

Comparing Game Planners

Sherman Luo
Anca Dragan

Electrical Engineering and Computer Sciences
University of California at Berkeley

Technical Report No. UCB/EECS-2020-44

<http://www2.eecs.berkeley.edu/Pubs/TechRpts/2020/EECS-2020-44.html>

May 12, 2020



Copyright © 2020, by the author(s).
All rights reserved.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission.

Acknowledgement

First and foremost I'd like to give thanks to my research advisor Professor Anca Dragan for her valuable guidance. She was extremely helpful and available, and was generally a fantastic mentor who gave me direction during difficulty. This project would not have been possible without her continuous advice and co-authorship. I'd also like to extend thanks to the members of the InterACT Lab, the incredibly supportive environment in which I have had the honor of conducting my research. In particular I am grateful towards Gokul Swamy and Lawrence Chan, who helped me formulate my ideas and generally assisted me with the writing portion of this thesis. Lastly, thank you to Professor Alistair Sinclair, who was the second reader of this thesis and provided suggestions to improve the writing.

Comparing Game Planners

by

Sherman Andrew Luo

A thesis submitted in partial satisfaction of the
requirements for the degree of

Master of Science

in

Electrical Engineering and Computer Science

in the

Graduate Division

of the


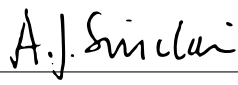
University of California, Berkeley

Committee in charge:

Professor Anca D. Dragan, Chair
Professor Alistair Sinclair

Spring 2020

The thesis of Sherman Andrew Luo, titled Comparing Game Planners, is approved:

Chair		Date	May 12, 2020
	_____	Date	_____
		Date	12 May, 2020
	_____	Date	_____

University of California, Berkeley

Comparing Game Planners

Copyright 2020
by
Sherman Andrew Luo

Abstract

Comparing Game Planners

by

Sherman Andrew Luo, Anca D. Dragan

Master of Science in Electrical Engineering and Computer Science

University of California, Berkeley

Professor Anca D. Dragan, Chair

What makes interactions challenging for robots that navigate around people in general, and autonomous cars in particular, is the need to account for the mutual influence between the robot's actions and the human's. These interactions are best modeled by dynamic game theory, which in turn motivates the use of *game-theoretic* planners for autonomous cars. Recent work proposed a number of such planners that leverage trajectory optimization and Model Predictive Control, but adapt them to account for the strategic, multi-agent structure of the problem. In this work, we provide a quantitative and qualitative empirical analysis of the performance of these planners, with the goal of gaining a deeper understanding of their advantages and limitations in challenging interactive driving situations.

Contents

Contents	i
List of Figures	ii
List of Tables	iv
1 Introduction	1
2 Game-Theoretic Planners	3
2.1 Problem Statement	3
2.2 Planning Algorithms	3
3 Methods and Experiments	6
3.1 Driving Environment	6
3.2 Scenarios and Test Cases	7
3.3 Independent Variables	8
3.4 Dependent Measures	10
4 Results and Analysis	11
4.1 When the Robot’s Assumptions are Correct	11
4.2 When the Robot’s Assumptions are Incorrect	17
4.3 Runtime	18
5 Discussion	19
5.1 Takeaways	19
5.2 Limitations and Future Work	19
Bibliography	21

List of Figures

- 3.1 From left to right: Upcoming Obstacle, Gap Opening, Merge, Deter. Each task requires the robot to position itself above the human in the left lane. Each task’s goal region of the current world state is shown by the yellow area. In Gap Opening especially, it is immediately apparent that it will be impossible to achieve the goal without forcing the human to slow down. 7
- 4.1 **Top:** The left bar represents the normalized reward accrued over the 50 time-steps in the case when the human is Obedient – executes the robot’s prediction. Each task’s results were normalized by subtracting the minimum response then dividing by the difference between the maximum and minimum responses for that task. The right bar examines all the cases in which the human is *not* obedient by normalizing $\sum_{t=0}^{49} \hat{\mathbf{u}}_{\mathbf{H}}^t(\mathbf{accel}) - \mathbf{u}_{\mathbf{H}}^t(\mathbf{accel})$ for each task + human planner joint in the same manner. The more an algorithm’s prediction deviates on average from our human planners through slowing the human, the better the performance when predictions are correct. **Bottom:** seven figures represent each algorithm’s final rollout in Upcoming Obstacle. We plot the positions of the cars relative to the truck obstacle, which is moving at a constant velocity over time. 12
- 4.2 IBR-hDef (Success), IBR-hAgg (Fail), CA-Def (Success), CA-Agg (Fail) at a certain time-step of a version of Upcoming Obstacle – note that the truck is also moving. This shows how each algorithm iterates upon their plans during a certain timestep. The last iteration for each planner visually almost equivalent to the final solution returned. For algorithms that succeeded (CA-Def, IBR-hDef), we investigated the first time-step they decide to do the maneuver. For the others we show a time-step when the successful initialization decided to first maneuver. We can see that initialization can directly affect the solution that a planner converges to. 13

- 4.3 At $t = 25$ for Upcoming Obstacle using the Nested Planner and Obedient human, we run IBR-hDef, CA-Agg, CA-Def, and CA-Ign. This plots the progression of their iterations, comparing the magnitude of the robot plan and the mean acceleration of the human plan (note that CA-Def and IBR-hDef are initialized at the same place). CA-Agg, IBR-Def, and Nested converge to the same solution and do not attempt to merge. CA-Def and CA-Ign converge to a solution that sub-optimally attempts the maneuver due to the gradient pointing towards the edge of the control bounds (note lower robot reward). IBR-hDef approaches this solution initially, but escapes and converges at a better solution. Note that the truck is moving at a constant velocity. 14
- 4.4 The progression of CA-Ign over time in Gap Opening (note the truck is moving at constant velocity). We can see that even though the initialization plans are near collision, CA-Ign slowly adjusts the two plans until the maneuver is much more reasonable. Note that the truck moves at constant velocity. 15
- 4.5 **Left:** The mean of the normalized performance in reward of each robot planner against each human planner. We normalized each response over tasks, by subtracting the minimum response and dividing by the difference between the maximum and minimum performance on that task. As the human becomes more and more aggressive (rightwards in the figure), the robot loses more and more reward, as evident by the negative gradient from left to right (barring CA-Agg). **Right:** The outcomes of the maneuver. Green for *success*, Pink for *failure*, and Red for a *collision*. We can see that as we move from left to right in a particular grid we generally transition from green to pink or green to red. 16

List of Tables

Acknowledgments

First and foremost I'd like to give thanks to my research advisor Professor Anca Dragan for her valuable guidance. She was extremely helpful and available, and was generally a fantastic mentor who gave me direction during difficulty. This project absolutely would not have been possible without her continuous advice and co-authorship. I'd also like to extend thanks to the members of the InterACT Lab, the incredibly supportive environment in which I have had the honor of conducting my research. In particular I am grateful towards Gokul Swamy and Lawrence Chan, who helped me formulate my ideas and generally assisted me with the writing portion of this thesis. Lastly, thank you to Professor Alistair Sinclair, who was the second reader of this thesis and provided suggestions to improve the writing.

Chapter 1

Introduction

Whenever a human and a robot navigate around each other in the same space, what is optimal for the robot to do depends on what the human ends up doing as well. Much work in robotics, including social navigation and autonomous driving, has focused on how a robot might be able to anticipate what the human will do, and plan its actions accordingly [2, 7, 16, 22, 10].

Imagine a car trying to merge in heavy traffic. Anticipating what the humans will do seems easy: they will continue driving at their current speed. Unfortunately, that leaves the car thinking there is no way for it to complete the merge, and it gets stuck waiting for a large enough gap in traffic. In contrast, humans in that situation would manage to make their way in. This is because we, the humans, know very well that when we attempt the maneuver the person behind will have to slow down. Much as human actions affect other human actions, we should expect that *robot* actions too will affect human actions [12, 19]. Our robots need to leverage this effect in order to seamlessly navigate the world around us [16].

Unfortunately, it gets a bit more complicated. When someone merges in front, most of the times people do slow down. But, not too often, they do the opposite: they accelerate and deter the car in front from completing the merge. So, not only do robot actions affect human actions, but humans too might try to affect the robot's actions.

This mutual influence between the robot and the human is best captured by a *dynamic general-sum game* [7]. The robot has an objective, and the human can have a slightly different objective – they both want to avoid collisions with each other, be efficient, obey traffic rules, drive comfortably, etc, but may have different goals in mind.

This realization presents a big challenge for cars today: cars live in continuous state and action spaces, where non-strategic planning is complex enough. Prior work has started tackling this challenge by introducing several approaches for computing approximate solutions, all in the context of Model Predictive Control[14]: the robot would find a plan, execute a first step, and replan after observing the new state of the world, including the new human state. Engwerda et al. [6] explores the existence of solutions when the game is Linear-Quadratic. Sadigh et al. [16] proposed to turn the dynamic game into a static Stackelberg [17] game

where actions are full trajectories: the robot plans a trajectory, and the human computes a best response. Wang et al. [21] proposed a solution based on Iterated Best Response: the planner initializes one of the player’s trajectories, computes the other player’s best responses, and iterates. Fisac et al. [7] proposed the addition of a value function computed by a coarse approximation as a terminal cost, which can benefit any such approach, but also anecdotally reported that a “coordinate ascent” (one gradient step at a time instead of a full best response performed better).

Overall, there are different ways to slice and dice the general sum game, all in the context of Model Predictive Controllers for autonomous cars. What we seek in this thesis is to shed some light on how these approaches compare in terms of the solutions they produce, so that we can make more informed choices about what to deploy on our robots, and so that we can learn what we should build on and what remains to be improved with these algorithms.

The key challenge with making such a comparison is that each algorithm comes up not only with a different solution for the robot’s plan, but also for the *human*’s plan. A solution might look very good, but because it is perhaps making a strong assumption about what the human will do. To address this, we compare the solutions both with human behavior that *follows* each planner’s assumption, as well as when the behavior *deviates*, looking at different ways people might be approximating the game for themselves.

Overall, we find coordinate ascent methods to appear to perform well but actually be unreliable, the static Stackelberg approach to be robust but too aggressive with people who are trying to influence the robot, and we find Iterated Best Response to require clever initialization; these methods will be described in detail in Chapter 2. We hope our results can serve to inform practitioners about which choice makes the most sense for their settings and requirements.

Chapter 2

Game-Theoretic Planners

2.1 Problem Statement

We study the general-sum game formulation of interaction between an autonomous agent R and a human H . We consider fully observable continuous states x which in our driving application consist of positions, orientations, and velocities for both agents. Over discrete time-steps, each agent can apply continuous controls u_R and u_H which change state via the dynamics $x^{t+1} = f(x^t, u_R^t, u_H^t)$. Given a horizon of N such that $\mathbf{u}_A = u_A^{1:N}$, each agent's objective is to maximize their own cumulative reward over the horizon

$$R_R(x^0, \mathbf{u}_R, \mathbf{u}_H) \text{ and } R_H(x^0, \mathbf{u}_R, \mathbf{u}_H)$$

In driving, the human and the robot are mutually interested in avoiding collisions, but each is selfishly interested in their own efficiency, which is what makes this a general-sum game. In this work, we engineer multiple versions of R_R and assume we have access to R_H (which we explain later).

2.2 Planning Algorithms

Crosscutting technique: MPC

Planners in this domain typically used Model Predictive Control [1, 3, 9], where they iteratively generate a \mathbf{u}_R^* (and, implicitly or explicitly, a prediction \mathbf{u}_H^* for the human), take a first step, and replan after observing the human's actual action. We will differentiate in this thesis between *plans* at every time-step produced for a time horizon but never fully executed, and *MPC rollouts* which happen as the planners are run at each time-step and only their first action executed.

Originally, planners would not consider the game-theoretic interaction and split the problem into 1) predicting the human actions \mathbf{u}_H , and 2) computing a plan \mathbf{u}_R for the robot that assumes the human actions are fixed and unchangeable [13, 5, 8]. For instance, the robot

might predict for the human:

$$\mathbf{u}_H^* = \arg \max_{\mathbf{u}_H} R'_H(x^0, -, \mathbf{u}_H)$$

where R' eliminates any collision cost with the robot, modeling that the person ignores the robot's existence as they plan what to do. The robot would then compute its own best response to this,

$$\mathbf{u}_R^* = \arg \max_{\mathbf{u}_R} R_H(x^0, \mathbf{u}_R, \mathbf{u}_H^*)$$

Unfortunately, without accounting for the influence that the robot can have on the human, certain maneuvers like merging in heavy traffic proved impossible – in reality, people do take the robot into account when planning. However, solving the game exactly is intractable in continuous state and action spaces, and several approximations have been proposed.

Nested

One approach to approximating the game and capturing the robot's influence on the person's plan is to model the person as computing a best response to the robot [16]. With this, the robot's prediction of what the person will do becomes a *function* of the robot's plan:

$$\mathbf{u}_H^*(\mathbf{u}_R) = \arg \max_{\mathbf{u}_H} R_H(x^0, \mathbf{u}_R, \mathbf{u}_H)$$

The robot then solves the *nested* optimization problem of computing a plan for itself that, when coupled with the human's best response plan, yields the largest reward possible:

$$\mathbf{u}_A^* = \arg \max_{\mathbf{u}_A} R_A(x^0, \mathbf{u}_A, \mathbf{u}_H^*(\mathbf{u}_A))$$

We solve this optimization problem using the quasi-Newton optimizer L-BFGS¹ that computes an approximate inverse Hessian using the gradient. We compute the gradient via implicit differentiation, as in [16].

Note that the Nested planner computes the solution to a Static Stackelberg game in which actions are full trajectories, and in which the robot leads.

Iterated Best Response (IBR)

The Iterated Best Response Planner [21] initializes one of the players' trajectory, then iterates though the players computing each trajectory as a best response to the other player. If it converges, the two trajectories are best responses to each other, corresponding to a Nash equilibrium [15]. The algorithm begins by initializing one of the players – assume for now

¹To clarify, we actually use L-BFGS-b [4, 20] instead of vanilla L-BFGS to constrain the controls of our final plan to a bounding box.

we initialize the human to \mathbf{u}_H^0 . We then iteratively compute a best response for the robot and then for the human:

$$\mathbf{u}_R^i(\mathbf{u}_H^i) = \arg \max_{\mathbf{u}_R} R_A(x^0, \mathbf{u}_R, \mathbf{u}_H^i)$$

$$\mathbf{u}_H^{i+1}(\mathbf{u}_R^i) = \arg \max_{\mathbf{u}_H} R_H(x^0, \mathbf{u}_R^i, \mathbf{u}_H)$$

We use L-BFGS-b to do each optimize subroutine, which means our responses are only locally optimal and convergence reaches a local Nash. Although here we started with a human initialization, the algorithm can choose either the robot or the human to initialize. In our experiments, we also test the algorithm’s sensitivity to both which player gets initialized and how.

Coordinate Ascent (CA)

Even though this algorithm has never been formally proposed in prior work, some recent thesiss anecdotally mention seeing good performance out of a “coordinate ascent” version of IBR– rather than computing full best responses, each agent only takes a gradient step on their maximization at each iteration:

$$\mathbf{u}_R^{i+1} = \mathbf{u}_R^i + \alpha \nabla_{\mathbf{u}_R} R_R(x^0, \mathbf{u}_R^i, \mathbf{u}_H^i)$$

$$\mathbf{u}_H^{i+1} = \mathbf{u}_H^i + \alpha \nabla_{\mathbf{u}_H} R_H(x^0, \mathbf{u}_R^i, \mathbf{u}_H^i)$$

Interestingly, unlike IBR, CA requires initializing both the robot and the human’s trajectory. This will prove to be an important detail in our experiments.

Chapter 3

Methods and Experiments

We designed an experiment to help us better understand the performance of these algorithms for highly interactive driving situations, and under different hypothetical behaviors for the human.

3.1 Driving Environment

We conduct experiments in a two-lane driving simulator¹ with one robot vehicle, one human vehicle, and potentially a large truck obstacle moving at constant speed. Both controls and states are continuous, and the discretized time-step is 0.1 seconds. A state in our world represents the joint state of both cars. Each car’s state $[w, h, \theta, v]^\top$ encodes their 2D position, velocity, and heading (orientation). Controls $[u_1, u_2]^\top$ encode the steering and linear acceleration along the heading of the car. Both cars use the Bicycle dynamics model [11] for motion, running for 50 time-steps or 5 seconds.

$$[\dot{w}, \dot{h}, \dot{\theta}, \dot{v}] = [v \cos(\theta), v \sin(\theta), vu_1, u_2 - \gamma v]$$

Rewards for both cars include features that encode safety, efficiency, and a goal incentive. Safety features include penalties in the form of elliptical Gaussians around the position of other cars and obstacles, Gaussian reward for being in the center of a lane, and sigmoids for both sides of the highway for being on the road. Efficiency features include quadratic penalties for controls of high magnitude and sigmoids for staying within control bounds, and goal features encode a higher weighting for being in the left lane, quadratic penalties for deviating from a target velocity, and a sigmoid for either a bonus for being in front of the human or for not being behind the human.

¹The environment is heavily adapted from [16] and [7]

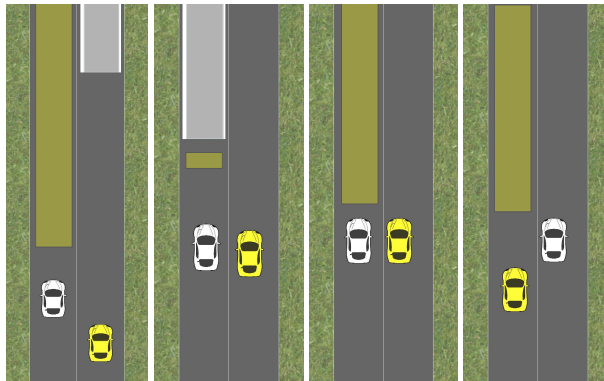


Figure 3.1: From left to right: Upcoming Obstacle, Gap Opening, Merge, Deter. Each task requires the robot to position itself above the human in the left lane. Each task’s goal region of the current world state is shown by the yellow area. In Gap Opening especially, it is immediately apparent that it will be impossible to achieve the goal without forcing the human to slow down.

3.2 Scenarios and Test Cases

We designed four scenarios for driving, shown in Fig 3.1, and introduced three variations of each scenario to vary the difficulty level for the robot. We chose these scenarios to be highly interactive, where in order for the robot to do well it does need to influence the behavior of the human-driven vehicle. Our scenarios consist of different initial conditions, and different reward functions for the human and the robot which give rise to the notion of what a “successful” maneuver for the robot would be.

Upcoming Obstacle

The human car starts in the left lane and the robot in the right lane with a penalty for the robot being behind the human. The robot must speed up and overtake the human car, but there is a large truck obstacle in the right lane that is slowing down and limiting the space that the robot has to merge.

Gap Opening

Gap Opening is a scenario in which the robot must squeeze itself between the human and the truck in the left lane. There is a bonus for the robot for being in front of the human. It is impossible to do this maneuver without the human slowing down, and the robot is successful if it is able to merge into the left lane in front of the human.

Merge

This is the same as Upcoming Obstacle except without a truck. The robot successfully does the maneuver by speeding up and merging into the human’s lane. Again, there is a robot penalty for being stuck behind the human.

Deter

The robot starts in the left lane and the human in the right lane ahead of the robot, with a robot bonus for being in front of the human. The human wants to merge left, but the robot must speed up quickly enough to convince the human not to merge into the left lane in front of it. The robot succeeds if it is able to speed past the human.

We introduced three test cases for Upcoming Obstacle and Gap Opening, and two test cases for Merge and Deter. We created these by varying the robot’s reward function in the weight for crashing, the spread of the collision Gaussians, the bonus for being in the left lane, and the bonus for being in front of the human when applicable.

3.3 Independent Variables

We do not only manipulate which planner we use, but also look at initialization, and more importantly, at different human behaviors that we can test the planner against. Note that each planner also produces a plan for the human, \mathbf{u}_H , implicitly assuming this is what the human intends to execute: while it is interesting to look at the combination of the robot’s and the human’s plans and the corresponding MPC rollouts, the reality is that we do not control human behavior, and it is just as important to understand the robustness of the robot’s planner. We thus manipulate three factors.

Main Robot Planner

This is the planner that the robot is running. The planners used include **Nested**, Iterated Best Response (**IBR**), and Coordinate Ascent (**CA**).

Initialization

The Nested algorithm can be thought about as one optimization for the robot (in an “underactuated” system where the reward depends on what the human does, but this is simply a function of what the robot does) – it is a static Stackelberg solution in which the human does not attempt to influence the robot. On the other hand, IBR and CA are qualitatively different because the solutions for the human and the robot are constructed from each other, allowing the robot to influence the human, but also the human to influence the robot. For IBR, each player runs at each step a full optimization to convergence. Therefore, when we

talk about initialization, we differentiate the initialization to a local optimizer for a player (which we handle by running multiple initializations in parallel²), from the initialization of IBR and CA (initial plans e.g. \mathbf{u}_H^0) which can influence the equilibrium to which the algorithm converges. We focus on the latter kind, seeking to understand the influence of the initial conditions on which equilibrium the planner finds. For the most part, we ignore the former notion of initialization and assume our local optimization techniques are globally optimal.

For Iterated Best Response, we compare initializing the human vs. initializing the robot. For the human initialization, we differentiate between a heuristically chosen “defensive” initialization, **IBR-hDef**, that is preferable for the robot, and an “aggressive” initialization **IBR-hAgg**.^{3,4} For the robot initialization of IBR, we initialize the robot to ignore the human and maximize its reward in isolation – we call this **IBR-rIgn**.

CA, unlike IBR, requires initializing both the robot and the human trajectories. We take the aggressive and defensive human initializations for IBR, compute the robot’s best response to them, and use those as initializations for CA for an apples-to-apples comparison. These result in what we call **CA-Def** and **CA-Agg**. But in addition, what is perhaps nice about CA is that it does not require the two trajectories, for the robot and the human, to be “compatible” in any way. We thus also experiment with **CA-Ign**, which initializes the robot to the same “ignore the human” initialization from **IBR-rIgn**, but initializes the human additionally to a “defensive” initialization. We do this instead of the human best-responding to the ignoring robot, because we have found those responses to be unstable, with e.g. the human having to go off-road to avoid the robot. As we will see in the analysis, by having a defensive but not extreme initialization, the algorithm produces better solutions.

Ground Truth Human Behavior

We manipulate the planner that the *actual* simulated human is running. We test the **Obedient** human, who steps along the plan computed by the robot at every time-step – note

²We run optimizations using L-BFGS-b, which is a local optimization algorithm. Unfortunately, rewards in our system are non-convex functions, and this often resulted in sub-optimal behavior. To combat this issue, we run L-BFGS four times from four different initial guesses: maximum left steering, maximum right steering, maintaining speed, and the agent’s plan at the previous time-step, giving our algorithms sufficient consistency and stability.

³For the first three scenarios which require lane changes, we heuristically choose these such that the defensive human slows down, while the aggressive human accelerates. For the Deter scenario, finding such heuristics is more difficult, and we actually found it easier to compute them by responding to a heuristic robot behavior: the defensive human responds to a robot that ignores the human, and the aggressive initialization responds to a robot that speeds up.

⁴Note that our goal here is merely to test sensitivity of the algorithms to different kinds of initializations, which is why we get to create these heuristically/manually. In real life, the robot would have to produce them autonomously – if these planners are sensitive to initialization, and if initializations are hard to produce autonomously, it might suggest that sacrificing some performance for more initialization robustness is preferable.

that as a result, this human is different for different planners. But we also simulate human behavior that deviates from these planners.

- A human that runs IBR themselves, initializing the *robot* first, with a neutral initialization that maintains speed. We call this **IBR-Neutral**.
- An aggressive human who tries to *influence* the robot by running the **Nested** planner, with the human and robot roles *flipped*: now the human expects the robot to compute a best response to its plan. With this, we test to what extent the solutions produced by these planners can handle very aggressive human driving.
- An unrealistically aggressive human who *ignores* the robot, solving their optimization problem without any collision costs with the robot. We call this the **Ignore** human.

3.4 Dependent Measures

We measure performance by the **reward** the robot accrues in the MPC rollout. We also annotate the resulting behavior's **success**: whether the intended maneuver (merging, going in front of the human) succeeded without collisions (success), whether it failed without collisions (fail), or whether it resulted in a collision (critical). Lastly, we measure the **computation time** it takes for these algorithms to produce solutions.

Chapter 4

Results and Analysis

We split our analysis into two parts: when the human acts according to the robot’s prediction (Obedient human planner) and when the human deviates from the robot’s plan.

4.1 When the Robot’s Assumptions are Correct

We first examine experiment results in which the robot’s predictions are completely correct and the human is Obedient: the human is simulated as an agent that acts exactly as the robot predicts. This gives us an understanding of how these planners compare via the different assumptions they are implicitly making about the human. Even though in reality we care most about how our planners do against real, unknown humans, understanding the assumptions each planner makes will give us insight on their performance on humans that may break said assumptions.

Overall performance

Fig 4.1 plots the (normalized) reward each planner gets across all test cases. We immediately notice quite a bit of difference among the planners and initializations. CA-Ign performs best, followed by CA-Def and Nested, then followed by IBR-hDef and IBR-rIgn, with IBR-hAgg and CA-Agg performing the worst (and often failing at the maneuver). The figure also shows example MPC rollouts for each for the Upcoming Obstacle scenario, and we can see how CA-Ign merges the most aggressively and earliest, causing the person to slow down the most. Ca-Def is slightly later, with a bit less reaction from the human. Nested has a later merge, and IBR-Def even later. In this case, IBR-Ign completely fails the merge and stays behind the truck.

Note that initialization has a large effect over the final solution, and we discuss this next. But also note that for the same defensive initialization, IBR-hDef is worse than CA-Def. We’ve found the reason for this to be somewhat unsatisfying: by not taking full best responses, CA ends up with a solution closer to the initialization compared to IBR. CA,

initialized with the defensive human and the robot responding by merging, ends up with a solution in which the robot merges. This solution is at the control bounds of the robot, which counter the gradient that would push the robot to accelerate and get further from the human. IBR sometimes does not find that merge solution through best responses, and

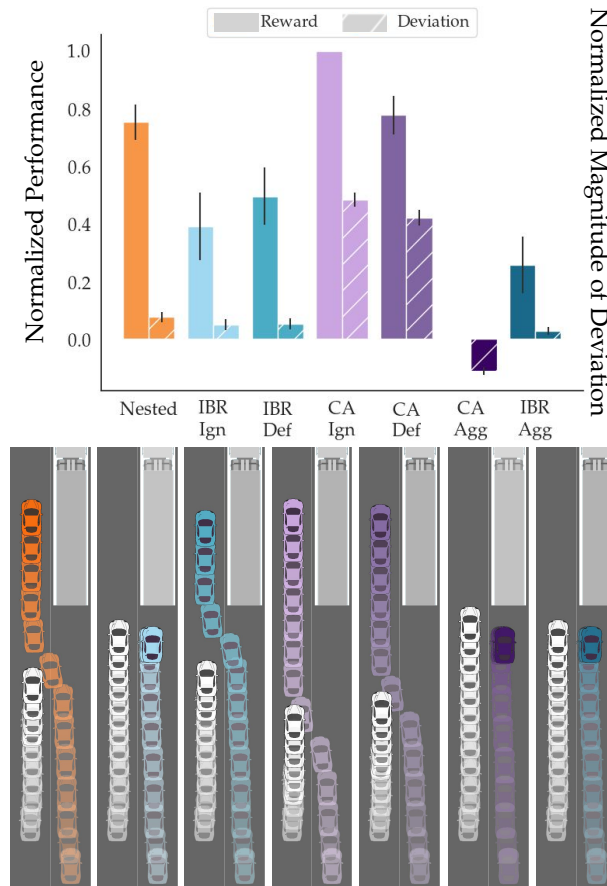


Figure 4.1: **Top:** The left bar represents the normalized reward accrued over the 50 time-steps in the case when the human is Obedient – executes the robot’s prediction. Each task’s results were normalized by subtracting the minimum response then dividing by the difference between the maximum and minimum responses for that task. The right bar examines all the cases in which the human is *not* obedient by normalizing $\sum_{t=0}^{49} \mathbf{u}_H^t(\text{accel}) - \mathbf{u}_H^t(\text{accel})$ for each task + human planner joint in the same manner. The more an algorithm’s prediction deviates on average from our human planners through slowing the human, the better the performance when predictions are correct. **Bottom:** seven figures represent each algorithm’s final rollout in Upcoming Obstacle. We plot the positions of the cars relative to the truck obstacle, which is moving at a constant velocity over time.

instead converges to a non-merge solution that in fact has higher reward. We can see an example of this in Fig 4.3. Despite the CA solutions being worse than the IBR solutions for the planning horizon with respect to reward, they end up leading to higher cumulative reward in the MPC rollout by completing a closer merge earlier.

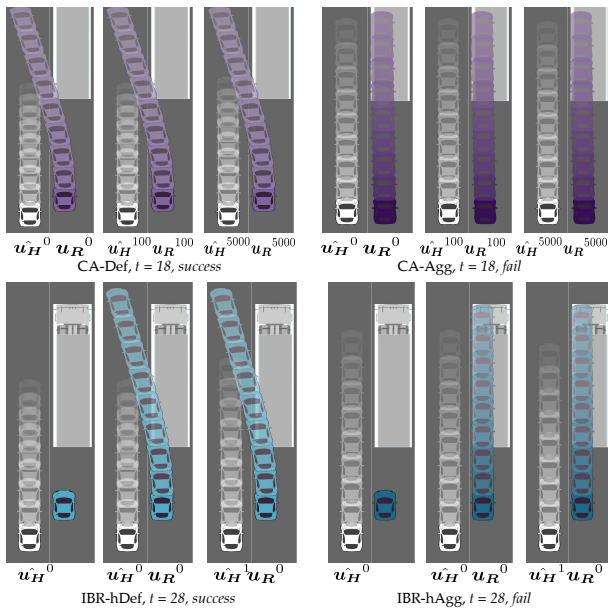


Figure 4.2: IBR-hDef (Success), IBR-hAgg (Fail), CA-Def (Success), CA-Agg (Fail) at a certain time-step of a version of Upcoming Obstacle – note that the truck is also moving. This shows how each algorithm iterates upon their plans during a certain timestep. The last iteration for each planner visually almost equivalent to the final solution returned. For algorithms that succeeded (CA-Def, IBR-hDef), we investigated the first time-step they decide to do the maneuver. For the others we show a time-step when the successful initialization decided to first maneuver. We can see that initialization can directly affect the solution that a planner converges to.

Fig 4.2 (bottom) showcases the difference for IBR between the defensive and aggressive human initialization. When the human is initialized aggressively, the robot’s best response is to keep going behind the truck (remember the truck is moving at constant velocity). Then, the human’s best response is almost the same – the human accelerates a bit less compared to the initialization. From there, the robot responds as before, and an equilibrium is reached. The same happens for CA (top right): we start from the aggressive human and the robot’s response, and then the gradients adjust the trajectories slightly, but there is nothing that pushes the robot to do the merge.

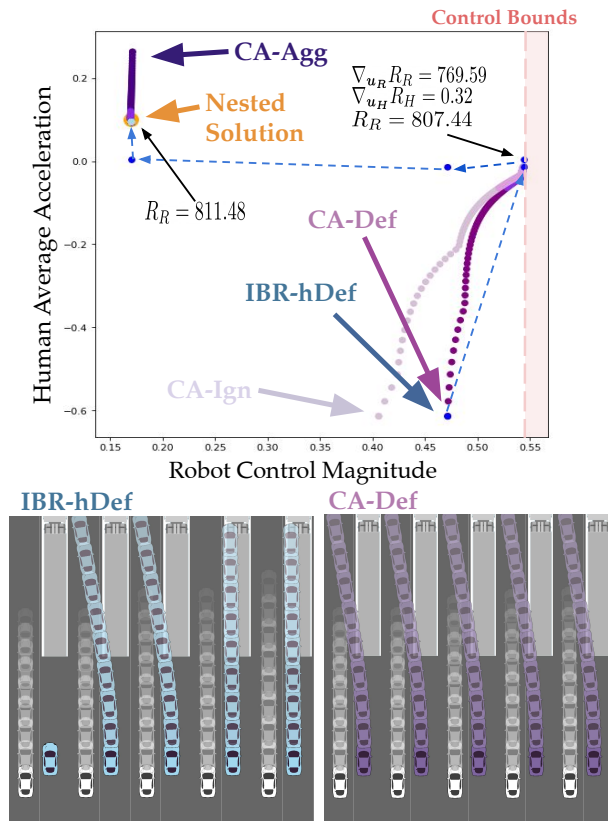


Figure 4.3: At $t = 25$ for Upcoming Obstacle using the Nested Planner and Obedient human, we run IBR-hDef, CA-Agg, CA-Def, and CA-Ign. This plots the progression of their iterations, comparing the magnitude of the robot plan and the mean acceleration of the human plan (note that CA-Def and IBR-hDef are initialized at the same place). CA-Agg, IBR-Def, and Nested converge to the same solution and do not attempt to merge. CA-Def and CA-Ign converge to a solution that sub-optimally attempts the maneuver due to the gradient pointing towards the edge of the control bounds (note lower robot reward). IBR-hDef approaches this solution initially, but escapes and converges at a better solution. Note that the truck is moving at a constant velocity.

Initialization

On the other hand, when we initialize with defensive humans – our heuristic for that is to have the person slow down – this is enough for the robot’s best response to be going in front of the person. Then, the person’s best response to that adjusts their speed to actually go a bit faster, the robot responds, etc. The solution reached like this is a successful merge in which the human slows down.

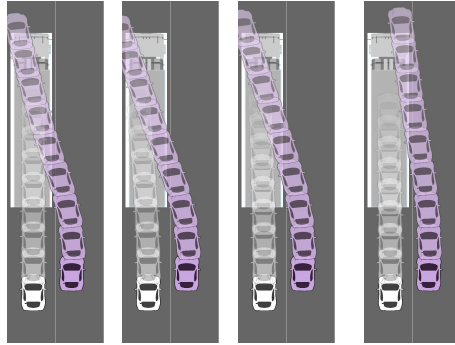


Figure 4.4: The progression of CA-Ign over time in Gap Opening (note the truck is moving at constant velocity). We can see that even though the initialization plans are near collision, CA-Ign slowly adjusts the two plans until the maneuver is much more reasonable. Note that the truck moves at constant velocity.

The “Ign” initialization for both IBR and CA is more interesting, and actually different between the two. In IBR, we heuristically initialize a defensive robot to ignore the human¹. However, we find that this heuristic is a lot tougher to get consistent results from. Because some rewards involve being in front of the robot, ignoring the human is not as simple as removing it, thus our robot’s initial merge plan was sometimes slower or faster than expected. This ultimately makes this more procedural initialization both perform worse than IBR-Def and more inconsistently. In contrast, CA is not restricted to the human best responding to the robot’s aggressive maneuver. We initialize the human to be defensive (heavily slow down) as well as the robot to ignore the human, seeming to work effectively with the robot initialization. Often, this could be a colliding pair of plans! It seems like an advantage of CA is this flexibility of initialization, as it can take this pair and slowly de-collide them as shown in Fig 4.4. Because the situation already starts with the robot merging and the human slowing down, this results in the robot making the maneuver earliest and accumulating the most reward.

Optimism Leads to Better Performance

Overall, the pattern we found is that via the way they solve the game and via initialization, the planners are making more or less optimistic assumptions about human behavior, and that this in turn affects how well they can do with humans that match the assumption. The more optimistic we are about the human’s plan – the more we can bully the human – the better we do!

¹Ignore doesn’t actually remove the human completely from R_R . We actually optimized R_R assuming the human’s h position was extremely negative so it be incentivized to merge and get reward for being ‘in front’ of the human.

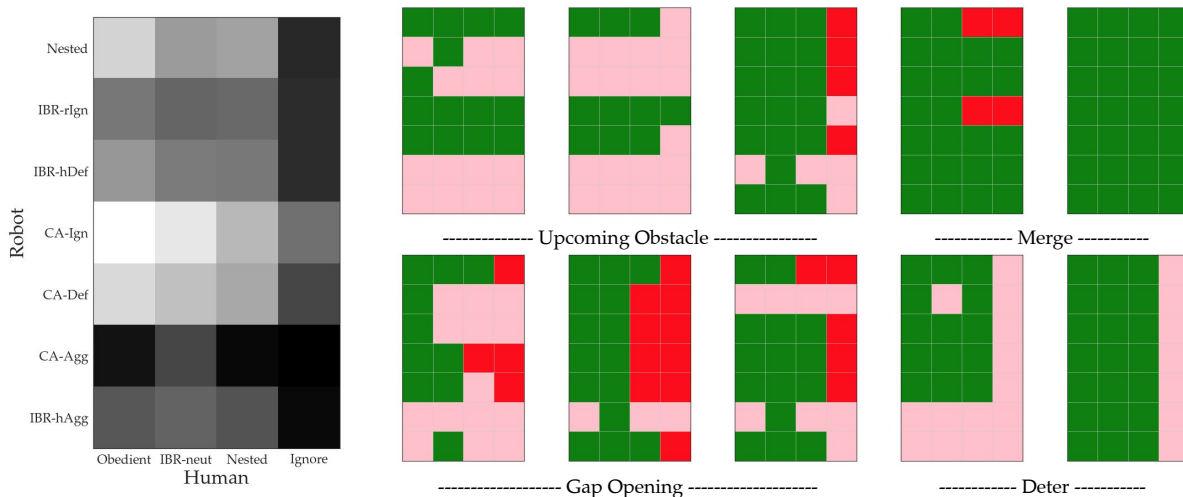


Figure 4.5: **Left:** The mean of the normalized performance in reward of each robot planner against each human planner. We normalized each response over tasks, by subtracting the minimum response and dividing by the difference between the maximum and minimum performance on that task. As the human becomes more and more aggressive (rightwards in the figure), the robot loses more and more reward, as evident by the negative gradient from left to right (barring CA-Agg). **Right:** The outcomes of the maneuver. Green for *success*, Pink for *failure*, and Red for a *collision*. We can see that as we move from left to right in a particular grid we generally transition from green to pink or green to red.

For instance, Nested is more optimistic than IBR in that it assumes the human will compute a best response rather than try to influence the robot. The defensive initializations are more optimistic than the aggressive ones because they imbue in the final equilibrium reached a notion of the person reacting to the robot and making space.

Although quantifying optimism is challenging, we created one proxy for it: the amount of deviation of the assumed human plans from the *other* human plans we introduced in our design: the IBR-Neutral, the Nested human who tries to influence the robot, and the Ignore human who simply ignores the robot. All these other human planners are much more aggressive, so by measuring how much the assumption deviates from them on average, we can capture to what extent the assumption is optimistic. This is essentially $v_H^t - \hat{v}_H^t$, which is *positive* when we assume the human is defensive (slow), and *negative* when we assume the human is aggressive.

From Fig 4.1, we can see that indeed optimism seems to lead to higher reward: as our planner deviates more from our non-obedient humans (i.e. assuming the human accelerate less than they actually do), it gains more reward during the obedient rollouts. When a robot predicts the human will act defensively, and it actually does, this gives the robot more opportunity to perform a successful maneuver.

Summary

Overall, we found Coordinate Ascent when properly initialized to be very aggressive, in large part because it is taking smaller steps than IBR and can thus be more influenced by a defensive human and aggressive robot initialization. One challenge for CA, however, is how to reliably find such initializations autonomously. The Nested optimizer also performs well, as expected when tested with humans that follow its predictions. The final test for all the algorithms is their performance with “non-obedient” humans who do not conveniently play by their assumptions.

4.2 When the Robot’s Assumptions are Incorrect

We now analyze the algorithms’ performance with increasingly aggressive human behavior, from IBR-Neutral to Nested to Ignoring the robot altogether.

Reward

From Fig 4.5(left), we can see that as the humans become more and more aggressive, the robot loses more and more reward: as we move to the right on every algorithm, the cells get darker, meaning lower reward for the robot. In the merge scenarios for instance, this happens because the robot expects to be able to complete the merge, but the person does not slow down as expected. In general, the better an algorithm performed under correct assumptions, the worse it degrades with more aggressive humans. An exception is CA-Ign and CA-Def, which degrade less: here, they still complete relatively dangerous maneuvers, but do so early such that the total reward is higher. It’s also interesting to note that CA-Agg assumes such an aggressive human and performs so poorly with Obedient human, that it actually improves against IBR-neutral

Success

The human becoming more aggressive naturally affects reward, but it also qualitatively affects what happens underneath: as Fig 4.5(right) shows, we start seeing more failed maneuvers (pink), and sometimes we start seeing collisions – these are situations in which the robot keeps stubbornly pretending like the person will slow down, and gets itself into a situation where it cannot escape a collision.

From a pure safety perspective, the only algorithm that never collides is CA-Agg, but this is also the algorithm that fails the maneuver most times (our worst performer under correct assumptions). If we do not consider the “Ignore” human, who completely ignores the robot’s existence, and instead are worried in the worst case about the Nested human, who tries to influence the robot, IBR-hAgg never collides, IBR-hDef, IBR-Ign, and CA-Def collide once, Nested collides twice, and CA-Agg collides thrice. All algorithms remain safe with IBR-Neutral.

Note that we do not have a perfect mapping between reward and safety. In fact, sometimes CA-Ign would have higher reward than another algorithm, but be in a collision. This is because the reward functions we used for the robot are inevitably imperfect and can fail to capture our exact preferences between any two trajectories. What we find interesting, however, is that a very aggressive planner like CA-Ign will push the boundaries of this type of misspecification, exposing behavior that has higher reward but is undesirable.

Summary

From a pure reward perspective, CA-Ign remains remarkably good even with aggressive humans. It does however push the boundaries of safety, where CA-Def or Nested seem like a better middle ground.

4.3 Runtime

Using the Theano [18] framework, we implemented CA to step 100,000 times, IBR 50 times, and ran everything on an Intel i7-9700k CPU. Runtime is relatively consistent from task to task, averaging approximately 13.54 seconds per time-step for CA, 3.11 seconds for IBR, and 5.56 seconds for Nested. Nested takes approximately 607.58 seconds to initialize the gradients, while CA and IBR have the same gradient initialization and take only 12.37 seconds². CA takes significantly longer to converge than IBR because the objectives are more of a *moving target*. Compared to IBR, which computes a best response to a static objective and uses second order methods such as L-BFGS to converge quickly, CA is forced to take small gradient steps. Lastly, Nested’s initialization is extremely slow due to the usage of the Hessian term in the gradient and ultimately bottle-necked the horizon in our experiments. Though we did not do so, we would ultimately require all the algorithms to be optimized for use for real-time planning.

²We timed and averaged several runs of each algorithm, which didn’t vary much from task to task

Chapter 5

Discussion

5.1 Takeaways

In this work we compared the different planning algorithms in a one robot, one human driving dynamic game across different initializations, human planners, and tasks. Now we take a step back to try to answer the question that prompted this work: Which planner do we choose and when?

Overall, we found the behavior from Fig 4.1 to be illustrative: with optimistic initializations, CA can be incredibly aggressive, even more so than Nested. But this should be taken with caution: the underlying reason this is happening is that CA is much more local, and has a hard time escaping the aggressive initialization it is in – it is finding merge solutions that have lower reward than even not merging at all because it gets so close to the human, but ends up with higher MPC rollout reward because it does that merge earlier.

In contrast, IBR initialized optimistically manages to always succeed when the human follows its assumptions, and requires a less strong reaction from the human. IBR-hDef is also relatively safe with more aggressive humans compared to Nested or CA. The main challenge with IBR-hDef in practice however is finding the “Def” autonomously – IBR is very sensitive to initialization, with both IBR-rIgn but especially IBR-hAgg performing worse. The human slowing down was a convenient heuristic for merge scenarios, but how can a robot find such an initialization reliably as the situations and the reward functions become more nuanced?

Overall, Nested seems to be reliably aggressive, IBR is effective when we know how to initialize it, and CA will vary wildly from ultra-aggressive to ultra-defensive as a function of how we initialize.

5.2 Limitations and Future Work

This work is merely a step towards understanding game-theoretic planners for human-robot interaction generally, and for autonomous driving scenarios specifically. We are limited by the scenarios we tested and reward we defined, by the human behavior we simulated, and by

the initializations we tried. Without human data, we also do not know how these algorithms would perform in interaction with real people, both on average and as a function of the specific person's driving style, what they are paying attention to, etc. – instead, what we focused on providing is an apples-to-apples comparison of what the planners would do in the exact same situation, also as a function of interacting with increasingly assertive drivers. Finally, we do not consider planners that hold uncertainty over the human's intentions or plans, or that look at multiple humans that the robot is interacting with simultaneously. While all these directions are important, we are excited to have shed a little more light on the differences between these options we have when our robots need to solve dynamic general-sum games in continuous state and action spaces.

Bibliography

- [1] Zeeshan Ali, Atanas A. Popov, and Guy Charles. “Model predictive control with constraints for a nonlinear adaptive cruise control vehicle model in transition manoeuvres”. In: *Vehicle System Dynamics* 51.6 (2013), pp. 943–963. DOI: 10.1080/00423114.2013.777079. eprint: <https://doi.org/10.1080/00423114.2013.777079>. URL: <https://doi.org/10.1080/00423114.2013.777079>.
- [2] Tirthankar Bandyopadhyay et al. “Intention-Aware Motion Planning”. In: *Algorithmic Foundations of Robotics X*. Ed. by Emilio Frazzoli et al. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 475–491. ISBN: 978-3-642-36279-8.
- [3] C. E. Beal and J. C. Gerdes. “Model Predictive Control for Vehicle Stabilization at the Limits of Handling”. In: *IEEE Transactions on Control Systems Technology* 21.4 (July 2013), pp. 1258–1269. ISSN: 2374-0159. DOI: 10.1109/TCST.2012.2200826.
- [4] R.H. Byrd, L. Peihuang, and J. A Nocedal. “limited-memory algorithm for bound-constrained optimization”. In: *Office of Scientific Technical Information Technical Reports*. 1996.
- [5] Ashwin Carvalho et al. “Robust vehicle stability control with an uncertain driver model”. In: July 2013, pp. 440–445. DOI: 10.23919/ECC.2013.6669718.
- [6] J.C. Engwerda. “Computational aspects of the open-loop Nash equilibrium in linear quadratic games”. In: *Journal of Economic Dynamics and Control* 22.8 (1998), pp. 1487–1506. ISSN: 0165-1889. DOI: [https://doi.org/10.1016/S0165-1889\(98\)00023-2](https://doi.org/10.1016/S0165-1889(98)00023-2). URL: <http://www.sciencedirect.com/science/article/pii/S0165188998000232>.
- [7] Jaime F. Fisac et al. *Hierarchical Game-Theoretic Planning for Autonomous Vehicles*. 2018. arXiv: 1810.05766 [cs.RO].
- [8] C. Hermes et al. “Long-term vehicle motion prediction”. In: *2009 IEEE Intelligent Vehicles Symposium*. 2009, pp. 652–657.
- [9] M. A. S. Kamal et al. “Model Predictive Control of Vehicles on Urban Roads for Improved Fuel Economy”. In: *IEEE Transactions on Control Systems Technology* 21.3 (May 2013), pp. 831–841. ISSN: 2374-0159. DOI: 10.1109/TCST.2012.2198478.
- [10] Kris Kitani et al. “Activity Forecasting”. In: Oct. 2012, pp. 201–214. DOI: 10.1007/978-3-642-33765-9_15.

- [11] J. Kong et al. “Kinematic and dynamic vehicle models for autonomous driving control design”. In: *2015 IEEE Intelligent Vehicles Symposium (IV)*. June 2015, pp. 1094–1099. DOI: 10.1109/IVS.2015.7225830.
- [12] Markus Kuderer et al. “Feature-Based Prediction of Trajectories for Socially Compliant Navigation”. In: *Proceedings of Robotics: Science and Systems*. Sydney, Australia, July 2012. DOI: 10.15607/RSS.2012.VIII.025.
- [13] Stéphanie Lefèvre et al. “Driver models for personalised driving assistance”. In: *Vehicle System Dynamics* 53.12 (2015), pp. 1705–1720. DOI: 10.1080/00423114.2015.1062899. eprint: <https://doi.org/10.1080/00423114.2015.1062899>. URL: <https://doi.org/10.1080/00423114.2015.1062899>.
- [14] Manfred Morari and Jay H. Lee. “Model predictive control: past, present and future”. In: *Computers Chemical Engineering*. 1998.
- [15] Martin J. Osborne and Ariel Rubinstein. *A Course in Game Theory*. Vol. 1. MIT Press Books 0262650401. The MIT Press, Feb. 1994. ISBN: ARRAY(0x47fce600). URL: <https://ideas.repec.org/b/mtp/titles/0262650401.html>.
- [16] Dorsa Sadigh et al. “Planning for Autonomous Cars that Leverage Effects on Human Actions”. In: *Robotics: Science and Systems*.
- [17] M. Simaan and J. B. Cruz. “Additional aspects of the Stackelberg strategy in nonzero-sum games”. In: 1972, pp. 183–187.
- [18] Theano Development Team. “Theano: A Python framework for fast computation of mathematical expressions”. In: *arXiv e-prints* abs/1605.02688 (May 2016). URL: <http://arxiv.org/abs/1605.02688>.
- [19] Peter Trautman and Andreas Krause. “Unfreezing the robot: Navigation in dense, interacting crowds”. In: *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems* (2010), pp. 797–803.
- [20] Pauli Virtanen et al. “SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python”. In: *Nature Methods* (2020). DOI: <https://doi.org/10.1038/s41592-019-0686-2>.
- [21] Mingyu Wang et al. “Game Theoretic Planning for Self-Driving Cars in Competitive Scenarios”. In: *Robotics: Science and Systems*. 2019.
- [22] Brian Ziebart et al. “Planning-based Prediction for Pedestrians”. In: Dec. 2009, pp. 3931–3936. DOI: 10.1109/IR0S.2009.5354147.