2018 IEEE International Conference on Robotics and Automation (ICRA)
May 21-25, 2018, Brisbane, Australia

Best Response Model Predictive Control for Agile Interactions Between Autonomous Ground Vehicles

Grady Williams, Brian Goldfain, Paul Drews, James M. Rehg, and Evangelos A. Theodorou

***Abstract*— We introduce an algorithm for autonomous control of multiple fast ground vehicles operating in close proximity to each other. The algorithm is based on a combination of the game theoretic notion of iterated best response, and an information theoretic model predictive control algorithm designed for non-linear stochastic systems. We test the algorithm on two one-fifth scale AutoRally platforms traveling at speeds upwards of 8 meters per second, while maintaining a following distance of under two meters from bumper-to-bumper.**

I. INTRODUCTION

Autonomous vehicles operating in the real world have the potential to improve transportation safety and efficiency, and convert human time currently spent on driving into productive work or leisure time [1]. To achieve these goals, autonomous vehicles must be able to interact with other vehicles, which requires the ability to anticipate and react to both human drivers and other autonomous vehicles operating nearby. This is not only a safety requirement, it is also a key benefit, as the ability to operate in tighter spaces than is currently possible could significantly increase highway capacity [2]. In order operate effectively in tight quarters, these systems must have rapid reaction times, and must be able to perform agile maneuvers in order to safely navigate through crowded environments.

A basic issue for vehicle planning and control in crowded environments is whether vehicles explicitly communicate and coordinate with each other. The benefit of explicit coordination [3]–[5] is that it yields a well-defined planning and control problem, since if the vehicles all communicate their internal plan or agree to a certain set of rules, then the evolution of the system can be analyzed using standard tools from control theory and optimization. However, communicating internal plans between vehicles is potentially problematic in that the encrypted communication link becomes a safety-critical component of the system, which if compromised could be catastrophic. Additionally, communicating internal plans at a high frequency requires high bandwidth, which may not always be available. The alternative is to develop methods which do not require explicit coordination [6]–[8]. It is highly desirable to have a non-communicative system which is capable of multi-vehicle interaction in close quarters, as either a primary or back-up control method.

A problem with multi-vehicle interaction in a non-communicative setting is that the robot must be able to "guess" at the other vehicles' current plans, and predict how the other vehicles might react to changes it makes in its own policy. The usual methods for predicting the behavior of other vehicles are to treat them as obstacles moving at a constant speed subject to stochastic disturbances [9], [10], or to use predefined behavior models [11], [12]. Although these methods can be effective, they both have significant drawbacks: treating other intelligent vehicles as unintelligent obstacles is inaccurate in many cases, and predefined behavior models are computationally expensive and require careful data collection and analysis to train [13].

Fig. 1. Two AutoRally vehicles operating on the dirt test track at the Georgia Tech Autonomous Racing Facility.

A third approach, which we propose here, is to formulate the problem as a differential game, and then use optimization to approximately solve the problem. The game-theoretic approach treats each vehicle as an independent agent attempting to optimize an objective, subject to their internal dynamic constraints, and it can be viewed as the generalization of optimal control to multiple, potentially competing, agents. This approach can generate realistic predictions of the behavior of other vehicles, while only requiring knowledge of the other vehicles' dynamics, objectives, and current state. This is advantageous because, in many cases, it is fundamentally easier to infer intent than to predict behavior. For example, during highway driving, activating a turn signal almost always means that a vehicle would like to change lanes. However, the precise behavior that results from that intent is dependent on the density of traffic, geometry of the road, etc. In the game theoretic approach the result of this interaction between intent and environment can be determined immediately by the optimization, whereas in a behavioral model approach it would be necessary to collect data and verify the model for each environmental sub-case.

The drawback of the game-theoretic approach is that the

The authors are with Institute for Robotics and Intelligent Machines at the Georgia Institute of Technology, Atlanta, GA, USA. Email: gradyrw@gatech.edu

978-1-5386-3081-5/18/$31.00 ©2018 IEEE

resulting optimization problem is very difficult to solve. For example, it is not practical to develop even a general numerical scheme which converges to a solution (a Nash Equilibrium) in the case of non-linear stochastic differential games. Instead, we propose to use a simple numerical method, known as *best-response iteration*, which is an iterative system of equations with the solutions to the differential game as fixed points. We then combine best-response iteration with information theoretic model predictive control, and we demonstrate the ability of the resulting best response model predictive control algorithm to independently control two one-fifth scale autonomous ground vehicles operating in close proximity. As a baseline for comparison, we utilize the method of treating the other vehicle as an obstacle moving in a straight line with constant velocity, and compare its performance to that of the best response controller.

II. RELATED WORK

The application of the theory of differential games for min-max problems is popular in several areas of robotics: single player games where system noise is considered adversarial [14], pursuit evasion games [15], [16], and cooperative games where control of a robot is shared [17]. Although the methods described in these works are effective in their own domains, we consider a more general problem formulation which does not fall under the umbrella of min-max differential games. This inspires our usage of best-response iteration, which is one of the oldest methods in game theory [18], and can be applied to any game. Although the idea of best response iteration is old, its use in the control of autonomous robotic systems is novel, and it is only possible because of recent advances in computing power and fast online optimization schemes.

The most similar framework to the one we present here is [8], where a best-response type iteration is used with deterministic model predictive control in order to study human-robot behavior in a variety of simulated driving scenarios with simplified vehicle dynamics. The main algorithmic difference between [8] and our work is the formulation of the problem as a semi-stochastic game, and the subsequent use of stochastic optimal control to compute a best-response. Taking stochasticity in the dynamics into account during the optimization enables the method to automatically determine a safe operating margin between itself and the other vehicles in the system, which results in a high level of performance despite the presence of significant model uncertainty and external disturbances.

The stochastic model predictive control method that we use as the basis for our best-response mechanism is based on our previous work on information theoretic model predictive control [19]. This work has its origins in path integral control theory [20], and is also closely related to the cross-entropy method for motion planning [21].

III. PROBLEM SETUP

Our primary motivation is the problem of two autonomous ground vehicles operating in close proximity. However, we begin by describing the differential[1] game problem formulation for a general two-player system (the generalization to more than two is straight-forward). Let F_a and F_b be the dynamics of player A and player B respectively. We assume that the equations of motions for the players are stochastic, discrete time equations of the form:

$$\mathbf{x}_i^{t+1} = \mathbf{f}_i(\mathbf{x}_i^t, \mathbf{v}_i^t) \tag{1}$$

$$\mathbf{v}_i^{t+1} \sim \mathcal{N}(\mathbf{u}_i^t, \Sigma_i) \tag{2}$$

$$i \in \{a, b\} \tag{3}$$

where $\mathbf{u}_i^t$ is the commanded input for player i at time t, and $\mathbf{v}_i^t$ is the input perturbed by a Gaussian disturbance. This type of control-dependent noise is a common assumption in robotics control. Next $\mathbf{x}_a^0 \in \mathbb{R}^{n_a}$ and $\mathbf{x}_b^0 \in \mathbb{R}^{n_b}$ denotes the initial conditions of the two players, $\mathcal{U}_a$ and $\mathcal{U}_b$ denotes the set of admissible control inputs for the systems, and C_a and C_b are the cost functions for the two players. It is assumed that both cost functions take the form:

$$C_i = \mathbb{E}_{F_a,F_b}\left[\phi_i(\mathbf{x}_a^T, \mathbf{x}_b^T) + \sum_{t=0}^{T-1} \mathcal{L}_i(\mathbf{x}_a^t, \mathbf{x}_b^t, \mathbf{u}_a^t, \mathbf{u}_b^t)\right] \tag{4}$$

$$\mathcal{L}_i = c_i(\mathbf{x}_a^t, \mathbf{x}_b^t) + \lambda(\mathbf{u}_i^t)^{\mathrm{T}}\Sigma_i^{-1}\mathbf{u}_i^t \tag{5}$$

where $t \in \{0, 1, \dots T\}$ and $i \in \{a, b\}$. Note that although both players cost functions have the same general structure, the two cost functions need *not* be the same. The resulting differential game can then be described by the tuple:

$$\mathcal{G} = \{F_a, F_b, \mathbf{x}_a^0, \mathbf{x}_b^0, C_a, C_b, \mathcal{U}_a, \mathcal{U}_b\} \tag{6}$$

Each players' objective is to minimize their cost function subject to their own dynamical constraints. However, since the costs are functions of both players, and each players' costs may not be aligned, it may be impossible to find a set of control which minimizes the cost functions for both players simultaneously. This creates the need for an alternative solution concept to functional minimization. The most appropriate notion of solution for our setting is the Nash equilibrium. Let $\mathcal{X}$ denote the concatenated states of both players $\mathcal{X}_t = (\mathbf{x}_a^t \ \mathbf{x}_b^t)^{\mathrm{T}}$, and let $\pi_a(\mathcal{X}_t)$ and $\pi_b(\mathcal{X}_t)$ denote policies for players A and B respectively, we then have the following definition:

Definition 1: A set of policies $\pi_a(\mathcal{X}_t)$ and $\pi_b(\mathcal{X}_t)$ are said to be in a Nash Equilibrium, for the game $\mathcal{G}$, if:

$$\forall i \in \{a, b\},\ C_i(\pi_a, \pi_b) = \min_{\pi}\left[C_i\left(\pi, \pi^{(a,b)\setminus i}\right)\right]$$

That is, each players policy is optimal given that the policy of the other player is fixed.

If the players' policies are in Nash equilibrium then neither player has an incentive to unilaterally change their policy, thus the Nash equilibrium acts as a natural solution concept for multi-player games. Our goal will therefore be to find a Nash equilibrium for our game. In this work, we use open-loop control laws $\pi_i = \{\mathbf{u}_i^0, \mathbf{u}_i^1, \dots \mathbf{u}_i^{T-1}\}$ as the policy parameterization, although it is theoretically possible to work with more powerful policy parameterizations.

[1]Technically a difference game, since we will consider discrete time dynamics.

A. Semi-Stochastic Game

In our problem formulation, we assumed that both of the systems under consideration have stochastic dynamics. This is a realistic assumption, however it creates a very difficult objective function from an optimization standpoint. When both systems have stochastic dynamics, the objective takes the form of an expectation over the joint distribution of F_a and F_b, which then needs to be estimated in order to approximate a solution.

A more tractable approach is to treat the problem as semi-stochastic, where each player assumes the other player acts in a noise-free manner, but treats their own dynamics as stochastic. The underlying assumption behind this approach is that the other player will be able to effectively correct for any stochastic disturbances that they encounter, with minimal changes to their trajectory. This semi-stochastic game set-up can be described using dynamics and costs for each player that take the form:

$$F_a(\mathbf{x}_a^t, \mathbf{v}_a^t, \mathbf{u}_a^t) = \begin{pmatrix} \mathbf{f}_a(\mathbf{x}_{a,n}^t, \mathbf{v}_a^t) \\ \mathbf{f}_a(\mathbf{x}_{a,d}^t, \mathbf{u}_a^t) \end{pmatrix}, \quad \mathbf{v}_a^t \sim \mathcal{N}(\mathbf{u}_a^t, \Sigma_a) \quad (7)$$

$$F_b(\mathbf{x}_b^t, \mathbf{v}_b^t, \mathbf{u}_b^t) = \begin{pmatrix} \mathbf{f}_b(\mathbf{x}_{b,n}^t, \mathbf{v}_b^t) \\ \mathbf{f}_b(\mathbf{x}_{b,d}^t, \mathbf{u}_b^t) \end{pmatrix}, \quad \mathbf{v}_b^t \sim \mathcal{N}(\mathbf{u}_b^t, \Sigma_b) \quad (8)$$

The state of each player consists of a noisy copy of the state, $\mathbf{x}_{i,n}^t$ and a deterministic copy $\mathbf{x}_{i,d}^t$, with $i \in \{a, b\}$. The objective functions then take the form:

$$C_a = \mathbb{E}_{\mathbf{f}_a}\left[\phi(\mathbf{x}_{a,n}^T, \mathbf{x}_{b,d}^T) + \sum_{t=0}^{T-1} \mathcal{L}(\mathbf{x}_{a,n}^t, \mathbf{x}_{b,d}^t, \mathbf{u}_a^t, \mathbf{u}_b^t)\right] \quad (9)$$

$$C_b = \mathbb{E}_{\mathbf{f}_b}\left[\phi(\mathbf{x}_{b,n}^T, \mathbf{x}_{a,d}^T) + \sum_{t=0}^{T-1} \mathcal{L}(\mathbf{x}_{b,n}^t, \mathbf{x}_{a,d}^t, \mathbf{u}_a^t, \mathbf{u}_b^t)\right] \quad (10)$$

Note how each players' objective function only considers the stochastic copy of their own state, and the deterministic copy of the other player's state. This eliminates the need to compute the expectation over the joint distribution, but still enables some stochasticity to enter into the problem through the dynamics.

IV. BEST RESPONSE MODEL PREDICTIVE CONTROL

Our problem formulation results in a differential game with non-linear stochastic dynamics, and a potentially non-convex cost function. This type of problem generality is required for controlling ground vehicles in agile close-quarters maneuvers, however it makes finding a solution in an online optimization framework extremely difficult. In this section, we propose a simple iterative solution method which has Nash equilibrium as fixed points. Our algorithm is based on the combination of the game-theoretic notion of iterated best-response, and information theoretic model predictive control, which has been demonstrated as a powerful tool for controlling stochastic non-linear systems subject to non-convex costs.

A. Iterated Best Response

The fundamental object that we consider in iterated best response is the *best response set*:

Definition 2: Assume that player B follows the policy π_b, then the best response set for player A is the set:

$$\left\{\pi_a \mid C_a(\pi_a, \pi_b) = \min_{\pi} \left[C_a(\pi, \pi_b)\right]\right\} \quad (11)$$

A similar definition applies for best response set for player B. Observe how if both players are playing policies that are best responses to each other, than the game is in a Nash equilibrium. Now let $\mathcal{H}_a(\mathcal{X}, \pi_b)$ and $\mathcal{H}_b(\mathcal{X}, \pi_a)$ be functions which take the current state of the game and the opponents strategy, and returns a strategy from the best-response set. The iterative best response system is then defined by the dynamical system:

$$\pi_a^{k+1} = \mathcal{H}_a(\mathcal{X}, \pi_b^k) \quad (12)$$

$$\pi_b^{k+1} = \mathcal{H}_b(\mathcal{X}, \pi_a^k) \quad (13)$$

At each iteration of this system, each player responds by playing their best-response to the other players current policy, and a point is a fixed point in the system if and only if it is a Nash equilibrium. For certain classes of games exhibiting cooperative properties (e.g. potential games), the iterated best response system converges to a Nash Equilibrium. However, in general the system may not converge, but instead cycle between policies. Our key assumption is that the dynamic constraints of the system combined with the stochastic nature of the environment are enough to either prevent cycling, or quickly break it in the event that it does occur. This is similar to the well-established phenomenon of symmetry breaking in standard stochastic optimal control theory [22].

B. Information Theoretic Model Predictive Control

In order to utilize iterated best response in an online optimization scheme, we need a method for rapidly approximating a best-response. In order to perform this computation we use information theoretic model predictive control [19]. Information theoretic model predictive control (IT-MPC) is a sampling-based approach to model predictive control which has been successfully applied to controlling non-linear systems, including agile ground vehicles. In IT-MPC the trajectory optimization problem is treated as a probability matching problem. Let $U = \{\mathbf{u}_0, \mathbf{u}_1, \dots \mathbf{u}_{T-1}\}$ be a sequence of commanded inputs, and define V as a sequence of perturbed inputs with mean U such that:

$$V = U + \mathcal{E} \quad (14)$$

$$\mathcal{E} = \{\epsilon_0, \epsilon_1, \dots \epsilon_T\}, \quad \epsilon_t \sim \mathcal{N}(0, \Sigma) \quad (15)$$

Next, using an information theoretic lower bound, it is possible to show that there exists an optimal distribution over controls, which is optimal in the sense that trajectories sampled from that distribution have a lower cost than any other control distribution. This distribution takes the form:

$$q^*(V) \propto \exp\left(-\frac{1}{\lambda} S(V)\right) p(V) \quad (16)$$

where $S(V)$ is the state-dependent cost of a trajectory and $p(V)$ is the probability density of V from some prior distribution (e.g. zero mean Gaussian) which implicitly defines a control cost. The goal is to then minimize the KL-Divergence between the controlled and optimal distribution, which in turn leads to the following surrogate objective:

$$U^* = \text{argmin}\left[\frac{1}{2}\sum_{t=0}^{T-1}\left(\mathbf{u}_t^{\mathrm{T}}\Sigma^{-1}\left(\mathbf{u}_t - \int q^*(V)\mathbf{v}_t \mathrm{d}V\right)\right)\right] \tag{17}$$

which can be optimized by sampling trajectories from the (simulated) system dynamics, and computing a weighted average over sampled trajectories.

In a model predictive control setting, the algorithm starts with a planned control sequence:

$$(\mathbf{u}_0, \mathbf{u}_1, \dots \mathbf{u}_{T-1}) = U \in \mathbb{R}^{m\times T} \tag{18}$$

It then samples a set of random control sequences, $(V_1, V_2 \dots V_K)$, where:

$$V_k = \left(\mathbf{v}_k^0, \dots \mathbf{v}_k^{T-1}\right), \quad \mathbf{v}_k^t \sim \mathcal{N}(\mathbf{u}_t, \Sigma) \tag{19}$$

Then the IT-MPC algorithm updates the control sequence as:

$$\eta = \sum_{k=1}^{K}\exp\left(-\frac{1}{\lambda}\left(S(V_k) + \gamma\sum_{t=0}^{T-1}\mathbf{u}_t^{\mathrm{T}}\Sigma^{-1}\mathbf{v}_k^t\right)\right) \tag{20}$$

$$U = \frac{1}{\eta}\sum_{k=1}^{K}\left[\exp\left(-\frac{1}{\lambda}\left(S(V_k) + \gamma\sum_{t=0}^{T-1}\mathbf{u}_t^{\mathrm{T}}\Sigma^{-1}\mathbf{v}_k^t\right)\right)V_k\right] \tag{21}$$

The parameter λ determines the selectiveness of the weighted average, and γ modulates the importance of the control cost. In order to be effective, the IT-MPC algorithm must perform millions of dynamics evaluations per second. This can be done in a fast control loop using the parallel processing power of a GPU. In our implementation we sample 1200 trajectories that are each 2 seconds long and sampled at a control frequency of 40 Hz. An outline of the IT-MPC algorithm specific to the game-theoretic setup in our problem formulation is given in Alg. 1, note that the algorithm is given from player A's perspective. The algorithm for player B can be obtained simply by switching the $\{a, b\}$ subscripts.

Algorithm 1: IT-MPC Optimization for Player A

Given: U_a: Control sequence for player A;
U_b: Control sequence for player B;
$\mathbf{x}_a, \mathbf{x}_b$: Current state estimates;
C_a: Cost function for player A;
$\mathbf{f}_a, \mathbf{f}_b$: System dynamics for both vehicles;
K, T: Number of samples and timesteps;
λ, γ: Cost functions/parameters;
$\mathbf{u}_{\min}, \mathbf{u}_{\max}$: Actuator limits;
SF: Convolutional smoothing filter;
$\mathbf{x}_{b,d}^0 \leftarrow \mathbf{x}_b$;
for $t \leftarrow 1$ **to** T **do**
  $\mathbf{x}_{b,d}^t \leftarrow \mathbf{f}_b(\mathbf{x}_{b,d}^{t-1}, \mathbf{u}_b^{t-1})$;
for $k \leftarrow 0$ **to** $K-1$ **do**
  $\mathbf{x}_{a,b}^0 \leftarrow \mathbf{x}_a$;
  Sample $\mathcal{E}^k = \left(\epsilon_0^k \dots \epsilon_{T-1}^k\right),\ \epsilon_t^k \in \mathcal{N}(0, \Sigma)$;
  for $t \leftarrow 1$ **to** T **do**
    $\mathbf{x}_{a,n}^t \leftarrow \mathbf{f}(\mathbf{x}_{a,n}^{t-1}, \mathbf{u}_a^{t-1} + \epsilon_{t-1}^k)$;
    S_k +=
     $c_a(\mathbf{x}_{a,n}^t, \mathbf{x}_{b,d}^t) + \gamma(\mathbf{u}_a^{t-1})^{\mathrm{T}}\Sigma^{-1}(\mathbf{u}_a^{t-1} + \epsilon_{t-1}^k)$;
  S_k += $\phi(\mathbf{x}_{a,n}^T, \mathbf{x}_{b,d}^T)$;
$\beta \leftarrow \min_k[S_k)]$;
$\eta \leftarrow \sum_{k=0}^{K-1}\exp\left(-\frac{1}{\lambda}(S_k - \beta)\right)$;
for $k \leftarrow 1$ **to** K **do**
  $w_k \leftarrow \frac{1}{\eta}\exp\left(-\frac{1}{\lambda}(S_k - \beta)\right)$;
$U_a \leftarrow U_a + \text{SF} * \left(\sum_{k=1}^{K} w_k\mathcal{E}^k\right)$;

C. Best Response MPC

The best response model predictive control algorithm combines the concept of iterated best response from IV-A with the information theoretic optimization procedure from IV-B. The algorithm starts with an initial guess of the other vehicle's control plan, and then estimates the current state of both vehicles. Then, it uses the IT-MPC optimization procedure to simultaneously compute an estimate of the control sequences for both itself and the other agent. Next, it executes the first element in the control sequence, and lastly it slides down the remaining elements in the sequence by one timestep and uses that sequence to start the optimization on the next round.

A key detail in the best response MPC algorithm is that the best responses for both players are computed simultaneously. This is important, because it enables calling the two IT-MPC optimizers in parallel, which significantly reduces the run time of a single best response iteration. The best response MPC (BR-MPC) algorithm is given in Alg. 2. Also, notice how the semi-stochastic game formulation enables the IT-MPC algorithm to run with only a single sampling loop, if the game were formulated as fully stochastic, there would need to be a sampling loop for both players. This would either significantly increase the number of samples required, or drastically increase the variance of the stochastic optimization.

We want to emphasize that the algorithm described in this section (Alg. 2) is meant for a *single vehicle*, and that the two vehicles are not iteratively transmitting any internal planning information to each other. Even though in the BR-MPC algorithm two control plans are computed, one is simply an informed guess at the other vehicles motion, and only one of the control plans is actually used to control a vehicle.

V. IMPLEMENTATION DETAILS

The goal in our experiments was to have two one-fifth scale ground vehicles autonomously operate in close proximity to each other, and to test the limits of the best response MPC algorithm as the target speed was increased. In order to implement the BR-MPC controller for this task, we require

Algorithm 2: BR-MPC for Player i

Given: $\mathcal{G}$: Differential game description;
IT-MPC-OPT: Information theoretic MPC optimizer;
U_a, U_b: Initial control sequences;
$\theta_{\text{IT-MPC}}$: IT-MPC hyper-parameters;
while *task not completed* **do**
  $\mathcal{X} \leftarrow$ GameStateEstimator();
  $U_a' = \text{IT-MPC-OPT}(U_a, U_b, \mathcal{X}, C_a, \mathbf{f}_a, \mathbf{f}_b, \theta_{\text{IT-MPC}})$;
  $U_b' = \text{IT-MPC-OPT}(U_b, U_a, \mathcal{X}, C_b, \mathbf{f}_b, \mathbf{f}_a, \theta_{\text{IT-MPC}})$;
  $U_a = U_a'$;
  $U_b = U_b'$;
  Execute($\mathbf{u}_i^0$);
  for $t \leftarrow 0$ **to** $T-2$ **do**
    $U_a^t \leftarrow U_a^{t+1}$;
    $U_b^t \leftarrow U_b^{t+1}$;
  $U_a^{T-1} = 0$;
  $U_b^{T-1} = 0$;

two key components: a cost function encoding the task, and vehicle dynamics models.

A. Cost Function Design

The state-dependent cost for the task that we are trying to achieve has three main components: (1) Stay on the track, (2) Go close to the set target speed, and (3) Stay 1 meter away from the other vehicle, but do not hit the other vehicle. This last condition is particularly challenging since at 1 meter distance between center of masses, the vehicles are nearly touching, so the algorithm has to balance this objective with the stochastic dynamics of the system. We describe the cost from player A's perspective, the cost for player B is the same, but with the a and b subscripts swapped. Recall that the state of the game is $\mathcal{X}_t = (\mathbf{x}_a^t \ \mathbf{x}_b^t)$. For player A, the cost components encoding the first two instructions are only concerned with the $\mathbf{x}_a$ portion of the state and the control input $\mathbf{u}_a^t$. This part of the cost takes the form:

$$C_a^{ind} = w_1 M(x_a^{pos}, y_a^{pos}) + w_2 \|v_a^x - v_{a,des}^x\|^2 - w_3 \tan^{-1}\left(\frac{v_a^y}{v_a^x}\right)^2 \quad (22)$$

where x_a^{pos} and y_a^{pos} are the position of player A, v_a^x and v_a^y are the body-frame forward and lateral velocities respectively. The function $-\arctan\left(\frac{v_a^y}{v_a^x}\right)$ is the slip angle of the vehicle and the quadratic penalty on slip angle helps stabilize the vehicle. The function $M(x_a^{pos}, y_a^{pos})$ is a cost-map of the track which returns 0 if the vehicle is on the centerline and 1 if the vehicle is on the boundary, it smoothly interpolates in between those values.

In addition to the individual portion of the cost function, the two vehicles have an interaction cost which forces them to stay close to each other:

$$C_a^{mix} = w_4 \left(\|(x_a^{pos}, y_a^{pos}) - (x_b^{pos}, y_b^{pos})\| - D\right)^2 \quad (23)$$

where D is the target distance (set to 1 meter in all of our experiments). Lastly, a crashing cost is added which is activated if the vehicle either collides with the other vehicle or leaves the track:

$$C_a^{crash} = w_5 \beta^t I \quad (24)$$

here β is a time-decay rate, and I is an indicator variable which is 1 is the vehicle crashed and 0 otherwise. The time-decay is added to the crash variable so that, if a crash is unavoidable, the vehicle chooses to wait until the last possible moment to crash. In addition to the state-dependent cost there is a quadratic control cost which is proportional to the sampling variance introduced by the IT-MPC optimization. The hyper-parameters for the cost and optimization were then set as: $(w_1, w_2, w_3, w_4, w_5, \lambda, \gamma) = (100, 4.25, 100.0, 1.0, 10000.0, 0.0015, 0.1)$.

TABLE I
DYNAMICS MODELS PERFORMANCE

	Basis Function Model	Neural Network Model
R2 Score	.68	.78
Mean Squared Error	2.07	1.39
Mean Absolute Error	.93	.76

B. Vehicle Dynamics Models

Pushing the vehicles to their limits while operating nearby each other requires precise agile maneuvering. This means that we need a high-fidelity, non-linear model capable of capturing sliding dynamics, and therefore rules out using simple kinematic models. Additionally, the nature of the dirt track that we perform experiments on makes traditional system identification difficult, so we use a data-driven approach and model the dynamics of the vehicle with a multi-layer neural network. This is the same network from [19].

Although the neural network that we trained is highly accurate, a drawback is that it is computationally expensive, and running two simultaneous IT-MPC optimizations with the neural network is too slow to operate in real-time. As a solution we use two dynamics models, the more accurate neural network is used for the optimization of the vehicle actually being controlled, and a faster, less accurate model is used to predict the motion of the other vehicle. The faster model is a non-linear basis function model described in [23]. The idea behind this double model approach is that, since we do not actually have to control the other vehicle, a less accurate model can be effective as long as it captures the dynamics constraints of the other vehicle. Table I summarizes the performance difference, on a AutoRally driving test set, between the two dynamics models.

VI. EXPERIMENTAL SETUP

Experimental data was collected using a pair of AutoRally robots at the Georgia Tech Autonomous Racing Facility (GT-ARF). Each AutoRally robot ran the BR-MPC algorithm on-board in order to control the vehicle and estimate the future motion of the other vehicle. In principle, no vehicle to vehicle communication is required for BR-MPC where the requisite information can be inferred from onboard sensor information. However, in order to simplify the experiments

TABLE II

AUTORALLY COMPUTE BOX COMPUTING AND POWER COMPONENTS.

Component	Detail
Motherboard	Asus Z170I Pro Gaming, Mini-ITX
CPU	Intel i7-6700, 3.4 GHz quad-core 65 W
RAM	32 GB DDR4, 2133 MHz
GPU	Nvidia GTX-1050ti SC, 768 cores, 4 GB, 1354 MHz
Battery	22.2 V, 11 Ah LiPo, 244 Wh

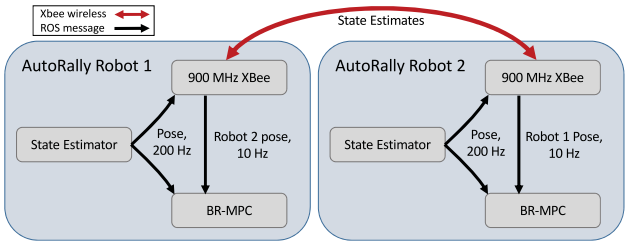

Fig. 2. Vehicle to vehicle system for broadcasting state estimates at 10 Hz from one robot to all other robots within communications range.

Fig. 3. View from the rear vehicle exiting a turn with BR-MPC at the 10 m/s target.

we chose to enable the AutoRally robots to share their pose estimates over low-bandwidth XBee Radios. This removes the requirement of estimating the other vehicles pose using on-board sensors, which is a difficult perception problem by itself.

A. AutoRally Robot

The AutoRally robot is a robust, all-electric autonomous vehicle testbed that is 1:5 the size of a passenger vehicle. The robot is equipped with an onboard Mini-ITX computer housed in a rugged enclosure. The computer configuration is shown in Table II. The sensor package on AutoRally includes 2 forward facing Point Grey USB3.0 cameras with 70 deg FOV lenses, a Lord Microstrain 3DM-GX4-25 IMU, an RTK corrected Hemisphere GPS receiver, and Hall Effect wheel speed sensors. The entire robot weighs approximately 22 kg, measures 0.9 meters in length, and has a top speed of 27 m/s. AutoRally allows for the self-contained testing of all algorithms, with no reliance on external position systems or computation beyond a GPS receiver. Complete instructions to build, configure, and operate AutoRally are available online, as well as a Gazebo simulation environment, ROS interface, and reference controllers [24], [25].

B. Vehicle to Vehicle Communication

We bypass the problem of estimating the pose of the other vehicle by transmitting the state estimate of each vehicle (current position, velocity, heading, and heading rate) over an Xbee wireless radio on each AutoRally robot. The 900 MHz XBee radios provide a high-reliability, low bandwidth network for all vehicles at the test site. Each robot runs a standalone state estimator that fuses IMU and GPS information using the factor graph based optimization packages GTSAM and iSAM2 [26] to produce an accurate state estimate at 200 Hz. The high rates are necessary for high speed, real time control, but would saturate the XBee network with even two vehicles within communication range. For that reason, we implemented a configurable rate, currently set at 10Hz, to down sample pose estimates before transmission over the XBee network. Figure 2 shows the wireless pose communication system for two vehicles and the routing of signals within each robot between the state estimator, XBee interface software, and each robots BR-MPC controller.

C. Baseline Algorithm

As a baseline comparison, we implemented the IT-MPC algorithm treating the other vehicle as a "dumb" obstacle with a constant velocity. All of the other implementation details are the same, but instead of using a second IT-MPC optimizer to simulate the motion of the other vehicle, it is simulated according to the linear extrapolation:

$$x_i^{pos}(t) = x_i^{pos}(0) + t\left(\mathrm{d}x_i^{pos}(0)\right) \tag{25}$$

$$y_i^{pos}(t) = y_i^{pos}(0) + t\left(\mathrm{d}y_i^{pos}(0)\right) \tag{26}$$

This type of model is reasonably accurate on the straight-aways, and on the corners at slow speeds. However, it quickly starts to become inaccurate around corners as the vehicle speed is increased. We refer to this method as Velocity-Obstacle Model Predictive Control (VO-MPC for short).

VII. RESULTS

We tested both the BR-MPC algorithm and the VO-MPC algorithm at the task of maneuvering two AutoRally vehicles around an elliptical dirt track. The desired distance between the two vehicles center of mass was one meter, and the target speed was varied between 5, 6, 7, 8, 9, and 10 m/s. Each speed setting was tested for 10 laps around the test track, except for the 7 m/s VO-MPC setting which was only run for four laps due to safety concerns. This amounts to a total of 84 laps, which is roughly 3.5 miles worth of driving data for each robot. Figure 4 and Table III show the change in distance between the two vehicles as the desired speed is increased from 5 to 10 m/s.

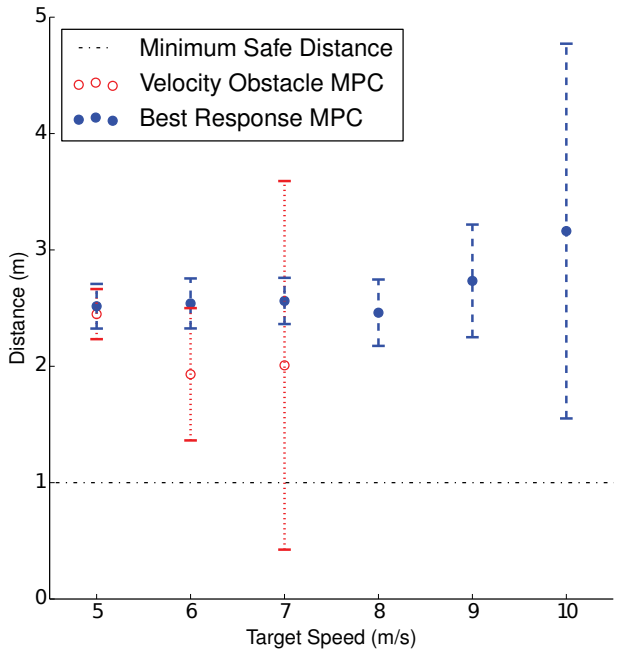

Fig. 4. Following distance and standard deviation for best response dynamics and velocity obstacles. Distance is measured from the center of mass, so a distance of less than one meter indicates a collision if the two vehicles are oriented end-to-end.

TABLE III
FOLLOWING DISTANCE PERFORMANCE

Method	Target	Min Dist.(m)	Max Dist.(m)	Avg. Dist (m)
BR-MPC	5 m/s	2.04	3.40	2.52
BR-MPC	6 m/s	1.92	3.22	2.54
BR-MPC	7 m/s	2.03	3.09	2.56
BR-MPC	8 m/s	1.80	3.80	2.46
BR-MPC	9 m/s	1.83	4.91	2.73
BR-MPC	10 m/s	1.59	10.65	3.16
VO-MPC	5 m/s	1.79	3.25	2.45
VO-MPC	6 m/s	0.58	5.09	1.93
VO-MPC	7 m/s	0.22	7.03	2.01

At the slowest speed setting of 5 m/s, there is minimal performance difference between the simple baseline method, and the BR-MPC method. This is expected, since the distances the AutoRally vehicles travel during the two second time horizon are very short, and can roughly be approximated with straight lines. They both maintain a distance of approximately 2.5 m between the two vehicles, even though the desired distance is 1 m, this extra cushion is automatically included due to the collision penalty and the stochastic dynamics. As the desired speed is increased, the VO-MPC method quickly degrades. At the 6 m/s target, VO-MPC results in 4 distinct collision events during the 10 lap trial run, with one collision requiring a manual take-over of the system. Then, at the 7 m/s target, the two vehicles consistently collide with each other when controlled using VO-MPC.

At the 6 m/s and 7 m/s targets, the BR-MPC method performs nearly identically, in terms of following distance, as the 5 m/s target. As the desired speed is further increased from 7 m/s to 10 m/s, the mean and variance of the following distance both increase. However, the two vehicles avoid ever colliding with each other during the trial runs. Table IV shows the lap statistics for the lead and trail vehicles with the BR-MPC method during the trial runs. At the highest speed target, the vehicles obtain maximum speeds over 8 m/s, while maintaining an average distance of 3.16 meters between their center of mass (this is 2.16 meters from bumper-to-bumper). Additionally, at the highest speed target both vehicles attain a significant side-slip angle, which is the difference between the heading angle and the vehicle's velocity vector, indicating highly dynamic maneuvers. One of the key benefits of the stochastic optimization approach is that only the high level objective needs to be specified, and the precise method for achieving that objective is left to the autonomy system. This is illustrated by Fig. 5, at the 5 m/s target the two vehicles follow nearly the same track with the trail vehicle directly behind the lead vehicle. However, at the 10 m/s target, the behavior looks considerably different, with the two vehicles entering the turns in a staggered formation. This formation enables there to be more room for error in the estimate of the other vehicles longitudinal direction, which is critical for operating at high speeds.

TABLE IV
BR-MPC PERFORMANCE STATISTICS (LEAD / TRAIL)

Target	Avg. Lap Time (s)	Max Speed (m/s)	Max Slip (Deg.)
5 m/s	20.85 / 20.84	3.8 / 4.3	5.6 / 8.8
6 m/s	15.53 / 15.51,	5.1 / 5.4	9.0 / 10.3
7 m/s	12.82 / 12.79	6.0 / 6.2	11.6 / 15.2
8 m/s	11.06 / 11.07	7.2 / 7.3	18.6 / 20.4
9 m/s	9.96 / 9.94	8.0 / 7.9	19.0 / 25.6
10 m/s	9.66 / 9.65	8.2 / 8.1	23.9 / 26.0

VIII. CONCLUSION

In this work we have introduced a best response model predictive control algorithm capable of controlling fast ground vehicles operating in close proximity to each other. The method combines iterated best response with information theoretic stochastic optimal control, in order to try and find a Nash equilibrium for semi-stochastic differential games. We demonstrated the algorithm operating on two one-fifth scale AutoRally robots, and showed that it outperformed a simple baseline method based on linearly extrapolating the other vehicles current position into the future from its initial heading and velocity.

Finding algorithmic solutions to the problem of controlling autonomous ground vehicles, with little or no ability to communicate to the other vehicles around them, is key to getting safe autonomous vehicles deployed in the real world. Our algorithm is one of the first methods which takes a game-theoretic optimization approach to the problem, and which is able to anticipate and react to the other vehicles in the environment only using knowledge of the other vehicles pose, dynamics, and objective. The result is that the two vehicles are able to perform agile maneuvers in close proximity to each other, without sharing any internal planning information.

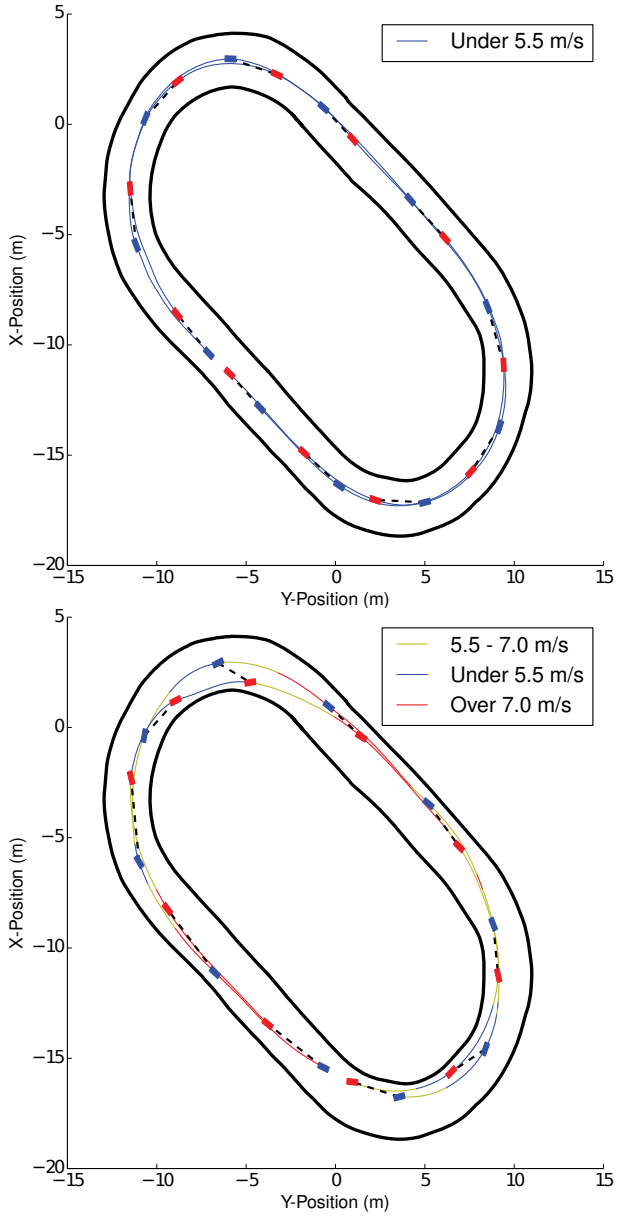

Fig. 5. Top: Following behavior at the 5 m/s target. Bottom: Following behavior at 10 m/s target. The blue marker indicates the lead vehicle, and the red the trail vehicle, with the dashed line showing which markers are synced in time. Colors on the trajectory traces indicate vehicle speed range.

ACKNOWLEDGEMENTS

This work was made possible by the ARO through DURIP award W911NF-12-1-0377, the Georgia Tech Vertical Lift Research Center of Excellence (VLRCOE), and the Qualcomm Innovation Fellowship.

REFERENCES

[1] S. Thrun, "Toward robotic cars," *Communications of the ACM*, vol. 53, no. 4, pp. 99–106, 2010.

[2] P. Tientrakool, Y.-C. Ho, and N. F. Maxemchuk, "Highway capacity benefits from using vehicle-to-vehicle communication and sensors for collision avoidance," in *Vehicular Technology Conference (VTC)*. IEEE, 2011, pp. 1–5.

[3] S. E. Shladover, C. A. Desoer, J. K. Hedrick, M. Tomizuka, J. Walrand, W.-B. Zhang, D. H. McMahon, H. Peng, S. Sheikholeslam, and N. McKeown, "Automated vehicle control developments in the path program," *Transactions on Vehicular Technology*, vol. 40, no. 1, pp. 114–130, 1991.

[4] Y. Zheng, S. E. Li, K. Li, F. Borrelli, and J. K. Hedrick, "Distributed model predictive control for heterogeneous vehicle platoons under unidirectional topologies," *Transactions on Control Systems Technology*, vol. 25, no. 3, pp. 899–910, 2017.

[5] K.-D. Kim and P. R. Kumar, "An mpc-based approach to provable system-wide safety and liveness of autonomous ground traffic," *Transactions on Automatic Control*, vol. 59, no. 12, pp. 3341–3356, 2014.

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

[7] A. Carvalho, Y. Gao, S. Lefevre, and F. Borrelli, "Stochastic predictive control of autonomous vehicles in uncertain environments," in *12th International Symposium on Advanced Vehicle Control*, 2014.

[8] D. Sadigh, S. Sastry, S. Seshia, and A. Dragan, "Planning for autonomous cars that leverage effects on human actions," in *Robotics Science and Systems (RSS)*, 2016.

[9] R. Miller and Q. Huang, "An adaptive peer-to-peer collision warning system," in *Vehicular Technology Conference.* IEEE, 2002, pp. 317–321.

[10] S. Ammoun and F. Nashashibi, "Real time trajectory prediction for collision risk estimation between vehicles," in *International Conference on Intelligent Computer Communication and Processing*. IEEE, 2009, pp. 417–422.

[11] M. Liebner, M. Baumann, F. Klanner, and C. Stiller, "Driver intent inference at urban intersections using the intelligent driver model," in *Intelligent Vehicles Symposium*. IEEE, 2012, pp. 1162–1167.

[12] C. Hermes, C. Wohler, K. Schenk, and F. Kummert, "Long-term vehicle motion prediction," in *Intelligent Vehicles Symposium*. IEEE, 2009, pp. 652–657.

[13] S. Lefèvre, D. Vasquez, and C. Laugier, "A survey on motion prediction and risk assessment for intelligent vehicles," *Robomech Journal*, 2014.

[14] J. Morimoto, G. Zeglin, and C. G. Atkeson, "Minimax differential dynamic programming: Application to a biped walking robot," in *Intelligent Robots and Systems (IROS)*, vol. 2. IEEE, 2003, pp. 1927–1932.

[15] J. Walrand, E. Polak, and H. Chung, "Harbor attack: A pursuit-evasion game," in *49th Annual Allerton Conference on Communication, Control, and Computing*. IEEE, 2011, pp. 1584–1591.

[16] S. Pan, H. Huang, J. Ding, W. Zhang, C. J. Tomlin, *et al.*, "Pursuit, evasion and defense in the plane," in *American Control Conference (ACC)*. IEEE, 2012, pp. 4167–4173.

[17] Y. Li, K. P. Tee, W. L. Chan, R. Yan, Y. Chua, and D. K. Limbu, "Continuous role adaptation for human–robot shared control," *Transactions on Robotics (T-RO)*, vol. 31, no. 3, pp. 672–681, 2015.

[18] D. Fudenberg and D. K. Levine, *The theory of learning in games*, 1998, vol. 2.

[19] G. Williams, N. Wagener, B. Goldfain, P. Drews, J. M. Rehg, B. Boots, and E. A. Theodorou, "Information theoretic mpc for model-based reinforcement learning," in *International Conference on Robotics and Automation (ICRA)*. IEEE, 2017, pp. 1714–1721.

[20] E. Theodorou, J. Buchli, and S. Schaal, "A generalized path integral control approach to reinforcement learning," *Journal of Machine Learning Research*, vol. 11, no. Nov, pp. 3137–3181, 2010.

[21] M. Kobilarov, "Cross-entropy motion planning," *The International Journal of Robotics Research*, vol. 31, no. 7, pp. 855–871, 2012.

[22] H. J. Kappen, "Path integrals and symmetry breaking for optimal control theory," *Journal of statistical mechanics: theory and experiment*, 2005.

[23] G. Williams, P. Drews, B. Goldfain, J. M. Rehg, and E. A. Theodorou, "Aggressive driving with model predictive path integral control," in *International Conference on Robotics and Automation (ICRA)*. IEEE, 2016, pp. 1433–1440.

[24] B. Goldfain. (2017) Autorally platform instructions. [Online]. Available: https://github.com/AutoRally/autorally_platform_instructions

[25] ——. (2017) Autorally software. [Online]. Available: https://github.com/AutoRally/autorally

[26] M. Kaess, H. Johannsson, R. Roberts, V. Ila, J. J. Leonard, and F. Dellaert, "iSAM2: Incremental smoothing and mapping using the Bayes tree," *International Journal of Robotics Research*, vol. 31, no. 2, SI, pp. 216–235, FEB 2012.