratner, Lasse Peters, Anca D. Dragan, and Claire 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)*, pages 1475–1481, 2020. doi: 10.1109/ICRA40945.2020.9197129.

Mrdjan Jankovic. Instinctive negotiation by autonomous agents in dense, unstructured traffic: A controls perspective. *Annual Review of Control, Robotics, and Autonomous Systems*, 7(Volume 7, 2024):97–121, 2024. ISSN 2573-5144. doi: https://doi.org/10.1146/annurev-control-060923-025701. URL https://www.annualreviews.org/content/journals/10.1146/annurev-control-060923-025701.

Hansung Kim. Github: Implicit-game-theoretic-mpc, 2024a. URL https://github.com/MPC-Berkeley/Implicit-Game-Theoretic-MPC.

Hansung Kim. Youtube: Learning two-agent motion planning strategies from generalized nash equilibrium for mpc, Nov. 2024b. URL https://youtu.be/9jlz95Nor2I.

Hansung Kim and Francesco Borrelli. Facilitating cooperative and distributed multi-vehicle lane change maneuvers. *IFAC-PapersOnLine*, 56(2), 2023. 22nd IFAC World Congress.

Hansung Kim, Edward L. Zhu, Chang Seok Lim, and Francesco Borrelli. Learning two-agent motion planning strategies from generalized nash equilibrium for model predictive control, 2024. URL https://arxiv.org/abs/2411.13983.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2014. URL https://api.semanticscholar.org/CorpusID:6628106.

Simon Le Cleac'h, Mac Schwager, and Zachary Manchester. Algames: a fast augmented lagrangian solver for constrained dynamic games. *Auton. Robots*, 46(1):201–215, January 2022. ISSN 0929-5593. doi: 10.1007/s10514-021-10024-7. URL https://doi.org/10.1007/s10514-021-10024-7.

Jil McIntosh. Four waymo cars cause driverless stand-off in san fran, 2024. URL https://driving.ca/auto-news/crashes/waymo-cars-confused-stand-off-san-francisco.

Dov Monderer and Lloyd S. Shapley. Potential games. *Games and Economic Behavior*, 14(1):124–143, 1996. ISSN 0899-8256. doi: https://doi.org/10.1006/game.1996.0044. URL https://www.sciencedirect.com/science/article/pii/S0899825696900445.

Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. 2017.

Zijian Wang, Riccardo Spica, and Mac Schwager. Game theoretic motion planning for multi-robot racing. In *International Symposium on Distributed Autonomous Robotic Systems*, 2018. URL https://api.semanticscholar.org/CorpusID:68244069.

Zach Williams, Jushan Chen, and Negar Mehr. Distributed potential ilqr: Scalable game-theoretic trajectory planning for multi-agent interactions. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 01–07, 2023. doi: 10.1109/ICRA48891.2023.10161176.

Xiaojing Zhang, Alexander Liniger, and Francesco Borrelli. Optimization-based collision avoidance. *IEEE Transactions on Control Systems Technology*, 29(3):972–983, 2021. doi: 10.1109/TCST.2019.2949540.

Edward L. Zhu. *From Cooperation to Competition: Prediction and Planning in Constrained Multi-Agent Settings Using Data-Driven and Model-Based Optimal Control Methods*. PhD thesis, University of California, Berkeley, 2024. URL https://www.proquest.com/dissertations-theses/cooperation-competition-prediction-planning/docview/3111087006/se-2. Copyright - Database copyright ProQuest LLC.

Edward L. Zhu and Francesco Borrelli. A sequential quadratic programming approach to the solution of open-loop generalized nash equilibria for autonomous racing, 2024. URL https://arxiv.org/abs/2404.00186.