Distributionally Robust Online Adaptation via Offline Population Synthesis

Aman Sinha[⋆∗]
Trustworthy AI
aman@trustworthy.ai

Matthew O'Kelly[⋆†]
Trustworthy AI
matt@trustworthy.ai

Hongrui Zheng[⋆]
University of Pennsylvania
hongruiz@seas.upenn.edu

Abstract

Balancing performance and safety is crucial to deploying autonomous vehicles in multi-agent environments. In particular, autonomous racing is a domain that penalizes safe but conservative policies, highlighting the need for robust, adaptive strategies. Current approaches either make simplifying assumptions about other agents or lack robust mechanisms for online adaptation. This work makes algorithmic contributions to both challenges. First, to generate a realistic, diverse set of opponents, we develop a novel method for self-play based on replica-exchange Markov chain Monte Carlo. Second, we propose a distributionally robust bandit optimization procedure that adaptively adjusts risk aversion relative to uncertainty in beliefs about opponents behaviors. We rigorously quantify the tradeoffs in performance and robustness when approximating these computations in real-time motion-planning, and we demonstrate our methods experimentally on autonomous vehicles that achieve scaled speeds comparable to Formula One racecars.

1 Introduction

Current autonomous vehicle (AV) technology still struggles in competitive multi-agent scenarios, such as merging onto a highway, where both maximizing performance (negotiating the merge without delay or hesitation) and maintaining safety (avoiding a crash) are important. In this paper, we investigate this tradeoff in the setting of autonomous racing (AR). In AR, an AV must lap a racetrack in the presence of other agents deploying unknown policies. The agent wins if it completes the race faster than its opponents; a crash automatically results in a loss. AR is a competitive multi-agent game, a general setting challenging for a number of reasons. First, failures are expensive and dangerous, so learning-based approaches must avoid such behavior or rely on simulation while training. Second, the agents only partially observe their opponent's state, and these observations do not uniquely determine the opponent's behavior. Finally, the agents must make decisions online; the opponent's strategy is a tightly-held secret and cannot be obtained by collecting data before the competition.

Problem: We frame the AR challenge in the context of robust reinforcement learning. We analyze the system as a partially-observed Markov decision process (POMDP) $(\mathcal{S}, \mathcal{A}, P_{sa}, \mathcal{O}, r, \lambda)$, with state space $\mathcal{S}$, action space $\mathcal{A}$, state-action transition probabilities P_{sa}, observation space $\mathcal{O}$, rewards $r : \mathcal{O} \to \mathbb{R}$, and discount factor λ. Furthermore, we capture uncertainty in behaviors of other agents

[⋆]Equal contribution
[∗]Work done while at Stanford University
[†]Work done while at the University of Pennsylvania
[‡]A more extensive version of this work appears at the Proceedings of *ICML 2020*.

Machine Learning for Autonomous Driving Workshop at the 34th Conference on Neural Information Processing Systems (NeurIPS 2020), Vancouver, Canada.

through an set $\mathcal{P}$ for the state-action transitions. Then the AV's objective is

$$\text{maximize} \inf_{P_{sa} \in \mathcal{P}} \sum_t \lambda^t \mathbb{E}[r(o(t))]. \tag{1}$$

The obvious price of robustness [10] is that a larger ambiguity set ensures a greater degree of safety while sacrificing performance against a particular opponent. If we knew the opponent's behavior, we would need no ambiguity set; equivalently, the ambiguity set would shrink to the nominal state-action transition distribution. Our goal is to automatically trade between performance and robustness as we play against opponents, which breaks down into two challenges: parametrizing the ambiguity set to allow tractable inference and computing the robust cost efficiently online.

Contributions: The paper has three contributions: (i) a novel population-based self-play method to parametrize opponent behaviors, (ii) a provably efficient approach to estimate the ambiguity set and the robust cost online, and (iii) a demonstration of these methods on real autonomous vehicles.

Section 1.1 gives context to our learning problem, including connections to classical control techniques. In Section 2, we describe the first challenge: learning how to parametrize the ambiguity set $\mathcal{P}$. Rather than directly consider the continuous action space of throttle and steering outputs, we synthesize a library of "prototype" opponent behaviors offline using population-based self-play. When racing against a particular opponent, the agent maintains a belief vector $w(t)$ of the opponent's behavior patterns as a categorical distribution over these prototype behaviors. We then parametrize the ambiguity set as a ball around this nominal belief $w(t)$.

The second challenge, presented in Section 3, is an online optimization problem, wherein the agent iteratively updates the ambiguity set (*e.g.* updates $w(t)$) and computes the robust cost of this set. In other words, the agent attempts to learn the opponent's behavior online to maximize its competitive performance. Since this optimization occurs on a moving vehicle with limited computational resources, we provide convergence results that highlight tradeoffs of performance and robustness with respect to these budgets. Finally, Section 4 details the practical implications of the theoretical results, emergent properties of the method, and the experimental performance of our approach.

1.1 Related work

Reinforcement learning (RL) has achieved unprecedented success on classic two-player games [*e.g.* 58], leading to new approaches in partially-observable games with continuous action spaces [2, 9]. The agents optimize expected performance rather than adapt to individual variations in opponent strategy, which can lead to poor performance against particular opponents [5]. In contrast, our method explicitly incorporates adaptivity to opponents.

Robust approaches to RL and control (like this work) explicitly model uncertainty. In RL, this amounts to planning in a robust MDP [47] or a POMDP [30]. Early results (*e.g.* Bagnell et al. [4] and Nilim and El Ghaoui [47]) describe solutions for robust planning in (PO)MDPs with tabular state/action spaces. Recent works [66, 42, 52, 22] describe minimax and adversarial RL frameworks for nonlinear systems and continuous action spaces. Like our approach, these methods fall broadly under the framework of robust optimization. Unlike these works, which consider worst-case planning under a fixed uncertainty distribution, our approach updates the distribution online.

Planning in belief space is closely related to our approach and is well-studied in robotics (see *e.g.* Kochenderfer [36]). Specifically in the AV domain, Galceran et al. [19] and Ding and Shen [15] use a Bayesian approach to plan trajectories for AVs in belief space; like this work, both of these approaches characterize the other agent's behavior in the environment categorically. Also similar to this work, Van Den Berg et al. [68] use a sampled set of goals obtained by planning from other agents' perspectives. The main difference in this work from standard belief-space planning formulations is inspired by recent results from distributionally robust optimization (DRO) in supervised-learning settings [7, 46]. These methods reweight training data to reduce the variance of the training loss [46]. While others apply DRO to episodic RL for training *offline* [60, 61], we reweight the belief *online*.

Our approach is similar to game-theoretic methods for AR and AV decision making that use the standard heuristic of iterated best response. Our work is distinct from previous work, which either assumes that all agents act with respect to the same cost function, simplifying the structure of the game [37, 71]; or, without this simplifying assumption, that uses demonstrations to learn possible sets of policies [55, 73]. In constrast, we learn the set of policies without demonstrations and use DRO to robustly score the AV's plans.

2 Offline population synthesis

The goal of offline population synthesis is to generate a diverse set of competitive agent behaviors. Formally, we would like to sample pairs $(x, \theta) \in \mathcal{X} \times \Theta$ that are both diverse as well as achieve small values for a function $f(x, \theta)$. In our AV application, θ parametrizes a neural network used to sample trajectories to follow, x is a weighting of various cost functions that the vehicle uses to select trajectories from the samples, and f is the simulated lap time. With this motivation, we treat the method in more generality, assuming (as in our application) that while we can differentiate $f(x, \theta)$ with respect to θ, x represents hyperparameters and admits only function evaluations $f(x, \theta)$ rather than first-order developments. The key challenge is that we do not *a priori* know a metric with which to evaluate diversity (*e.g.*, a kernel for a determinantal point process or DPP) nor do we know a base value of f that is deemed acceptable for competitive performance.

We make this problem more tractable via temperature-based Markov-chain Monte Carlo (MCMC) and annealing methods [43, 23, 35, 12, 28, 27]. Our goal is to sample from a Boltzmann distribution $g(x, \theta; \beta(t)) \propto e^{-\beta(t) f(x,\theta)}$, where $\beta(t)$ is an inverse "temperature" parameter that grows (or "anneals") with iterations t. When $\beta(t) = 0$, all configurations (x, θ) are equally likely and all MCMC proposals are accepted; as $\beta(t)$ increases, accepted proposals favor smaller f. Unlike standard hyperparameter optimization methods [8, 29] that aim to find a single near-optimal configuration, our goal is to sample a diverse population of (x, θ) achieving small $f(x, \theta)$. As such, our approach—annealed adaptive population tempering (AAdaPT)—maintains a population of configurations and employs high-exploration proposals based on the classic hit-and-run algorithm [62, 6, 38].

2.1 AAdaPT

AAdaPT builds upon replica-exchange MCMC, also called parallel tempering, which is a standard approach to maintaining a population of configurations [65, 21]. In parallel tempering, one maintains replicas of the system at L different temperatures $\beta_1 \geq \beta_2 ... \geq \beta_L$ (which are predetermined and fixed), defining the density of the total configuration as $\prod_{i=1}^{L} g(x^i, \theta^i; \beta_i)$. The configurations at each level perform standard MCMC steps (also called "vertical" steps) as well as "horizontal" steps wherein particles are swapped between adjacent temperature levels (see Figure 1). Horizontal proposals consist of swapping two configurations in adjacent temperature levels uniformly at random; the proposal is accepted using standard Metropolis-Hastings (MH) criteria [23]. The primary benefit of maintaining parallel configurations is that the configurations at "colder" levels (higher β) can exploit high-exploration moves from "hotter" levels (lower β) which "tunnel" down during horizontal steps [21]. This approach allows for faster mixing times, particularly when parallel MCMC proposals occur concurrently in a distributed computing environment.

Maintaining a population: In AAdaPT (Algorithm 1), we maintain a *population* of D configurations at each separate temperature level. Note that this design always maintains D individuals at the highest performance level (highest β). The overall configuration density is $\prod_{i=1}^{L} \prod_{j=1}^{D} g(x^{i,j}, \theta^{i,j}; \beta_i(t))$. Similar to parallel tempering, horizontal proposals are chosen uniformly at random from configurations at adjacent temperatures (see Appendix A). We get the same computational benefits of fast mixing in distributed computing environments and a greater ability to exploit high-temperature "tunneling" due to the greater number of possible horizontal exchanges between adjacent temperature levels. The benefit of the horizontal steps is even more pronounced in the RL setting as only vertical steps require new evaluations of f (*e.g.* simulations).

High-exploration vertical proposals: Another benefit of maintaining parallel populations is to improve exploration. We further improve exploration by using hit-and-run proposals [62, 6, 38] for the vertical MCMC chains. Namely, from a current point (x, θ) we sample a uniformly random direction $\hat{u}$ and then choose a point uniformly on the segment $\mathcal{X} \cap (\{x + \mathbb{R} \cdot \hat{u}\} \times \{\theta\})$. This approach has several guarantees for efficient mixing [38–40]. Note that in our implementation the MCMC steps are only performed on x, while θ updates occur via SGD (see below).

Algorithm 1 AADAPT

```
input: annealing parameter $\alpha$, vertical steps $V$, horizontal exchange steps $E$, temperature levels $L$, population size $d$, initial samples $\{x^{i,j}, \theta^{i,j}\}_{i\in\{1,L\}}^{j\in\{1,D\}}$, iterations $T$
Evaluate $f(x^{i,j}, \theta^{i,j})$
for $t = 1$ to $T$
    for $j = 1$ to $L$ do anneal $\beta_{L-j+1}(t)$ (problem (2))
    for $k = 1$ to $V$ asynchronously, in parallel
        for each population $i$ asynchronously, in parallel
            Sample $\hat{x}^{i,j}$ according to hit-and-run proposal
            Evaluate $f(\hat{x}^{i,j}, \theta^{i,j})$
            Apply MH criteria to update $x^{i,j}$
            Train $\theta^{i,j}$ via SGD
    for $e = 1$ to $E$ do horizontal swaps (Appendix A)
```

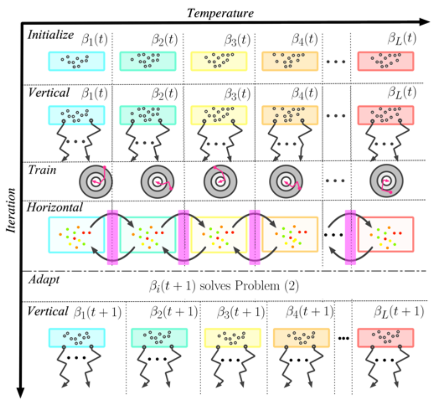


Figure 1. Illustration of AADAPT. Vertical MCMC steps (jagged black arrows) occur in parallel for all $x^{i,j}$, followed by gradient descent steps for trainable parameters $\theta^{i,j}$ (magenta arrows) and horizontal MCMC swaps of configurations between populations (curved black arrows). Then, temperatures $\beta_i(t)$ are updated via problem (2).

Adaptively annealed temperatures: A downside to parallel tempering is the need to determine the temperature levels β_i beforehand. In AADAPT. we adaptively update temperatures. Specifically, we anneal the prescribed horizontal acceptance probability of particle exchanges between temperature levels as $\alpha^{t/(L-1)}$ for a fixed hyperparameter $\alpha \in (0,1)$. Define the empirical acceptance probability of swaps of configurations between levels $i-1$ and i as

$$p_{i-1,i} := \frac{1}{D^2}\sum_{j=1}^{D}\sum_{k=1}^{D}(y_{i-1,i}^{j,k})^{\beta_{i-1}-\beta_i}, \qquad y_{i-1,i}^{j,k} := \min\left(1, e^{f(x^{i-1,j},\theta^{i-1,j})-f(x^{i,k},\theta^{i,k})}\right).$$

Then, at the beginning of each iteration (in which we perform a series of vertical and horizontal MCMC steps), we update the $\beta_i(t)$ sequentially; we fix $\beta_L(t) := \beta_L = 0$ and for a given β_i, we set β_{i-1} by solving the following convex optimization problem:

$$\underset{\{\beta_{i-1}\geq\beta_i,\ p_{i-1,i}\leq\alpha^{\frac{t}{(L-1)}}\}}{\text{minimize}} \quad \beta_{i-1}, \tag{2}$$

using binary search. This adaptive scheme is crucial in our problem setting, where we *a priori* have no knowledge of appropriate scales for f and, as a result, β. In practice, we find that forcing β_i to monotonically increase in t yields better mixing, so we set $\beta_i(t) = \max(\beta_i(t-1), \hat{\beta}_i(t))$, where $\hat{\beta}_i(t)$ solves problem (2).

Evaluating proposals via self-play: In this paper we apply AADAPT to a multi-agent game. It is only possible to evaluate $f(x, \theta)$ in the context of other agents. Since we are interested in the setting where demonstrations from potential opponents are either difficult to obtain or held secret, we iteratively evaluate f via self-play. For each configuration (x, θ), we place two vehicles with the same policy in the simulated environment and perform a race (with $f(x, \theta)$ being the lap time of the agent that starts behind the other). Vertical MCMC steps propose new x, which are then accepted according to MH criteria. After a number of vertical iterations, a stochastic gradient descent (SGD) step is applied to θ (which maximizes the likelihood of the trajectories chosen by the agent with cost functions parametrized by x). Following this process, the updated agents in adjacent temperature levels are exchanged via horizontal MCMC steps. Although we choose $f(x, \theta)$ as the laptime, explicit entropic terms can also be included to further encourage diversity within a single vertical chain or across the population.

At the conclusion of AADAPT, we use the coldest population of D agents at inverse temperature $\beta_1(T)$ to build a DPP sampler. Specifically, define the matrix H via configurations $x^{1,\cdot}$ at the lowest temperature

$$H_{ab} = \|x^{1,a} - x^{1,b}\|. \tag{3}$$

Then we define the DPP kernel K as $K_{ab} = \exp\left(-H_{ab}^2/\sigma^2\right)$ with a scale parameter $\sigma = 0.5$, and we sample $d \leq D$ configurations from this DPP.

3 Online learning with computation budgets

Now we exploit the population of d learned prototype behaviors to enable robust performance. The agent's (our) goal is to act robustly against uncertainty in opponent behaviors and adapt online to a given opponent. We parametrize the agent's (stochastic) policy as follows. At each time step, we sample goal states (consisting of pose and velocity) via a generative model $G(\theta)$ parametrized by θ (as in Section 2). For a given goal state, we compute the parameters of a cubic spline that reaches the goal by solving a nonconvex trajectory optimization problem [44]; on this proposed trajectory we evaluate a collection of cost functions (such as the maximum curvature or minimum acceleration along the path) weighted by the vector x (recall Section 2), similar to Sadat et al. [54] (see Appendix D for a description of all costs). Finally, we choose the sampled goal trajectory with minimum robust cost and perform an action to track this trajectory.

Some of the costs that evaluate the utility of a goal state involve beliefs about the opponent's future trajectory. For a goal p, we rewrite the performance objective at time t with respect to a protoype opponent i as a receding-horizon cost

$$c_i(t;p) := -\sum_{s>t} \lambda^{s-t}\mathbb{E}[r(o(s);p)],$$

where we omit dependence on the agent's cost weights x for convenience. We parametrize the agent's belief of the opponent's behavior as a categorical distribution of beliefs over the prototypes. Specifically, let $w(t) \in \Delta$ be a weight vector at a given time t, where $\Delta := \{a \in \mathbb{R}^d_+ \mid a^T\mathbf{1} = 1\}$, and let $P_0(t) := \text{Categorical}(w(t))$. Then $P_0(t)$ is the nominal distribution describing the agent's belief about the opponent. Furthermore, we consider ambiguity sets $\mathcal{P}(t)$ defined by divergence measures on the space of probability measures over Δ. For a convex function ϕ with $\phi(1) = 0$, the ϕ-divergence between distributions P and Q is $D_\phi(P\|Q) = \int \phi(\frac{dP}{dQ})dQ$. We use sets $\mathcal{P}(t) := \{Q : D_\phi(Q\|P_0)(t) \le \rho\}$ where $\rho > 0$ is a specified constant. Our implementation employs the χ^2-divergence $\phi(t) = t^2 - 1$.

Having defined the ambiguity set $\mathcal{P}(t)$ and the cost with respect to each prototype opponent, we rewrite the robust performance objective (1) to clearly illustrate the optimization problem. Let $C(t;p)$ be a random variable representing the expected cost with respect to the belief of the opponent (and goal state p). Then the robust cost at time t is

$$\sup_{Q\in\mathcal{P}(t)} \mathbb{E}_Q[C(t;p)] = \sup_{q:\sum_i w_i\phi(\frac{q_i}{w_i})\le\rho} \sum_i q_i c_i(t;p). \tag{4}$$

When $\rho = 0$, this is the expected cost under P_0; larger ρ adds robustness. Solving the convex optimization problem (4) first requires computing the costs $c_i(t)$. Using $\lambda \ge 0$ for the constraint $D_\phi(Q\|P_0) \le \rho$, a partial Lagrangian is $\mathcal{L}(q,\lambda) = \sum_i q_i c_i(t) - \lambda\left(\sum_i w_i\phi(q_i/w_i) - \rho\right)$. The corresponding dual function is $v(\lambda) = \sup_{q\in\Delta}\mathcal{L}(q,\lambda)$, and minimizing $v(\lambda)$ via bisection yields the solution to problem (4). Maximizing $\mathcal{L}(q,\lambda)$ with respect to q for a given λ requires $O(d)$ time using a variant of median-based search [16] (see Appendix B). Thus, computing an ϵ-suboptimal solution uses $O(d\log(1/\epsilon))$ time.

The supremum in the robust cost (4) is over belief ambiguity. Thus, our approach generalizes beyond the goal-sampling and trajectory-optimization approach presented at the beginning of this section; it is compatible with any policy that minimizes a cost $c_i(t)$ with respect to a parametrization for opponent i's policy. In this way, it is straightforward to combine our framework with robust model predictive control formulations that have rigorous stability guarantees.

In order to perform competitive actions, the agent updates the ambiguity set $\mathcal{P}(t)$ and computes the robust cost (4) on an embedded processor on board the vehicle in real-time (*e.g.* within 100 milliseconds). In the next two subsections, we describe how to perform both operations in the presence of a severely limited computational budget, and we quantitatively analyze the implications of the budget on the robustness/performance tradeoff.

3.1 Approximating the robust cost

For a large library of prototypical opponents (large d), computing every c_i in the objective (4) is prohibitively expensive. Instead, we consider an empirical approximation of the objective, where we draw N_w indices $J_k \overset{\text{i.i.d.}}{\sim} P_0(t)$ (where $N_w < d$) and consider the weighted sum of these costs

c_{j_k}. Specifically, we define the empirical approximation $\mathcal{P}_{N_w} := \{q : D_\phi\left(q \| \mathbf{1}/N_w\right) \le \rho\}$ to $\mathcal{P}$ and solve the following empirical version of problem (4):

$$\underset{q \in \mathcal{P}_{N_w}}{\operatorname{maximize}} \quad \sum_k q_k c_{j_k}(t;p). \tag{5}$$

This optimization problem (5) makes manifest the price of robustness. Consider the setup of the problem—computing the c_{j_k}. First, we denote the empirical distribution as $\hat{w}(t)$ with $\hat{w}_i(t) = \sum_k^{N_w} \mathbf{1}\{j_k = i\}/N_w$. Even for relatively small N_w/d, $\hat{w}(t)$ concentrates closely around $w(t)$ (see *e.g.* Weissman et al. [72] for a high-probability bound). Thus, when the vehicle's belief about its opponent $w(t)$ is nearly uniform, the j_k values have few repeats. Conversely, when the belief is peaked at a few opponents, the number of unique indices is much smaller than N_w, allowing faster computation of c_{j_k}. The short setup-time enables faster planning or, alternatively, the ability to compute the costs c_{j_k} with longer horizons. Therefore, theoretical performance automatically improves as the vehicle learns about the opponent and the robust evaluation approaches the true cost. A second, more technical, illustration of the price of robustness is described in Appendix B.

3.2 Updating the ambiguity set

To maximize performance against an opponent, the agent updates the ambiguity set $\mathcal{P}$ as the race progresses. Since we consider ϕ-divergence balls of fixed size ρ, this update involves only the nominal belief vector $w(t)$. As with computation of the robust cost, this update must occur efficiently due to time and computational constraints.

For a given sequence of observations of the opponent $o^H_{\text{opp}}(t) := \{o_{\text{opp}}(t), o_{\text{opp}}(t-1), ..., o_{\text{opp}}(t-h+1)\}$ over a horizon h, we define the likelihood of this sequence coming from the i^{th} prototype opponent as $l^h_i(t) = \log d\mathbb{P}\left(o^h_{\text{opp}}(t) | G(\theta^{1,i})\right)$, where $G(\theta^{1,i})$ is a generative model of goal states for the i^{th} prototype opponent. Letting $\bar{l}$ be a uniform upper bound on $l^h_i(t)$, we define the losses $L_i(t) := 1 - l^h_i(t)/\bar{l}$. If we had enough time/computation budget, we could compute $L_i(t)$ for all prototype opponents i and perform an online mirror descent update with an entropic Bregman divergence [57]. In a resource-constrained setting, we can only select a few of these losses, so we use EXP3 [3] to update $w(t)$. Unlike a standard adversarial bandit setting, where we pull just one arm (*e.g.*compute a loss $L_i(t)$) at every time step, we may have resources to compute up to N_w losses in parallel at any given time (the same indices J_k discussed in Section 3.1). Denote our unbiased subgradient estimate as $\gamma(t)$:

$$\gamma_i(t) = \frac{1}{N_w} \sum_{k=1}^{N_w} \frac{L_i(t)}{w_i(t)} \mathbf{1}\{J_k = i\}. \tag{6}$$

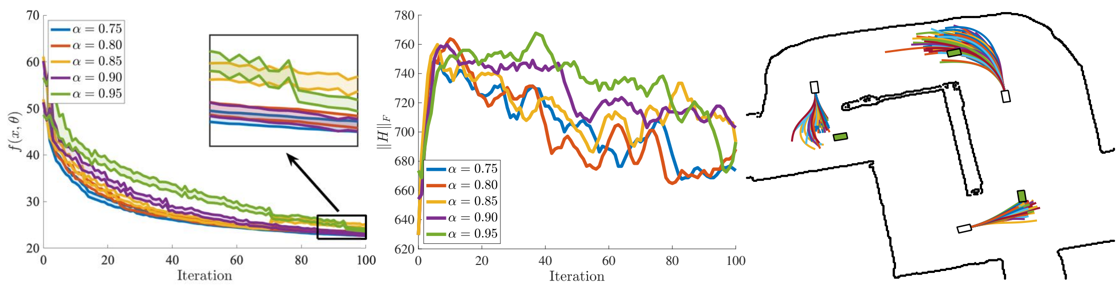
Algorithm 3 in Appendix B describes our modified EXP3 method, which has the following expected regret.

Proposition 1. *Let $z := \frac{d-1}{N_w} + 1$. Algorithm 3 run for T iterations with stepsize $\eta = \sqrt{\frac{2\log(d)}{zT}}$ has expected regret bounded by $\sum_{t=1}^T \mathbb{E}\left[\gamma(t)^T (w(t) - w^\star)\right] \le \sqrt{2zT\log(d)}$.*

See Appendix B for the proof. This regret bound looks similar to that if we simply ran N_w standard EXP3 steps per iteration t (in which case $z = d/N_w$). However, our approach enables parallel computation which is critical in our time-constrained setting. Note that the "multiple-play" setting we propose here has been studied before with better regret bounds but higher computational complexity per iteration [67, 74]. We prefer our approach for its simplicity and ability to be easily combined with the robust-cost computation.

4 Experiments

In this section we first describe the AR environment used to conduct our experiments. Next we explore the hyperparameters of the algorithms in Section 2 and 3, identifying a preferred configuration. Then we consider the overarching hypothesis: online adaptation can improve the performance of robust control strategies. We show the statistically significant results affirming the theory and validate the approach's performance on real vehicles.



(a) Performance vs. iteration (b) Diversity vs. iteration (c) Snapshot trajectories

Figure 2. Hyperparameter selection for AADAPT. (a) 95%-confidence intervals for $f(x, \theta)$ in the coldest temperature level. (b) Frobenius norm of the Mahalanobis distance matrix H (3). The value $\alpha = 0.9$ achieves the best performance and diversity. (c) Qualitative illustrations of multimodal behavior in the learned population of cost functions.

The experiments use an existing low-cost 1/10th-scale, Ackermann-steered AV (Figure 5 in Appendix D). Additionally, we create a simulator and an associated OpenAI Gym API [11] suitable for distributed computing. The simulator supports multiple agents as well as deterministic executions. We experimentally determine the physical parameters of the agent models for simulation and use SLAM to build the virtual track as a mirror of a real location (see videos: `https://youtu.be/lGYm6senwT0`, `https://youtu.be/FHotossCIOc` as well as Appendices C and D for details).

The agent software uses a hierarchical planner [20] similar to that of Ferguson et al. [17]. The key difference is the use of a masked autoregressive flow (MAF) [53] which provides the generative model for goal states, $G(\theta)$. Belief inference and robust cost computation require sampling and evaluating the likelihood of goal states. MAFs can evaluate likelihoods quickly but generate samples slowly. Inspired by Oord et al. [50] we overcome this inefficency by training a "student" inverse autogressive flow (IAF) [34] on MAF samples. Given a sample of goals from the IAF, the agent synthesizes dynamically feasible trajectories following McNaughton [44]. Each sample is evaluated according to Equation 4; the cost function weights are learned by AADAPT (Appendix D contains formal definitions of the cost components). Belief updates use Algorithm 3, employing the MAF to compute the losses $L_i(t)$.

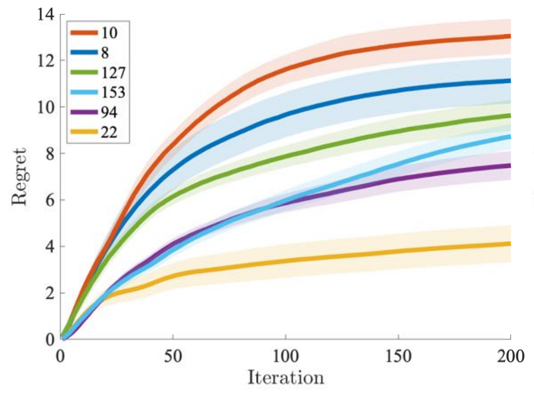
4.1 Offline population synthesis

We run AADAPT with $L = 5$ populations, $D = 160$ configurations per population, and $T = 100$ iterations. For vertical MCMC steps, we randomly sample 16 configuratons per population and perform $V = 2$ iterations of 5 hit-and-run proposals. Furthermore, we perform $E = DL^2/\alpha^{t/(L-1)}$ horizontal steps (motivated by the fact fact that "tunneling" from the highest-temperature level to the coldest takes $O(L^2)$ accepted steps). Finally, for training θ, we use Adam [33] with a learning rate of 10^{-4}.

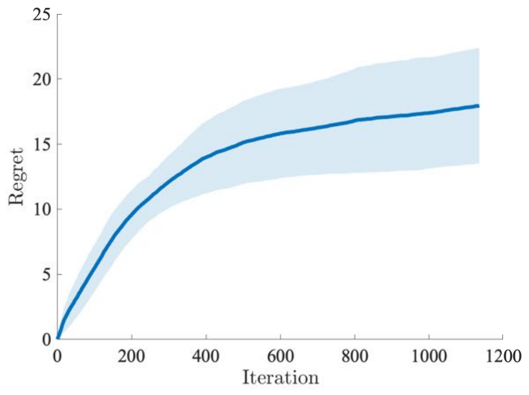
Figure 2 shows results with 5 choices for the most influential hyperparameter, the annealing rate: $\alpha \in \{0.75, 0.80, 0.85, 0.90, 0.95\}$. Figure 2(a) displays 95%-confidence intervals for the mean laptime in the coldest level. The annealing rates $\alpha \in \{0.75, 0.80, 0.90\}$ all result in comparable performance of 22.95 ± 0.14 (mean $\pm$ standard error) seconds at the end of the two-lap run. Figure 2(b) illustrates a metric for measuring diversity, the Frobenius norm of the Mahalanobis distance matrix (3). We see that $\alpha = 0.9$ results in the highest diversity while also attaining the best performance. Thus, in further experimentation, we use the results from the run conducted with $\alpha = 0.9$. Figure 2(c) illustrates qualitative differences between cost functions. Namely, we display the trajectories chosen by all 160 agents in the population at $\beta_1(T)$ at various snapshots along the track. There is a wider spread of behavior near turns than areas where the car simply drives straight.

4.2 Simulated Experiments

We conduct a series of tests in simulation to determine the effects of distributional robustness and adaptivity on overall safety and performance. For a given robustness level $\rho/N_w \in \{0.001, 0.025, 0.2, 0.4, 0.75, 1.0\}$ (with $N_w = 8$ for all experiments), we simulate 40 two-lap races



(a) Simulation



(b) Real

Figure 3. 95%-confidence intervals for regret using $N_w = 8$ arms in (a) simulation and (b) reality. The legend in (a) denotes opponent id and the opponent in (b) has id 22. Our agent has id 33.

against each of the $d = 10$ diverse opponents sampled from the DPP. For fair comparisons, half of the races have the opponent starting on the outside and the other half with the opponent on the inside of the track. Importantly, these experiments involve only the most elite policies from the temperature level $\beta_1(T)$. Since the physical characteristics of the vehicles are identical, win rates between elite policies significantly greater than 0.5 are meaningful. In contrast, against a set of weaker opponents sampled via DPP from the 3rd temperature level $\beta_3(T)$, the win-rate (fraction of races that our agent from the coldest temperature wins) is 0.848 ± 0.012.

Table 1. Robustness effect on aggressiveness

Agent	% of iTTC values < 0.5s
$\rho/N_w = 0.001$	7.86± 0.90
$\rho/N_w = 0.025$	6.46± 0.78
$\rho/N_w = 0.2$	4.75± 0.65
$\rho/N_w = 0.4$	5.41± 0.74
$\rho/N_w = 0.75$	5.50± 0.82
$\rho/N_w = 1.0$	5.76± 0.84

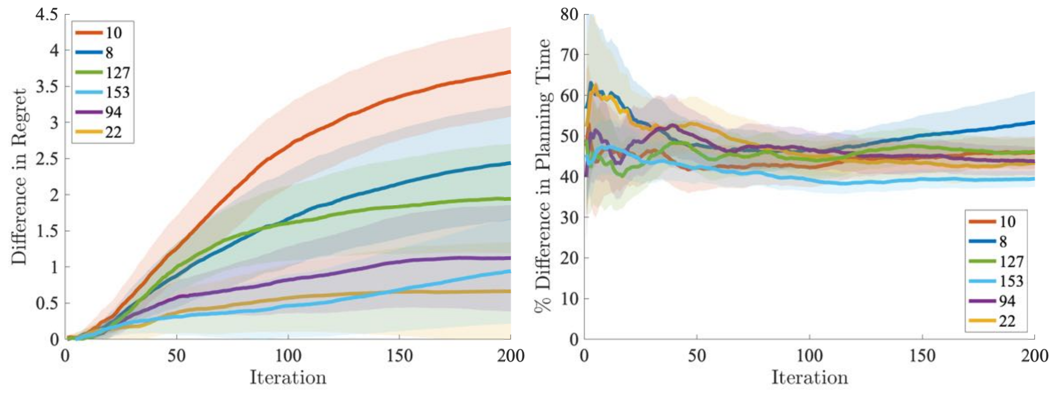
Table 2: The effect of adaptivity on win-rate

Agent	Win-rate Non-adaptive	Win-rate Adaptive	p-value
$\rho/N_w = 0.001$	0.593± 0.025	0.588± 0.025	0.84
$\rho/N_w = 0.025$	0.593± 0.025	0.600± 0.024	0.77
$\rho/N_w = 0.2$	0.538± 0.025	0.588± 0.025	0.045
$\rho/N_w = 0.4$	0.503± 0.025	0.573± 0.025	0.0098
$\rho/N_w = 0.75$	0.513± 0.025	0.593± 0.025	0.0013
$\rho/N_w = 1.0$	0.498± 0.025	0.590± 0.025	0.00024

Effects of distributional robustness We test the hypothesis that distributional robustness results in more conservative policies. For every race both agents have a fixed robustness level ρ and no adaptivity. To measure aggressiveness/conservativeness, we consider instantaneous time-to-collision (iTTC) of the vehicles during the race (see Appendix F). Smaller iTTC values imply more dangerous scenarios and more aggressive policies. In Table 1, we track the rate at which iTTC < 0.5 seconds. As expected, aggressiveness decreases with robustness (the rate of small iTTC values decreases as ρ increases). The trend is $a + b\log(\rho)$, where $a = 5.16 \pm 0.34$ and $b = -0.36 \pm 0.10$ ($R^2 = 0.75$).

Effects of adaptivity Now we investigate the effects of online learning on the outcomes of races. Figure 3(a) shows that Algorithm 3 identifies the opponent vehicle within approximately 150 timesteps (15 seconds), as illustrated by the settling of the regret curve.[1] Given evidence that the opponent model can be identified, we investigate whether adaptivity improves performance, as measured by win-rate. Table 2 displays results of paired t-tests for multiple robustness levels (with a null-hypothesis that adaptivity does not change the win-rate). Each test compares the effect of adaptivity for our agent on the 400 paired trials (and the opponents are always nonadaptive). Adaptivity significantly improves performance for the larger robustness levels $\rho/N_w \geq 0.2$. As hypothesized above, adaptivity automatically increases aggressiveness as the agent learns about its opponent and samples fewer of the other arms to compute the empirical robust cost (5). This effect is more prominent when robustness levels are greater, where adaptivity brings the win-rate back to its level without robustness ($\rho/N_w = 0.001$). Thus, the agent successfully balances safety and performance by combining distributional robustness with adaptivity.

[1]We omit 3 of the regret lines for clarity in the plot.



(a) Difference in regret (b) Difference in planning time

Figure 4. 95%-confidence intervals for the (a) difference in regret and (b) percent difference in cumulative planning time when using sampling approximations vs. online mirror descent. Online mirror descent yields lower regret at the expense of longer planning times.

4.3 Real-world validation

The real world experiments consist of races between agents 22 and 33; we examine the transfer of the opponent modeling approach from simulation to reality. In Figure 3(b) we plot 33's cumulative regret; it takes roughly 4 times as many observations relative to simulation-based experiments to identify the opponent (agent 22). We demonstrate the qualitative properties of the experiments in a video of real rollouts synchronized with corresponding simulations.[2] State estimation error and measurement noise drive the gap between simulated and real performance. First, both vehicle poses are estimated with a particle filter, whereas simulation uses ground-truth states. Since we infer beliefs about an opponent's policy based on a prediction of their actions at a given state, pose estimation error negatively impacts the accuracy of this inference. Second, the simulator only captures the geometry of the track; in reality glass and metal surfaces significantly affect the LIDAR range measurements, which in turn impact the MAF and IAF networks. The convergence of the cumulative regret in Figure 3(b) reflects that, despite the simulation/reality gap, our simulation-trained approach transfers to the real world. Diminishing the effect of the simulation/reality gap is the subject of future work (see Appendix E).

4.4 Approximation analysis

Sampling N_w indices $J_k \overset{\text{i.i.d.}}{\sim} P_0(t)$ allows us to quickly compute the approximate robust cost (Section 3.1) and perform a bandit-style update to the ambiguity set (Section 3.2). Now we analyze the time-accuracy tradeoff of performing this sampling approximation rather than using all d prototypical opponents at every time step. Figure 4(a) shows the difference in regret for the same experiments as in Figure 3(a) if we perform full online mirror-descent updates. Denoting the simulations in Figure 3(a) as S and those with the full mirror descent update as M, we compute difference as $\text{Regret}_S - \text{Regret}_M$. As expected, the difference is positive, since receiving the true gradient is better than the noisy estimate (6). Similarly, Figure 4(b) shows the percent increase in cumulative planning time for the same pairs (sampling vs. full online mirror descent), where percent increase is given by $100(\text{Time}_M - \text{Time}_S)/\text{Time}_S$. As the agent learns who the opponent is, it draws many repeats in the N_w arms, whereas the full mirror descent update always performs d computations. As a result, the percent increase in cumulative iteration time approaches a contant of approximately $1.5\times$. All of these comparisons are done in simulation, where the agent is not constrained to perform actions in under 100 milliseconds. Performing a full mirror descent update is impossible on the real car, as it requires too much time.

[2] `https://youtu.be/FHotossCIOc`

Table 3: The effect of adaptivity on win-rate vs. OOD1

Agent	Win-rate Non-adaptive	Win-rate Adaptive	p-value
$\rho/N_w = 0.001$	0.633±0.036	0.683±0.035	0.280
$\rho/N_w = 1.0$	0.483±0.037	0.717±0.034	5.721E-6

Table 4: The effect of adaptivity on win-rate vs. OOD2

Agent	Win-rate Non-adaptive	Win-rate Adaptive	p-value
$\rho/N_w = 0.001$	0.494±0.037	0.589±0.037	0.059
$\rho/N_w = 1.0$	0.572±0.037	0.739±0.033	0.001

4.5 Out-of-distribution opponents

Now we measure performance against two agents—OOD1 and OOD2—that are not in the distribution developed by our offline population synthesis approach (see Appendix F.2 for details on each agent's policy). We perform only simulated experiments due to the COVID-19 pandemic. For given robustness levels $\rho/N_w \in \{0.001, 1.0\}$ and $N_w = 8$ for all experiments, we perform 180 two-lap races against each of the two human-created racing agents. Again, for fair comparison, half of the experiments have the opponent start on the outside and half on the inside. Tables 3 and 4 show the results. Overall, the trends match those of the in-distribution opponents. Namely, adaptivity significantly increases the win-rate when robustness is high ($\rho/N_w = 1.0$), whereas for low robustness ($\rho/N_w = 0.001$) there is no significant change. Interestingly, adaptivity with robustness not only recovers but surpasses the win-rate of the non-adaptive non-robust policy. We hypothesize that, because out-of-distribution opponents do not match any of the learned prototypes, maintaining an uncertainty over belief automatically helps the agent plan against the "surprising" out-of-distribution actions. Validation of this hypothesis by comparing performance against more out-of-distribution opponents is an interesting direction for future work. Overall, we observe that even against out-of-distribution opponents, we achieve the overall goal of balancing performance and safety.

5 Conclusion

The central hypothesis of this paper is that distributionally robust evaluation of plans relative to the agent's belief state about opponents, which is updated as new observations are made, can lead to policies achieving the same performance as non-robust approaches without sacrificing safety. To evaluate this hypothesis we identify a natural division of the underlying problem. First, we parameterize the set of possible opponents via population-based synthesis without requiring expert demonstrations. Second, we propose an online opponent-modeling framework which enables the application of distributionally robust optimization (DRO) techniques under computational constraints. We provide strong empirical evidence that distributional robustness combined with adaptivity enables a principled method automatically trading between safety and performance. Also, we demonstrate the transfer of our methods from simulation to real autonomous racecars. The addition of recursive feasibility arguments for stronger safety guarantees could improve the applicability of these techniques to real-world settings. Furthermore, although autonomous racing is the current focus of our experiments, future work should explore the generality of our approach in other settings such as human-robot interaction.

References

[1] M. Althoff, M. Koschi, and S. Manzinger. Commonroad: Composable benchmarks for motion planning on roads. In *2017 IEEE Intelligent Vehicles Symposium (IV)*, pages 719–726. IEEE, 2017.

[2] K. Arulkumaran, A. Cully, and J. Togelius. Alphastar: An evolutionary computation perspective. *arXiv preprint arXiv:1902.01724*, 2019.

[3] P. Auer, N. Cesa-Bianchi, Y. Freund, and R. E. Schapire. The nonstochastic multiarmed bandit problem. *SIAM journal on computing*, 32(1):48–77, 2002.

[4] J. A. Bagnell, A. Y. Ng, and J. G. Schneider. Solving uncertain markov decision processes. 2001.

[5] T. Bansal, J. Pachocki, S. Sidor, I. Sutskever, and I. Mordatch. Emergent complexity via multi-agent competition. *arXiv preprint arXiv:1710.03748*, 2017.

[6] C. J. Bélisle, H. E. Romeijn, and R. L. Smith. Hit-and-run algorithms for generating multivariate distributions. *Mathematics of Operations Research*, 18(2):255–266, 1993.

[7] A. Ben-Tal, D. den Hertog, A. D. Waegenaere, B. Melenberg, and G. Rennen. Robust solutions of optimization problems affected by uncertain probabilities. *Management Science*, 59(2):341–357, 2013.

[8] J. Bergstra and Y. Bengio. Random search for hyper-parameter optimization. *Journal of machine learning research*, 13(Feb):281–305, 2012.

[9] C. Berner, G. Brockman, B. Chan, V. Cheung, P. Dkebiak, C. Dennison, D. Farhi, Q. Fischer, S. Hashme, C. Hesse, et al. Dota 2 with large scale deep reinforcement learning. *arXiv preprint arXiv:1912.06680*, 2019.

[10] D. Bertsimas and M. Sim. The price of robustness. *Operations research*, 52(1):35–53, 2004.

[11] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba. Openai gym, 2016.

[12] V. Černỳ. Thermodynamical approach to the traveling salesman problem: An efficient simulation algorithm. *Journal of optimization theory and applications*, 45(1):41–51, 1985.

[13] R. C. Coulter. Implementation of the pure pursuit path tracking algorithm. Technical report, Carnegie-Mellon UNIV Pittsburgh PA Robotics INST, 1992.

[14] J. Dean and S. Ghemawat. Mapreduce: simplified data processing on large clusters. *Communications of the ACM*, 51(1):107–113, 2008.

[15] W. Ding and S. Shen. Online vehicle trajectory prediction using policy anticipation network and optimization-based context reasoning. *arXiv preprint arXiv:1903.00847*, 2019.

[16] J. Duchi, S. Shalev-Shwartz, Y. Singer, and T. Chandra. Efficient projections onto thel1-ball for learning in high dimensions. *Proceedings of the 25th international conference on Machine learning - ICML '08*, 2008. doi: 10.1145/1390156.1390191. URL `http://dx.doi.org/10.1145/1390156.1390191`.

[17] D. Ferguson, T. M. Howard, and M. Likhachev. Motion planning in urban environments. *Journal of Field Robotics*, 25(11-12):939–960, 2008.

[18] S. Fujimoto, H. Van Hoof, and D. Meger. Addressing function approximation error in actor-critic methods. *arXiv preprint arXiv:1802.09477*, 2018.

[19] E. Galceran, A. G. Cunningham, R. M. Eustice, and E. Olson. Multipolicy decision-making for autonomous driving via changepoint-based behavior prediction. In *Robotics: Science and Systems*, volume 1, 2015.

[20] E. Gat, R. P. Bonnasso, R. Murphy, et al. On three-layer architectures. *Artificial intelligence and mobile robots*, 195:210, 1998.

[21] C. J. Geyer. Markov chain monte carlo maximum likelihood. 1991.

[22] A. Gleave, M. Dennis, N. Kant, C. Wild, S. Levine, and S. Russell. Adversarial policies: Attacking deep reinforcement learning. *arXiv preprint arXiv:1905.10615*, 2019.

[23] W. K. Hastings. Monte carlo sampling methods using markov chains and their applications. 1970.

[24] W. Hess, D. Kohler, H. Rapp, and D. Andor. Real-time loop closure in 2d lidar slam. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1271–1278. IEEE, 2016.

[25] P. Hintjens. *ZeroMQ: messaging for many applications*. " O'Reilly Media, Inc.", 2013.

[26] T. M. Howard. *Adaptive model-predictive motion planning for navigation in complex environments*. Carnegie Mellon University, 2009.

[27] J. Hu and P. Hu. Annealing adaptive search, cross-entropy, and stochastic approximation in global optimization. *Naval Research Logistics (NRL)*, 58(5):457–477, 2011.

[28] L. Ingber. Simulated annealing: Practice versus theory. *Mathematical and computer modelling*, 18(11):29–57, 1993.

[29] M. Jaderberg, V. Dalibard, S. Osindero, W. M. Czarnecki, J. Donahue, A. Razavi, O. Vinyals, T. Green, I. Dunning, K. Simonyan, et al. Population based training of neural networks. *arXiv preprint arXiv:1711.09846*, 2017.

[30] L. P. Kaelbling, M. L. Littman, and A. R. Cassandra. Planning and acting in partially observable stochastic domains. *Artificial intelligence*, 101(1-2):99–134, 1998.

[31] S. Karaman and E. Frazzoli. Sampling-based algorithms for optimal motion planning. *The international journal of robotics research*, 30(7):846–894, 2011.

[32] A. Kelly and B. Nagy. Reactive nonholonomic trajectory generation via parametric optimal control. *The International Journal of Robotics Research*, 22(7-8):583–601, 2003.

[33] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[34] D. P. Kingma, T. Salimans, R. Jozefowicz, X. Chen, I. Sutskever, and M. Welling. Improved variational inference with inverse autoregressive flow. In *Advances in neural information processing systems*, pages 4743–4751, 2016.

[35] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. Optimization by simulated annealing. *science*, 220(4598):671–680, 1983.

[36] M. J. Kochenderfer. *Decision making under uncertainty: theory and application*. MIT press, 2015.

[37] A. Liniger and J. Lygeros. A noncooperative game approach to autonomous racing. *IEEE Transactions on Control Systems Technology*, 2019.

[38] L. Lovász. Hit-and-run mixes fast. *Mathematical Programming*, 86(3):443–461, 1999.

[39] L. Lovász and S. Vempala. Hit-and-run is fast and fun. *preprint, Microsoft Research*, 2003.

[40] L. Lovász and S. Vempala. Hit-and-run from a corner. *SIAM Journal on Computing*, 35(4): 985–1005, 2006.

[41] D. Luenberger. *Optimization by Vector Space Methods*. Wiley, 1969.

[42] A. Mandlekar, Y. Zhu, A. Garg, L. Fei-Fei, and S. Savarese. Adversarially robust policy learning: Active construction of physically-plausible perturbations. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3932–3939. IEEE, 2017.

[43] J. Matyas. Random optimization. *Automation and Remote control*, 26(2):246–253, 1965.

[44] M. McNaughton. Parallel algorithms for real-time motion planning. 2011.

[45] B. Nagy and A. Kelly. Trajectory generation for car-like robots using cubic curvature polynomials. *Field and Service Robots*, 11, 2001.

[46] H. Namkoong and J. C. Duchi. Variance regularization with convex objectives. In *Advances in Neural Information Processing Systems 30*, 2017.

[47] A. Nilim and L. El Ghaoui. Robust control of markov decision processes with uncertain transition matrices. *Operations Research*, 53(5):780–798, 2005.

[48] J. Norden, M. O'Kelly, and A. Sinha. Efficient black-box assessment of autonomous vehicle safety. *arXiv preprint arXiv:1912.03618*, 2019.

[49] M. O'Kelly, H. Zheng, J. Auckley, A. Jain, K. Luong, and R. Mangharam. Technical Report: TunerCar: A Superoptimization Toolchain for Autonomous Racing. Technical Report UPenn-ESE-09-15, University of Pennsylvania, September 2019. `https://repository.upenn.edu/mlab_papers/122/`.

[50] A. Oord, Y. Li, I. Babuschkin, K. Simonyan, O. Vinyals, K. Kavukcuoglu, G. Driessche, E. Lockhart, L. Cobo, F. Stimberg, et al. Parallel wavenet: Fast high-fidelity speech synthesis. In *International Conference on Machine Learning*, pages 3918–3926, 2018.

[51] G. Papamakarios, T. Pavlakou, and I. Murray. Masked autoregressive flow for density estimation. In *Advances in Neural Information Processing Systems*, pages 2338–2347, 2017.

[52] L. Pinto, J. Davidson, R. Sukthankar, and A. Gupta. Robust adversarial reinforcement learning. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 2817–2826. JMLR. org, 2017.

[53] D. J. Rezende and S. Mohamed. Variational inference with normalizing flows. In *Proceedings of the 32nd International Conference on International Conference on Machine Learning-Volume 37*, pages 1530–1538. JMLR. org, 2015.

[54] A. Sadat, M. Ren, A. Pokrovsky, Y.-C. Lin, E. Yumer, and R. Urtasun. Jointly learnable behavior and trajectory planning for self-driving vehicles. *arXiv preprint arXiv:1910.04586*, 2019.

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

[56] P. Samson. Concentration of measure inequalities for Markov chains and ϕ-mixing processes. *Annals of Probability*, 28(1):416–461, 2000.

[57] S. Shalev-Shwartz et al. Online learning and online convex optimization. *Foundations and Trends® in Machine Learning*, 4(2):107–194, 2012.

[58] D. Silver, T. Hubert, J. Schrittwieser, I. Antonoglou, M. Lai, A. Guez, M. Lanctot, L. Sifre, D. Kumaran, T. Graepel, et al. A general reinforcement learning algorithm that masters chess, shogi, and go through self-play. *Science*, 362(6419):1140–1144, 2018.

[59] A. Sinha and J. C. Duchi. Learning kernels with random features. In *Advances in Neural Information Processing Systems*, pages 1298–1306, 2016.

[60] A. Sinha, H. Namkoong, and J. Duchi. Certifiable distributional robustness with principled adversarial training. In *Proceedings of the Fifth International Conference on Learning Representations*, 2017. arXiv:1710.10571 [cs.LG].

[61] E. Smirnova, E. Dohmatob, and J. Mary. Distributionally robust reinforcement learning. *arXiv preprint arXiv:1902.08708*, 2019.

[62] R. L. Smith. Efficient monte carlo procedures for generating points uniformly distributed over bounded regions. *Operations Research*, 32(6):1296–1308, 1984.

[63] J. M. Snider et al. Automatic steering methods for autonomous automobile path tracking. *Robotics Institute, Pittsburgh, PA, Tech. Rep. CMU-RITR-09-08*, 2009.

[64] S. Sontges, M. Koschi, and M. Althoff. Worst-case analysis of the time-to-react using reachable sets. In *2018 IEEE Intelligent Vehicles Symposium (IV)*, pages 1891–1897. IEEE, 2018.

[65] R. H. Swendsen and J.-S. Wang. Replica monte carlo simulation of spin-glasses. *Physical review letters*, 57(21):2607, 1986.

[66] A. Tamar, S. Mannor, and H. Xu. Scaling up robust mdps using function approximation. In *International Conference on Machine Learning*, pages 181–189, 2014.

[67] T. Uchiya, A. Nakamura, and M. Kudo. Algorithms for adversarial bandit problems with multiple plays. In *International Conference on Algorithmic Learning Theory*, pages 375–389. Springer, 2010.

[68] J. Van Den Berg, P. Abbeel, and K. Goldberg. Lqg-mp: Optimized path planning for robots with motion uncertainty and imperfect state information. *The International Journal of Robotics Research*, 30(7):895–913, 2011.

[69] B. Vedder. Vedder electronic speed controller. URL `https://vesc-project.com/documentation`.

[70] C. Walsh and S. Karaman. Cddt: Fast approximate 2d ray casting for accelerated localization. abs/1705.01167, 2017. URL `http://arxiv.org/abs/1705.01167`.

[71] Z. Wang, R. Spica, and M. Schwager. Game theoretic motion planning for multi-robot racing. In *Distributed Autonomous Robotic Systems*, pages 225–238. Springer, 2019.

[72] T. Weissman, E. Ordentlich, G. Seroussi, S. Verdu, and M. J. Weinberger. Inequalities for the l1 deviation of the empirical distribution. *Hewlett-Packard Labs, Tech. Rep*, 2003.

[73] G. Williams, B. Goldfain, P. Drews, J. M. Rehg, and E. A. Theodorou. Autonomous racing with autorally vehicles and differential games. *arXiv preprint arXiv:1707.04540*, 2017.

[74] D. P. Zhou and C. J. Tomlin. Budget-constrained multi-armed bandits with multiple plays. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

A Offline population synthesis

Here we provide extra details for Section 2.

Horizontal steps Horizontal steps occur as follows. Two random particles are sampled uniformly at random from adjacent temeprature levels. This forms a proposal for the swap, which is then accepted via standard MH acceptance conditions. Because the rest of the particles remain as-is, the acceptance condition reduces to a particualrly simple form (*cf.* Algorithm 2).

Algorithm 2 HORIZONTAL SWAP

```
Sample $i \sim \text{Uniform}(1, 2, \dots, L-1)$.
Sample $j, k \overset{\text{i.i.d.}}{\sim} \text{Uniform}(1, 2, \dots, D)$.
Sample $p \sim \text{Uniform}([0, 1])$
Let $a = \min\left(1, e^{f(x^{i,j}, \theta^{i,j}) - f(x^{i+1,k}, \theta^{i+1,k})}\right)$
if $p < a^{\beta_i - \beta_{i+1}}$
    swap configurations $(x^{i,j}, \theta^{i,j})$ and $(x^{i+1,k}, \theta^{i+1,k})$
```

We ran our experiments on a server with 88 Intel Xeon cores @ 2.20 GHz. Each run of 100 iterations for a given hyperparameter setting α took 20 hours.

B Online robust planning

Here we provide extra details for Section 3.

In Section 3.1, we described a tradeoff between robsustness and performance. The second way we illustrate the price of robustness is by quantifying the quality of the approximation (5) with respect to the number of samples N_w. For shorthand, define the true expected and approximate expected costs for goal p and distributions Q and q respectively as $R(Q; p) := \mathbb{E}_Q[C(t; p)]$, and $\hat{R}(q; p) := \frac{1}{N_w} \sum_{k=1}^{N_w} q_k c_{j_k}(t; p)$. Then, we have the following bound:

Proposition 2 (Approximation quality)**.** *Suppose $C(t; p) \in [-1, 1]$ for all t, p. Let $A_\rho = \frac{2(\rho+1)}{\sqrt{1+\rho}-1}$ and $B_\rho = \sqrt{8(1+\rho)}$. Then with probability at least $1 - \delta$ over the N_w samples $J_k \overset{\text{i.i.d.}}{\sim} P_0$,*

$$\left| \sup_{q \in \mathcal{P}_{N_w}} \hat{R}(q; p) - \sup_{Q \in \mathcal{P}} R(Q; p) \right| \le 4A_\rho \sqrt{\frac{\log(2N_w)}{N_w}} + B_\rho \sqrt{\frac{\log \frac{2}{\delta}}{N_w}}$$

See below for the proof. Intuitively, increasing accuracy of the robust cost requires more samples (larger N_w), which comes at the expense of computation time. Similar to computing the full cost (4), ϵ-optimal solutions require $O(N_u \log(1/\epsilon))$ time for $N_u \le N_w$ unique indices j_k. In our experiments (*cf.* Section 4), most of the computation time involves the setup to compute the N_u costs c_{j_k}.

B.1 Modified EXP3 Algorithm

Here, we provide details for the modified EXP3 algorithm which has multiple arm-pulls per iteration.

Algorithm 3 EXP3 with N_w arm-pulls per iteration

```
Input: Stepsize sequence $\eta_t$, $w(0) := \mathbf{1}/d$, steps $T$
for $t = 0$ to $T - 1$
    Sample $N_w$ indices $J_k \overset{\text{i.i.d.}}{\sim} \text{Categorical}(w(t))$
    Compute $\gamma(t)$ (Equation (6))
    $w_i(t+1) := \frac{w_i(t)\exp(-\eta_t \gamma_i(t))}{\sum_{j=1}^{d} w_j(t)\exp(-\eta_t \gamma_j(t))}$
```

B.2 Solving problem (5)

We can rewrite the constraint $D_\phi(q \| \mathbf{1}/N_w) \le \rho$ as $\|q - \mathbf{1}/N_w\|^2 \le \rho/N_w$. Then, the partial Lagrangian can be written as

$$\mathcal{L}(q, \lambda) = \sum_i q_i c_i(t) - \frac{\lambda}{2}\left(\|q - \mathbf{1}/N_w\|^2 - \rho/N_w\right).$$

By inspection of the right-hand side, we see that, for a given λ, finding $v(\lambda) = \sup_{q \in \Delta} \mathcal{L}(q, \lambda)$ is equivalent to a Euclidean-norm projection of the vector $\mathbf{1}/N_w + c(t)/\lambda$ onto the probability simplex Δ. This latter problem is directly amenable to the methods of Duchi et al. [16].

B.3 Proof of Proposition 2

We redefine notation to suppress dependence of the cost C on other variables and just make explicit the dependence on the random index J. Namely, we let $C : \mathcal{J} \to [-1, 1]$ be a function of the random index J. We consider the convergence of

$$\sup_{Q \in \mathcal{P}_{N_w}} \mathbb{E}_Q[C(J)] \text{ to } \sup_{Q \in \mathcal{P}} \mathbb{E}_Q[C(J)].$$

To ease notation, we hide dependence on J and for a sample $J_1, \ldots, J_{N_w}$ of random vectors J_k, we denote $C_k := C(J_k)$ for shorthand, so that the C_k are bounded independent random variables. Our proof technique is similar in style to that of Sinha and Duchi [59]. We provide proofs for technical lemmas that follow in support of Proposition 2 that are shorter and more suitable for our setting (in particular Lemmas 1 and 3).

Treating $C = (C_1, \ldots, C_{N_w})$ as a vector, the mapping $C \mapsto \sup_{Q \in \mathcal{P}_{N_w}} \mathbb{E}_Q[C]$ is a $\sqrt{\rho+1}/\sqrt{N_w}$-Lipschitz convex function of independent bounded random variables. Indeed, letting $q \in \mathbb{R}_+^{N_w}$ be the empirical probability mass function associated with $Q \in \mathcal{P}_{N_w}$, we have $\frac{1}{N_w}\sum_{i=1}^{N_w}(N_w q_i)^2 \le \rho + 1$ or $\|q\|_2 \le \sqrt{(1+\rho)/N_w}$. Using Samson's sub-Gaussian concentration inequality [56] for Lipschitz convex functions of bounded random variables, we have with probability at least $1 - \delta$ that

$$\sup_{Q \in \mathcal{P}_{N_w}} \mathbb{E}_Q[C] \in \mathbb{E}\left[\sup_{Q \in \mathcal{P}_{N_w}} \mathbb{E}_Q[C]\right] \pm 2\sqrt{2}\sqrt{\frac{(1+\rho)\log\frac{2}{\delta}}{N_w}}. \tag{7}$$

By the containment (7), we only need to consider convergence of

$$\mathbb{E}\left[\sup_{Q \in \mathcal{P}_{N_w}} \mathbb{E}_Q[C]\right] \text{ to } \sup_{Q \in \mathcal{P}} \mathbb{E}_Q[C],$$

which we do with the following lemma.

Lemma 1 (Sinha and Duchi [59]). *Let $Z = (Z_1, \ldots, Z_{N_w})$ be a random vector of independent random variables $Z_i \overset{\text{i.i.d.}}{\sim} P_0$, where $|Z_i| \le M$ with probability 1. Let $C_\rho = \frac{2(\rho+1)}{\sqrt{1+\rho}-1}$. Then*

$$\mathbb{E}\left[\sup_{Q \in \mathcal{P}_{N_w}} \mathbb{E}_Q[Z]\right] \ge \sup_{Q \in \mathcal{P}} \mathbb{E}_Q[Z] - 4C_\rho M\sqrt{\frac{\log(2N_w)}{N_w}}$$

and

$$\mathbb{E}\left[\sup_{Q \in \mathcal{P}_{N_w}} \mathbb{E}_Q[Z]\right] \le \sup_{Q \in \mathcal{P}} \mathbb{E}_Q[Z].$$

See Appendix B.4 for the proof.

Combining Lemma 1 with containment (7) gives the result.

B.4 Proof of Lemma 1

Before beginning the proof, we first state a technical lemma.

Lemma 2 (Ben-Tal et al. [7]). *Let ϕ be any closed convex function with domain $\operatorname{dom}\phi \subset [0, \infty)$, and let $\phi^*(s) = \sup_{t \ge 0}\{ts - \phi(t)\}$ be its conjugate. Then for any distribution P and any function $g : \mathcal{W} \to \mathbb{R}$ we have*

$$\sup_{Q : D_\phi(Q\|P) \le \rho} \int g(w)dQ(w) = \inf_{\lambda \ge 0, \eta}\left\{\lambda \int \phi^*\left(\frac{g(w) - \eta}{\lambda}\right) dP(w) + \rho\lambda + \eta\right\}.$$

See Appendix B.5 for the proof.

We prove the result for general ϕ-divergences $\phi(t) = t^k - 1$, $k \geq 2$. To simplify algebra, we work with a scaled version of the ϕ-divergence: $\phi(t) = \frac{1}{k}(t^k - 1)$, so the scaled population and empirical constraint sets we consider are defined by

$$\mathcal{P} = \left\{ Q : D_\phi\left(Q \| P_0\right) \leq \frac{\rho}{k} \right\} \text{ and } \mathcal{P}_{N_w} := \left\{ q : D_\phi\left(q \| \mathbf{1}/N_w\right) \leq \frac{\rho}{k} \right\}.$$

Then by Lemma 2, we obtain

$$\begin{aligned} \mathbb{E}\left[\sup_{Q \in \mathcal{P}_{N_w}} \mathbb{E}_Q[Z]\right] &= \mathbb{E}_{P_0}\left[\inf_{\lambda \geq 0, \eta} \frac{1}{N_w} \sum_{i=1}^{N_w} \lambda \phi^*\left(\frac{Z_i - \eta}{\lambda}\right) + \eta + \frac{\rho}{k}\lambda\right] \\ &\leq \inf_{\lambda \geq 0, \eta} \mathbb{E}_{P_0}\left[\frac{1}{N_w} \sum_{i=1}^{N_w} \lambda \phi^*\left(\frac{Z_i - \eta}{\lambda}\right) + \eta + \frac{\rho}{k}\lambda\right] \\ &= \inf_{\lambda \geq 0, \eta} \left\{ \mathbb{E}_{P_0}\left[\lambda \phi^*\left(\frac{Z - \eta}{\lambda}\right)\right] + \frac{\rho}{k}\lambda + \eta \right\} \\ &= \sup_{Q \in \mathcal{P}} \mathbb{E}_Q[Z]. \end{aligned}$$

This proves the upper bound in Lemma 1.

Now we focus on the lower bound. For the function $\phi(t) = \frac{1}{k}(t^k - 1)$, we have $\phi^*(s) = \frac{1}{k^*}\left[s\right]_+^{k^*} + \frac{1}{k}$, where $1/k^* + 1/k = 1$, so that the duality result in Lemma 2 gives

$$\sup_{Q \in \mathcal{P}_{N_w}} \mathbb{E}_Q[Z] = \inf_\eta \left\{ (1+\rho)^{1/k} \left(\frac{1}{N_w} \sum_{i=1}^{N_w} [Z_i - \eta]_+^{k^*} \right)^{\frac{1}{k^*}} + \eta \right\}.$$

Because $|Z_i| \leq M$ for all i, we claim that any η minimizing the preceding expression must satisfy

$$\eta \in \left[-\frac{1 + (1+\rho)^{\frac{1}{k^*}}}{(1+\rho)^{\frac{1}{k^*}} - 1}, 1 \right] \cdot M. \tag{8}$$

For convenience, we first define the shorthand

$$S_{N_w}(\eta) := (1+\rho)^{1/k} \left(\frac{1}{N_w} \sum_{i=1}^{N_w} [Z_i - \eta]_+^{k^*} \right)^{\frac{1}{k^*}} + \eta.$$

Then it is clear that $\eta \leq M$, because otherwise we would have $S_{N_w}(\eta) > M \geq \inf_\eta S_{N_w}(\eta)$. Let the lower bound be of the form $\eta = -cM$ for some $c > 1$. Taking derivatives of the objective $S_{N_w}(\eta)$ with respect to η, we have

$$\begin{aligned} S'_{N_w}(\eta) &= 1 - (1+\rho)^{1/k} \frac{\frac{1}{N_w} \sum_{i=1}^{N_w} [Z_i - \eta]_+^{k^*-1}}{\left(\frac{1}{N_w} \sum_{i=1}^{N_w} [Z_i - \eta]_+^{k^*} \right)^{1 - \frac{1}{k^*}}} \\ &\leq 1 - (1+\rho)^{1/k} \left(\frac{(c-1)M}{(c+1)M} \right)^{k^*-1} \\ &= 1 - (1+\rho)^{1/k} \left(\frac{c-1}{c+1} \right)^{k^*-1}. \end{aligned}$$

For any $c > c_{\rho,k} := \frac{(1+\rho)^{\frac{1}{k^*}} + 1}{(1+\rho)^{\frac{1}{k^*}} - 1}$, the preceding display is negative, so we must have $\eta \geq -c_{\rho,k}M$. For the remainder of the proof, we thus define the interval

$$U := \left[-Mc_{\rho,k}, M\right], \ c_{\rho,k} = \frac{(1+\rho)^{\frac{1}{k^*}} + 1}{(1+\rho)^{\frac{1}{k^*}} - 1},$$

and we assume w.l.o.g. that $\eta \in U$.

Again applying the duality result of Lemma 2, we have that

$$\begin{aligned}\mathbb{E}\left[\sup_{Q\in\mathcal{P}_{N_w}}\mathbb{E}_Q[Z]\right] &= \mathbb{E}\left[\inf_{\eta\in U} S_{N_w}(\eta\right] \\ &= \mathbb{E}\left[\inf_{\eta\in U}\{S_{N_w}(\eta) - \mathbb{E}[S_{N_w}(\eta)] + \mathbb{E}[S_{N_w}(\eta)]\}\right] \\ &\geq \inf_{\eta\in U}\mathbb{E}[S_{N_w}(\eta)] - \mathbb{E}\left[\sup_{\eta\in U}|S_{N_w}(\eta) - \mathbb{E}[S_{N_w}(\eta)]|\right]. \qquad (9)\end{aligned}$$

To bound the first term in expression (9), we use the following lemma.

Lemma 3 (Sinha and Duchi [59]). *Let $Z \geq 0, Z \not\equiv 0$ be a random variable with finite $2p$-th moment for $1 \leq p \leq 2$. Then we have the following inequality:*

$$\mathbb{E}\left[\left(\frac{1}{n}\sum_{i=1}^{n} Z_i^p\right)^{\frac{1}{p}}\right] \geq \|Z\|_p - \frac{p-1}{p}\sqrt{\frac{2}{n}}\sqrt{\mathrm{Var}(Z^p/\mathbb{E}[Z^p])}\|Z\|_2, \qquad (10a)$$

and if $\|Z\|_\infty \leq C$, then

$$\mathbb{E}\left[\left(\frac{1}{n}\sum_{i=1}^{n} Z_i^p\right)^{\frac{1}{p}}\right] \geq \|Z\|_p - C\frac{p-1}{p}\sqrt{\frac{2}{n}}. \qquad (10b)$$

See Appendix B.6 for the proof. Now, note that $[Z-\eta]_+ \in [0, 1+c_{\rho,k}]M$ and $(1+\rho)^{1/k}(1+c_{\rho,k}) =: C_{\rho,k}$. Thus, by Lemma 3 we obtain that

$$\mathbb{E}[S_{N_w}(\eta)] \geq (1+\rho)^{1/k}\mathbb{E}\left[[Z-\eta]_+^{k^*}\right]^{1/k^*} + \eta - C_{\rho,k}M\frac{k^*-1}{k^*}\sqrt{\frac{2}{N_w}}.$$

Using that $\frac{k^*-1}{k^*} = \frac{1}{k}$, taking the infimum over η on the right hand side and using duality yields

$$\inf_{\eta}\mathbb{E}[S_{N_w}(\eta)] \geq \sup_{Q\in\mathcal{P}}\mathbb{E}_Q[Z] - C_{\rho,k}\frac{M}{k}\sqrt{\frac{2}{N_w}}.$$

To bound the second term in expression (9), we use concentration results for Lipschitz functions. First, the function $\eta \mapsto S_{N_w}(\eta)$ is $\sqrt{1+\rho}$-Lipschitz in η. To see this, note that for $1 \leq k^\star \leq 2$ and $X \geq 0$, by Jensen's inequality,

$$\frac{\mathbb{E}[X^{k^\star-1}]}{(\mathbb{E}[X^{k^\star}])^{1-1/k^\star}} \leq \frac{\mathbb{E}[X]^{k^\star-1}}{(\mathbb{E}[X^{k^\star}])^{1-1/k^\star}} \leq \frac{\mathbb{E}[X]^{k^\star-1}}{\mathbb{E}[X]^{k^\star-1}} = 1,$$

so $S'_{N_w}(\eta) \in [1-(1+\rho)^{\frac{1}{k}}, 1]$ and therefore S_{N_w} is $(1+\rho)^{1/k}$-Lipschitz in η. Furthermore, the mapping $T : z \mapsto (1+\rho)^{\frac{1}{k}}(\frac{1}{N_w}\sum_{i=1}^{N_w}[z_i-\eta]_+^{k^*})^{\frac{1}{k^*}}$ for $z \in \mathbb{R}^{N_w}$ is convex and $(1+\rho)^{\frac{1}{k}}/\sqrt{N_w}$-Lipschitz. This is verified by the following:

$$\begin{aligned}|T(z) - T(z')| &\leq (1+\rho)^{1/k}\left|\left(\frac{1}{N_w}\sum_{i=1}^{N_w}\left|[z_i-\eta]_+ - [z_i'-\eta]_+\right|^{k^*}\right)^{\frac{1}{k^*}}\right| \\ &\leq \frac{(1+\rho)^{1/k}}{{N_w}^{1/k^*}}\left|\left(\sum_{i=1}^{N_w}|z_i - z_i'|^{k^*}\right)^{\frac{1}{k^*}}\right| \\ &\leq \frac{(1+\rho)^{1/k}}{\sqrt{N_w}}\|z-z'\|_2,\end{aligned}$$

where the first inequality is Minkowski's inequality and the third inequality follows from the fact that for any vector $x \in \mathbb{R}^n$, we have $\|x\|_p \leq n^{\frac{2-p}{2p}}\|x\|_2$ for $p \in [1,2]$, where these denote the usual vector norms. Thus, the mapping $Z \mapsto S_{N_w}(\eta)$ is $(1+\rho)^{1/k}/\sqrt{N_w}$-Lipschitz continuous with respect to the ℓ_2-norm on Z. Using Samson's sub-Gaussian concentration result for convex Lipschitz functions, we have

$$\mathbb{P}\left(|S_{N_w}(\eta) - \mathbb{E}[S_{N_w}(\eta)]| \geq \delta\right) \leq 2\exp\left(-\frac{N_w\delta^2}{2C_{\rho,k}^2M^2}\right)$$

for any fixed $\eta \in \mathbb{R}$ and any $\delta \geq 0$. Now, let $\mathcal{N}(U,\epsilon) = \{\eta_1, \ldots, \eta_{N(U,\epsilon)}\}$ be an ϵ cover of the set U, which we may take to have size at most $N(U,\epsilon) \leq M(1+c_{\rho,k})\frac{1}{\epsilon}$. Then we have

$$\sup_{\eta\in U}|S_{N_w}(\eta) - \mathbb{E}[S_{N_w}(\eta)] \leq \max_{i\in\mathcal{N}(U,\epsilon)}|S_{N_w}(\eta_i) - \mathbb{E}[S_{N_w}(\eta_i)]| + \epsilon(1+\rho)^{1/k}.$$

Using the fact that $\mathbb{E}[\max_{i\le n}|X_i|] \le \sqrt{2\sigma^2\log(2n)}$ for X_i all σ^2-sub-Gaussian, we have

$$\mathbb{E}\left[\max_{i\in\mathcal{N}(U,\epsilon)} |S_{N_w}(\eta_i) - \mathbb{E}[S_{N_w}(\eta_i)]|\right] \le C_{\rho,k}\sqrt{2\frac{M^2}{N_w}\log 2N(U,\epsilon)}.$$

Taking $\epsilon = M(1+c_{\rho,k})/N_w$ gives that

$$\mathbb{E}\left[\sup_{\eta\in U} |S_{N_w}(\eta) - \mathbb{E}[S_{N_w}(\eta)]|\right] \le \sqrt{2}MC_{\rho,k}\sqrt{\frac{1}{N_w}\log(2N_w)} + \frac{C_{\rho,k}M}{N_w}.$$

Then, in total we have (using $C_\rho \ge C_{\rho,k}$, $k\ge 2$, and $N_w \ge 1$),

$$\begin{aligned}\mathbb{E}\left[\sup_{Q\in\mathcal{P}_{N_w}} \mathbb{E}_Q[Z]\right] &\ge \sup_{Q\in\mathcal{P}} \mathbb{E}_Q[Z] - \frac{C_\rho M\sqrt{2}}{\sqrt{N_w}}\left(\frac{1}{k} + \sqrt{\log(2N_w)} + \frac{1}{\sqrt{2N_w}}\right)\\ &\ge \sup_{Q\in\mathcal{P}} \mathbb{E}_Q[Z] - 4C_\rho M\sqrt{\frac{\log(2N_w)}{N_w}}.\end{aligned}$$

This gives the desired result of the lemma.

B.5 Proof of Lemma 2

Let $L\ge 0$ satisfy $L(w) = dQ(w)/dP(w)$, so that L is the likelihood ratio between Q and P. Then we have

$$\begin{aligned}&\sup_{Q:D_\phi(Q\|P)\le\rho}\int g(w)dQ(w) = \sup_{\int\phi(L)dP\le\rho,\mathbb{E}_P[L]=1}\int g(w)L(w)dP(w)\\ &= \sup_{L\ge 0}\inf_{\lambda\ge 0,\eta}\left\{\int g(w)L(w)dP(w) - \lambda\left(\int f(L(w))dP(w) - \rho\right) - \eta\left(\int L(w)dP(w) - 1\right)\right\}\\ &= \inf_{\lambda\ge 0,\eta}\sup_{L\ge 0}\left\{\int g(w)L(w)dP(w) - \lambda\left(\int f(L(w))dP(w) - \rho\right) - \eta\left(\int L(w)dP(w) - 1\right)\right\},\end{aligned}$$

where we have used that strong duality obtains because the problem is strictly feasible in its non-linear constraints (take $L\equiv 1$), so that the extended Slater condition holds [41, Theorem 8.6.1 and Problem 8.7]. Noting that L is simply a positive (but otherwise arbitrary) function, we obtain

$$\begin{aligned}&\sup_{Q:D_\phi(Q\|P)\le\rho}\int g(w)dQ(w)\\ &= \inf_{\lambda\ge 0,\eta}\int\sup_{\ell\ge 0}\{(g(w)-\eta)\ell - \lambda\phi(\ell)\}\,dP(w) + \lambda\rho + \eta\\ &= \inf_{\lambda\ge 0,\eta}\int\lambda\phi^*\left(\frac{g(w)-\eta}{\lambda}\right)dP(w) + \eta + \rho\lambda.\end{aligned}$$

Here we have used that $\phi^*(s) = \sup_{t\ge 0}\{st - \phi(t)\}$ is the conjugate of ϕ and that $\lambda\ge 0$, so that we may take divide and multiply by λ in the supremum calculation.

B.6 Proof of Lemma 3

For $a > 0$, we have

$$\inf_{\lambda\ge 0}\left\{\frac{a^p}{p\lambda^{p-1}} + \lambda\frac{p-1}{p}\right\} = a,$$

(with $\lambda = a$ attaining the infimum), and taking derivatives yields

$$\frac{a^p}{p\lambda^{p-1}} + \lambda\frac{p-1}{p} \ge \frac{a^p}{p\lambda_1^{p-1}} + \lambda_1\frac{p-1}{p} + \frac{p-1}{p}\left(1 - \frac{a^p}{\lambda_1^p}\right)(\lambda - \lambda_1).$$

Using this in the moment expectation, by setting $\lambda_n = \sqrt[p]{\frac{1}{n}\sum_{i=1}^n Z_i^p}$, we have for any $\lambda\ge 0$ that

$$\begin{aligned}\mathbb{E}\left[\left(\frac{1}{n}\sum_{i=1}^n Z_i^p\right)^{\frac{1}{p}}\right] &= \mathbb{E}\left[\frac{\sum_{i=1}^n Z_i^p}{pn\lambda_n^{p-1}} + \lambda_n\frac{p-1}{p}\right]\\ &\ge \mathbb{E}\left[\frac{\sum_{i=1}^n Z_i^p}{pn\lambda^{p-1}} + \lambda\frac{p-1}{p}\right] + \frac{p-1}{p}\mathbb{E}\left[\left(1 - \frac{\sum_{i=1}^n Z_i^p}{n\lambda^p}\right)(\lambda_n - \lambda)\right].\end{aligned}$$

Now we take $\lambda = \|Z\|_p$, and we apply the Cauchy-Schwarz inequality to obtain

$$\begin{aligned}
\mathbb{E}\left[\left(\frac{1}{n}\sum_{i=1}^n Z_i^p\right)^{\frac{1}{p}}\right] &\geq \|Z\|_p - \frac{p-1}{p}\mathbb{E}\left[\left(1 - \frac{\frac{1}{n}\sum_{i=1}^n Z_i^p}{\|Z\|_p^p}\right)^2\right]^{\frac{1}{2}} \mathbb{E}\left[\left(\left(\frac{1}{n}\sum_{i=1}^n Z_i^p\right)^{\frac{1}{p}} - \|Z\|_p\right)^2\right]^{\frac{1}{2}} \\
&= \|Z\|_p - \frac{p-1}{p\sqrt{n}}\sqrt{\mathrm{Var}(Z^p/\mathbb{E}[Z^p])}\mathbb{E}\left[\left(\left(\frac{1}{n}\sum_{i=1}^n Z_i^p\right)^{\frac{1}{p}} - \mathbb{E}[Z^p]^{\frac{1}{p}}\right)^2\right]^{\frac{1}{2}} \\
&\geq \|Z\|_p - \frac{p-1}{p\sqrt{n}}\sqrt{\mathrm{Var}(Z^p/\mathbb{E}[Z^p])}\mathbb{E}\left[\left(\frac{1}{n}\sum_{i=1}^n Z_i^p\right)^{\frac{2}{p}} + \mathbb{E}[Z^p]^{\frac{2}{p}}\right]^{\frac{1}{2}} \\
&\geq \|Z\|_p - \frac{p-1}{p}\sqrt{\frac{2}{n}}\sqrt{\mathrm{Var}(Z^p/\mathbb{E}[Z^p])}\|Z\|_2,
\end{aligned}$$

where the last inequality follows by the fact that the norm is non-decreasing in p.

In the case that we have the unifom bound $\|Z\|_\infty \leq C$, we can get tighter guarantees. To that end, we state a simple lemma.

Lemma 4. *For any random variable $X \geq 0$ and $a \in [1, 2]$, we have*

$$\mathbb{E}[X^{ak}] \leq \mathbb{E}[X^k]^{2-a}\mathbb{E}[X^{2k}]^{a-1}$$

Proof For $c \in [0, 1]$, $1/p + 1/q = 1$ and $A \geq 0$, we have by Holder's inequality,

$$\mathbb{E}[A] = \mathbb{E}[A^c A^{1-c}] \leq \mathbb{E}[A^{pc}]^{1/p}\mathbb{E}[A^{q(1-c)}]^{1/q}$$

Now take $A := X^{ak}$, $1/p = 2 - a$, $1/q = a - 1$, and $c = \frac{2}{a} - 1$. □

First, note that $\mathbb{E}[Z^{2p}] \leq C^p\mathbb{E}[Z^p]$. For $1 \leq p \leq 2$, we can take $a = 2/p$ in Lemma 4, so that we have

$$E[Z^2] \leq \mathbb{E}[Z^p]^{2-\frac{2}{p}}\mathbb{E}[Z^{2p}]^{\frac{2}{p}-1} \leq \|Z\|_p^p C^{2-p}.$$

Now, we can plug these into the expression above (using $\mathrm{Var}Z^p \leq \mathbb{E}[Z^{2p}] \leq C^p\|Z\|_p^p$), yielding

$$\mathbb{E}\left[\left(\frac{1}{n}\sum_{i=1}^n Z_i^p\right)^{\frac{1}{p}}\right] \geq \|Z\|_p - C\frac{p-1}{p}\sqrt{\frac{2}{n}}$$

as desired.

B.7 Proof of Proposition 1

We utilize the following lemma for regret of online mirror descent.

Lemma 5. *The expected regret for online mirror descent with unbiased stochastic subgradient $\gamma(t)$ and stepsize η is*

$$\sum_{t=1}^T \mathbb{E}\left[\gamma(t)^T(w(t) - w^\star)\right] \leq \frac{\log(d)}{\eta} + \frac{\eta}{2}\mathbb{E}\left[\sum_{t=1}^T\sum_{j=1}^d w_j(t)\gamma_j(t)^2\right] \tag{11}$$

See Appendix B.8 for the proof. Now we bound the right-hand term of the regret bound (11) in our setting. For this we utilize the following:

$$\begin{aligned}
\mathbb{E}\left[\gamma_i(t)^2|w(t)\right] &= \frac{1}{N_w^2}\frac{L_i^2(t)}{w_i^2(t)}\mathbb{E}\left[\left(\sum_{k=1}^{N_w}\mathbf{1}\left\{J_k = i\right\}\right)^2\middle|w(t)\right] \\
&= \frac{1}{N_w^2}\frac{L_i^2(t)}{w_i^2(t)}\left(N_w(N_w - 1)w_i(t)^2 + N_w w_i(t)\right),
\end{aligned}$$

where the latter fact is simply the second moment for the sum of N_w random variables $\overset{\text{i.i.d.}}{\sim}$ Bernoulli$(w_i(t))$. Then,

$$\begin{aligned}\sum_{i=1}^d w_i(t)\mathbb{E}\left[\gamma_i(t)^2|w(t)\right] &= \sum_{i=1}^d L_i(t)^2\left(\frac{N_w-1}{N_w}w_i(t)+\frac{1}{N_w}\right)\\ &\le \sum_{i=1}^d\left(\frac{N_w-1}{N_w}w_i(t)+\frac{1}{N_w}\right)\\ &= \frac{N_w-1}{N_w}+\frac{d}{N_w}\\ &=: z.\end{aligned}$$

Plugging in the prescribed $\eta=\sqrt{\frac{2\log(d)}{zT}}$ into the bound (11) yields the result.

B.8 Proof of Lemma 5

We first show the more general regeret of online mirror descent with a Bregman divergence and then specialize to the entropic regularization case. Let $\psi(w)$ be a convex fuction and $\psi^*(\theta)$ its Fenchel conjugate. Define the Bregman divergence $B_\psi(w,w')=\psi(w)-\psi(w')-\nabla\psi(w')^T(w-w')$. In the following we use the subscript $\cdot_t$ instead of $(\cdot)(t)$ for clarity. The standard online mirror descent learner sets

$$w_t=\operatorname*{argmin}_w\left(\gamma_t^T w+\frac{1}{\eta}B_\psi(w,w_t)\right).$$

Using optimality of w_{t+1} in the preceding equation, we have

$$\begin{aligned}\gamma_t^T(w_t-w^*) &= \gamma_t^T(w_{t+1}-w^*)+\gamma_t^T(w_t-w_{t+1})\\ &\le \frac{1}{\eta}(\nabla\psi(w_{t+1})-\nabla\psi(w_t))^T(w^*-w_{t+1})\\ &+\gamma_t^T(w_t-w_{t+1})\\ &=\frac{1}{\eta}\left(B_\psi(w^*,w_t)-B_\psi(w^*,w_{t+1})-B_\psi(w_{t+1},w_t)\right)\\ &+\gamma_t^T(w_t-w_{t+1}).\end{aligned}$$

Summing this preceding display over iterations t yields

$$\begin{aligned}\sum_{t=1}^T\gamma_t^T(w_t-w^*) \le{}& \frac{1}{\eta}B_\psi(w^*,w_1)\\ &+\sum_{t=1}^T\left(-\frac{1}{\eta}B_\psi(w_{t+1},w_t)+\gamma_t^T(w_t-w_{t+1})\right)\end{aligned}$$

Now let $\psi(w)=\sum_i w_i\log w_i$. Then, with $w_1=\mathbf{1}/d$, $B\psi(w^*,w_1)\le\log(d)$. Now we bound the second term with the following lemma.

Lemma 6. *Let $\psi(x)=\sum_j x_j\log x_j$ and $x,y\in\Delta$ be defined by: $y_i=\frac{x_i\exp(-\eta g_i)}{\sum_j x_j\exp(-\eta g_j)}$ where $g\in\mathbb{R}_+^d$ is non-negative. Then*

$$-\frac{1}{\eta}B_\psi(y,x)+g^T(x-y)\le\frac{\eta}{2}\sum_{i=1}^d g_i^2x_i.$$

See Appendix B.9 for the proof. Setting $y=w_{t+1}$, $x=w_t$, and $g=\gamma_t$ in Lemma 6 yields

$$\sum_{t=1}^T\gamma_t^T(w_t-w^*)\le\frac{\log(d)}{\eta}+\frac{\eta}{2}\sum_{t=1}^T\sum_{j=1}^d w_j(t)\gamma_j(t)^2.$$

Taking expectations on both sides yields the result.

B.9 Proof of Lemma 6

Note that $B_\psi(y,x) = \sum_i y_i \log \frac{y_i}{x_i}$. Substituting the values for x and y into this expression, we have

$$\sum_i y_i \log \frac{y_i}{x_i} = -\eta g^T y - \sum_i y_i \log \left(\sum_j x_j e^{-\eta g_j} \right)$$

Now we use a Taylor expansion of the function $g \mapsto \log \left(\sum_j x_j e^{-\eta g_j} \right)$ around the point 0. If we define the vector $p_i(g) = x_i e^{-\eta g_i} / \left(\sum_j x_j e^{-\eta g_j} \right)$, then

$$\log \left(\sum_j x_j e^{-\eta g_j} \right) = \log(\mathbf{1}^T x) - \eta p(0)^T g + \frac{\eta^2}{2} g^\top \left(\operatorname{diag}(p(\widetilde{g})) - p(\widetilde{g}) p(\widetilde{g})^\top \right) g$$

where $\widetilde{g} = \lambda g$ for some $\lambda \in [0,1]$. Noting that $p(0) = x$ and $\mathbf{1}^T x = \mathbf{1}^T y = 1$, we obtain

$$B_\psi(y,x) = \eta g^T (x-y) - \frac{\eta^2}{2} g^\top \left(\operatorname{diag}(p(\widetilde{g})) - p(\widetilde{g}) p(\widetilde{g})^\top \right) g,$$

whereby

$$-\frac{1}{\eta} B_\psi(y,x) + g^T(x-y) \le \frac{\eta}{2} \sum_{i=1}^d g_i^2 p_i(\widetilde{g}). \tag{12}$$

Lastly, we claim that the function

$$s(\lambda) = \sum_{i=1}^d g_i^2 \frac{x_i e^{-\lambda g_i}}{\sum_j x_j e^{-\lambda g_j}}$$

is non-increasing on $\lambda \in [0,1]$. Indeed, we have

$$\begin{aligned} s'(\lambda) &= \frac{\left(\sum_i g_i x_i e^{-\lambda g_i}\right)\left(\sum_i g_i^2 x_i e^{-\lambda g_i}\right)}{\left(\sum_i x_i e^{-\lambda g_i}\right)^2} - \frac{\sum_i g_i^3 x_i e^{-\lambda g_i}}{\sum_i x_i e^{-\lambda g_i}} \\ &= \frac{\sum_{ij} g_i g_j^2 x_i x_j e^{-\lambda g_i - \lambda g_j} - \sum_{ij} g_i^3 x_i x_j e^{-\lambda g_i - \lambda g_j}}{\left(\sum_i x_i e^{-\lambda g_i}\right)^2} \end{aligned}$$

Using the Fenchel-Young inequality, we have $ab \le \frac{1}{3}|a|^3 + \frac{2}{3}|b|^{3/2}$ for any a, b so $g_i g_j^2 \le \frac{1}{3} g_i^3 + \frac{2}{3} g_j^3$. This implies that the numerator in our expression for $s^{'}(\lambda)$ is non-positive. Thus, $s(\lambda) \le s(0) = \sum_{i=1}^d g_i^2 x_i$ which gives the result when combined with inequality (12).

C Hardware

The major components of the vehicle used in experiments are shown in Figure 5. The chassis of the 1/10-scale vehicles used in experiments are based on a Traxxas Rally 1/10-scale radio-controlled car with an Ackermann steering mechanism. An electronic speed controller based on an open source design [69] controls the RPM of a brushless DC motor and actuates a steering servo. A power distribution board manages the power delivery from a lithium polymer (LiPo) battery to the onboard compute unit and sensors. The onboard compute unit is a Nvidia Jetson Xavier, a system-on-a-chip that contains 8 ARM 64 bit CPU cores and a 512 core GPU. The onboard sensor for localization is a planar LIDAR that operates at 40Hz with a maximum range of 30 meters. The electronic speed controller also provides odometry via the back EMF of the motor.

D Vehicle Software Stack

This section, outlined in Figure 6 gives a detailed overview of the software used onboard the vehicles.

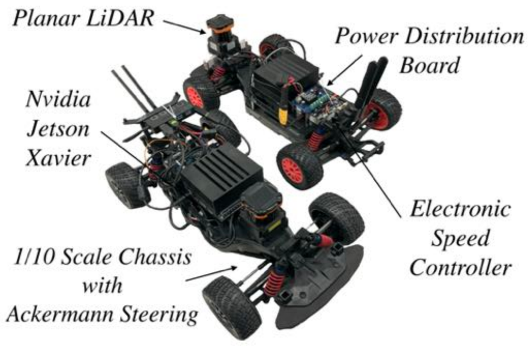

Planar LiDAR
Power Distribution Board
Nvidia Jetson Xavier
Electronic Speed Controller
1/10 Scale Chassis with Ackermann Steering

Figure 5: Components of the 1/10 Scale Vehicle

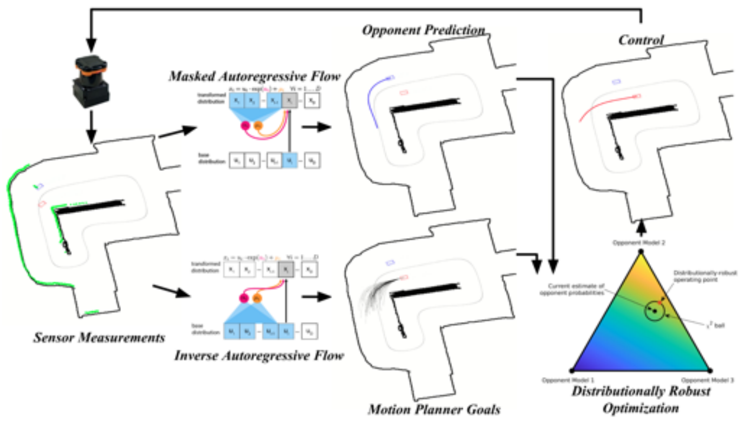

Opponent Prediction
Control
Masked Autoregressive Flow
Sensor Measurements
Inverse Autoregressive Flow
Motion Planner Goals
Distributionally Robust Optimization

Figure 6. FormulaZero on vehicle implementation. Online each agent measures the world using on-board sensors such as a planar LIDAR. Given the sensor measurement the vehicle performs opponent prediction via the use of a masked autoregressive flow and simultaneously selects motion planner goals using an inverse autoregressive flow. Given the set of goals each is evaluated within our DRO framework, the best goal is chosen, and a new control command is applied to the vehicle kicking off the process again.

D.1 Mapping

We create occupancy grid maps of tracks using Google Cartographer [24]. The map's primary use is as an efficient prior for vehicle localization algorithms. In addition, maps serve as a representation of the static portion of the simulation environment describing where the vehicle may drive and differentiating which (if any) portions of the LIDAR scan have line-of-sight to other agents. A feature of our system useful to other researchers is that any environment which can be mapped may be trivially added to the simulator described in Appendix E.

D.2 Localization

Due to the speeds at which the vehicles travel, localization must provide pose estimates at a rate of at least 20 Hz. Thus, to localize the vehicle we use a particle filter [70] that implements a ray-marching scheme on the GPU in order to efficiently simulate sensor observations in parallel. We add a small modification which captures the covariance of the pose estimate. We do not use external localization systems (*e.g.* motion capture cameras) in any experiment.

D.3 Planning

The vehicle software uses a hierarchical planner [20] similar to that of Ferguson et al. [17]. At the top level the planner receives a map and waypoints representing the centerline of the track; the goal is to traverse the track from start to finish. Unlike route planning in road networks, there are no routing decisions to be made. In more complex instances of our proposed environment, this module could be necessary for making strategic decisions such as pit stops. The second key difference is the mid-level planner. Whereas Ferguson et al. [17] uses a deterministic lattice of points, our vehicle draws samples from a neural autoregressive flow. Each sample contains a goal pose and speed profile. Given this specification, the local planner calculates a trajectory parameterized as a cubic spline, evaluates static and dynamic costs of the proposed plan in belief space, and selects the lowest cost option.

D.3.1 Sampling behavior proposals

There are two advantages to using a neural autoregressive flow in our planning framework. First, each agent in the population weights the individual components of its cost function differently; the flow enables the goal generation mechanism to learn a distribution which places more probability mass on the agent's preferences. Second, as planning takes place in the context of the other agent's actions, the ego-agent's beliefs can be updated by inverting the flow and estimating the likelihood of the other agent's actions under a given configuration of the cost function.

The goal-generation process utilizes an inverse autoregressive flow (IAF) [34]. The IAF samples are drawn from a density conditioned on a 101-dimensional observation vector composed of a subsampled LIDAR scan and current speed. Each sample is a 6 dimensional vector: Δt, the perpendicular offset of the goal pose from the track's centerline; Δs, the arc-length along the track's centerline relative to the vehicle's current pose; $\Delta \theta$, the difference between the goal pose's heading angle and the current heading angle; three velocity offsets from the vehicle's current velocity at three equidistant knot points along the trajectory.

The second benefit of using a generative model for sampling behavior proposals is the ability to update an agent's beliefs about the opponent's policy type. As noted in Section 4, masked [51] and inverse autoregressive flows (MAF and IAF respectively) have complementary strengths. While sampling from a MAF is slow, density estimation using this architecture is fast. Thus, we use a MAF network trained to mimic the samples produced by the IAF for this task. The architectures of each network are the same, and we describe this architecture below.

The IAF and MAF networks used in this paper have 5 MADE layers [51] each containing: a masked linear mapping ($\mathbb{R}^6 \to \mathbb{R}^{100}$), RELU layer, masked linear mapping ($\mathbb{R}^{100} \to \mathbb{R}^{100}$), RELU layer, and a final masked linear layer ($\mathbb{R}^{100} \to \mathbb{R}^{12}$). Note that output of a MADE layer includes both the transformed sample and the logarithm of the absolute value of the determinant of the Jacobian of the transformation. For sampling, the latter is discarded, and the transformed sample is passed to the next layer. In addition, the masking pattern is sequential and held constant during both training and inference. This choice was made to aid in debugging of experiments and to simplify communication during distributed training.

Each population member has a dedicated IAF model, which is trained iteratively according to the AADAPT algorithm described in Section 2 using the hyperparameters given in Section 4. We initialize each IAF with a set of weights which approximate an identity transformation for random pairs of samples from a normal distribution and simulated observations. In addition each population member also has a MAF model, which is trained using the same hyperparameters as the IAF but only after AADAPT has finished. The code submitted in the supplementary materials extends an existing library[3] created by other authors; we add support for the IAF architecture as well as generalize the network architecture to 3-dimensional tensors. The latter extension enables sampling from multiple agents' IAF models simultaneously and efficiently.

D.3.2 Model Predictive Control

The goal of the trajectory generator is to compute kinematically and dynamically feasible trajectories that take the vehicle from its current pose to a set of sampled poses from the IAF. The trajectory generator combines approaches from [26, 45, 32, 44]. Each trajectory is represented by a cubic spiral with five parameters $p = [s, a, b, c, d]$ where s is the arc length of the spiral, and (a, b, c, d) encode the curvature at equispaced knot points along the trajectory. Powell's method or gradient descent can be used to find the spline parameters that (locally) minimize the sum of the Euclidean distance between the desired endpoint pose and the forward simulated pose. Offline, a lookup table

[3]`https://github.com/kamenbliznashki/normalizing_flows`

Table 5: The resolution and ranges of the Trajectory Generator Look-up Table

Index	Resolution	Min	Max
Δx	0.1 m	-1.0 m	10.0 m
Δy	0.1 m	-8.0 m	8.0 m
$\Delta \theta$	$\pi/32$ rad	$-\pi/2$ rad	$\pi/2$ rad
κ_0	0.2 rad/m	-1.0 rad/m	1.0 rad/m

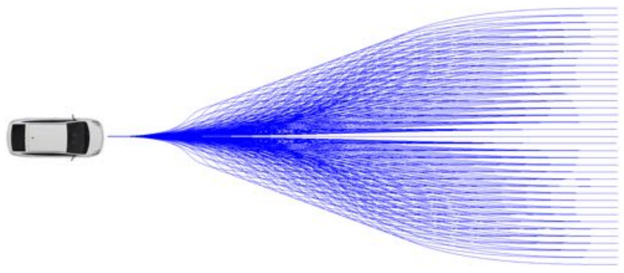


Figure 7: Sample trajectories from the look-up table

of solutions for a dense grid of goal poses is precomputed, enabling fast trajectory generation online. Each trajectory is associated with an index which selects the Δx, Δy, and the $\Delta \theta$ of the goal pose relative to the current pose (where positive x is ahead of the vehicle and postiive y is to the left), and κ_0, the initial curvature of the trajectory. The resolution and the range of the table is listed in Table 5. Figure 7 shows a selection of trajectories. The point on the left of the figure is the starting pose of the vehicle, and the collection of goal poses is shown as the points on the right of the figure.

D.3.3 Trajectory Cost Functions

Each of the generated trajectories is evaluated with the weighted sum of the following cost functions. Note, in order to ensure safety, goals which would result in collision result in infinite cost and are automatically rejected prior to computing the robust cost, which operates only on finite-cost proposals.

1. **Trajectory length:** $c_{al} = s$, where $1/s$ is the arc length of each trajectory. Short and myopic trajectories are penalized.
2. **Maximum absolute curvature:** $c_{mc} = \max_i\{|\kappa_i|\}$, where κ_i are the curvatures at each point on a trajectory. Large curvatures are penalized to preserve smoothness of trajectories.
3. **Mean absolute curvature:** $c_{ac} = \frac{1}{N}\sum_{i=0}^{N}|\kappa_i|$, the notation is the same as c_{mc} and the effect of this feature is similar, but less myopic.
4. **Hysteresis loss:** Measured between the previous chosen trajectory and each of the sampled trajectories, $c_{hys} = ||\theta_{prev}^{[n_1,n_2]} - \theta^{[0,n_2-n_1]}||_2^2$, where θ_{prev} is the array of heading angles of each pose on the previous selected trajectory by the vehicle, θ is the array of heading angles of each pose on the trajectory being evaluated, and the ranges $[n_1, n_2]$ and $[0, n_2 - n_1]$ define contiguous portions of trajectories that are compared. Trajectories dissimilar to the previously selected trajectory are penalized.
5. **Lap progress:** Measured along the track from the start to the end point of each trajectory in the normal and tangential coordinate system, $c_p = \frac{1}{s_{end}-s_{start}}$, where s_{end} is the corresponding position in the tangential coordinate along the track of the end point of a trajectory, and s_{start} is that of the start point of a trajectory. Shorter progress in distance is penalized.
6. **Maximum acceleration:** $c_{ma} = \max_i|\frac{\Delta v_i}{\Delta t_i}|$ where Δv is the array of difference in velocity between adjacent points on a trajectory, and Δt is the array of corresponding time intervals between adjacent points. High maximum acceleration is penalized.
7. **Maximum absolute curvature change:** Measured between adjacent points along each trajectory, $c_{dk} = \max_i|\frac{\Delta \kappa_i}{\Delta t_i}|$. High curvature changes are penalized.

8. **Maximum lateral acceleration:** $c_{la} = \max_i\{|\kappa|_i v_i^2\}$, where κ and v are the arrays of curvature and velocity of all points on a trajectory. High maximum lateral accelerations are penalized.
9. **Minimum speed:** $c_{ms} = \frac{1}{(\min_i\{v_i\})_+}$. Low minimum speeds are penalized.
10. **Minimum range:** $c_{mr} = \min_i\{r_i\}$, where r is the array of range measurements (distance to static obstacles) generated by the simulator. Smaller minimum range is penalized, and trajectories with minimum ranges lower than a threshold are given infinite cost and therefore discarded.
11. **Cumulative inter-vehicle distance short:**
$$c_{dyshort} = \begin{cases} \infty, \text{ if } d(\texttt{ego}_i, \texttt{opp}_i) \leq \texttt{thresh} \\ \sum_{i=0}^{N_{short}} d(\texttt{ego}_i, \texttt{opp}_i), \text{ otherwise} \end{cases}$$
Where the function $d()$ returns the instantaneous minimum distance between the two agents at point i, N_{short} is a point that defines the shorter time horizon for a trajectory of N points. Trajectories with infinite cost on the shorter time horizon are considered infeasible and discarded.
12. **Discounted cumulative inter-vehicle distance long:** $c_{dylong} = \sum_{i=N_{short}}^{N_{long}} 0.9^{i-N_{short}} \frac{1}{d(\texttt{ego}_i, \texttt{opp}_i)}$, where N_{long} is a point that defines the longer time horizon for a trajectory of N points. Note that $N_{short} < N_{long} < N$. Lower minimum distances between agents on the longer time horizon are penalized.
13. **Relative progress:** Measured along the track between the sampled trajectories' endpoints and the opponent's selected trajectory's endpoint, $c_{dp} = (s_{opp\_end} - s_{end})_+$, where s_{opp_end} is the position along the track in tangential coordinates of the endpoint of the opponent's chosen trajectory. Lagging behind the opponent is penalized.

D.3.4 Path tracker

Once a trajectory has been selected it is given to the path-tracking module. The goal of the path tracker is to compute a steering input which drives the vehicle to follow the desired trajectory. Our implementation uses a simple and industry-standard geometrical tracking method called pure pursuit [13, 63]. Due to the decoupling of the trajectory generation and tracking modules it is possible for the tracker to run at a much higher frequency than the trajectory generator; this is essential for good performance.

D.4 Communication and system architecture

The ZeroMQ [25] messaging library is used to create interfaces between the FormulaZero software stack and the underlying ROS nodes that control and actuate the vehicle test bed. Unlike in the simulator, some aspects of the FormulaZero planning function operate non-deterministicaly and asynchronously. In particular we use a sink node to collect observations from ROS topics related to the various sensors on the vehicle in order to approximate the step-function present in the Gym API. When a planning cycle is complete, the trajectory is published back to ROS and tracked asynchronously using pure-pursuit as new pose estimates become available. Because perception is not the primary focus of this project we simplify the problem of detecting and tracking the other vehicle. In particular, each vehicle estimates its current pose in the map obtained by its onboard particle filter, and this information is communicated to the other vehicle via ZeroMQ over a local wireless network. Since tracking and detection has been well studied in robotics, solutions which rely less on communication could be explored by other future work which builds upon this paper.

E Simulation Stack

The simulation stack includes a lightweight 2D physics engine with a dynamical vehicle model. Then on top of the physics engine, a multi-agent simulator with an OpenAI Gym [11] API is used to perform rollouts of the experiments.

E.1 Vehicle Dynamics

The single-track model in Althoff et al. [1] is chosen because it considers tire slip influences on the slip angle, which enables accurate simulation at physical limits of the vehicle test bed. It is also easily enables changes to the driving surface friction coefficient in simulation which allows the simulator to model a variety of road surfaces.

E.2 System Identification

Parameter identification was performed to derive the following vehicle parameters: mass, center of mass, moment of inertia, surface friction coefficient, tire cornering stiffness, and maximum acceleration/deceleration rates following the methods described in O'Kelly et al. [49].

E.3 Distributed Architecture

Due to the nature of the AADAPT algorithm, the rollouts in a single vertical step do not need to be in sequence. The ZeroMQ messaging library is used to create a MapReduce [14] pattern between the task distributor, result collector, and the workers. Each worker receives the description of the configuration to be simualted, *e.g.* (x, θ). Then the workers asynchronously perform simulations and send results to the collector.

E.4 Addressing the simulation/reality gap

As noted in Section 4 there are several differences between the observations in simulated rollouts and reality. First, pose estimation errors are not present in the simulator. A simple fix would be to add Gaussian white noise to the pose observations returned by the simulator. We avoided this and other domain randomization techniques in order to preserve the determinism of the simulator, but we will investigate its effect in further experiments. Second, the LIDAR simulation does not account for material properties of the environment. In particular, surfaces such as glass do not produce returns, causing subsets of the LIDAR beams to be dropped. We hypothesize that simple data augmentation schemes which select a random set of indices to drop from simulated LIDAR observations would improve the robustness to such artifacts when the system is deployed on the real car; we are currently investigating this hypothesis.

F Experiments

F.1 Instantaneous time-to-collision (iTTC)

Let $T_i(t)$ be the instantaneous time-to-collision between the ego vehicle and the i-th environment vehicle at time step t. The value $T_i(t)$ can be defined in multiple ways (see *e.g.* Sontges et al. [64]). Norden et al. [48] define it as the amount of time that would elapse before the two vehicles' bounding boxes intersect assuming that they travel at constant fixed velocities from the snapshot at time t. Time-to-collision captures directly whether or not the ego-vehicle was involved in a crash. If it is positive no crash occurred, and if it is 0 or negative there was a collision.

F.2 Out-of-distribution agent strategies

In the following sections, we describe the human-created algorithms used in our out-of-distribution analysis.

F.2.1 OOD1: RRT* with MPC-based Opponent Prediction

This approach exploits the fact that the two-car racing scenario is similar to driving alone on the track with the only exception being during overtaking the opponent. This approach uses a costmap-based RRT* [31] planning algorithm. The agent first uses the opponent's current pose and velocity in the world, and uses Model-Predictive Control to calculate an open loop trajectory of N optimal inputs resulting in N+1 states based on a given cost function and constraints. Specifically, the optimization problem is constrained by a linearized version of the single track model described in Althoff et al. [1], and by the boundary values of the inputs and states of the vehicle. The cost function that the optimization tries to minimize consists of the trajectory length and input power requirement. The costmap used by RRT* also incorporates this predicted trajectory of the opponent vehicle by inflating the two-dimensional spline representing the prediction, and weighting the portion of the spline closer to the ego vehicle higher. RRT* samples the two dimensional space that the vehicle lies in. The path generated by RRT* is then tracked with the Pure Pursuit controller [13].

F.2.2 OOD2: RL-based Lane Switching

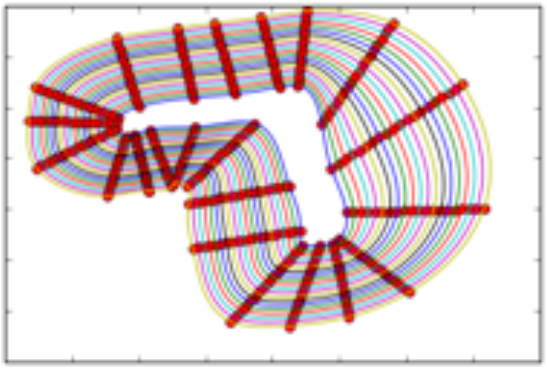


Figure 8: Lanes that cover the track

The second algorithm is based on a lane-switching planning strategy that uses an RL algorithm to make lane switching decisions, and filters out unsafe decisions using a collision indicator. First, as shown in 8, different lanes going through numerous checkpoints on the track are created to cover the entirety of the race track. Then a network is trained to make lane switching decisions. The state of the RL problem consists of the sub-sampled LIDAR scans of the ego vehicle; the pose (x, y, θ) of the opponent car with respect to the ego vehicle; velocity (v_x, v_y) of the opponent vehicle with respect of the ego vehicle; projected distance from the ego vehicle's current position to all pre-defined paths. The reward of a rollout is zero in the beginning. At each timestep, the timestep itself is subtracted from the total reward. A rollout receives -100 as the reward when the ego agent collide with the environment or the other agent. And finally, if both agents finish 2 laps, the difference between lap times (positive if the ego agent wins) of the two agents are added to the reward. Clipped Double Q-Learning [18] is used to estimate the Q function and make the lane switching decisions. iTTC defined in Appendix F.1 is used as an indicator for future collisions. If any decisions made by the RL network would result in a collision indicated by the iTTC value, the safety function kicks in and makes the lane switching decision based on the collision indicator. Finally, ego vehicle actuation is provided by the same Pure Pursuit controller [13] tracking the selected lane. We used an existing implementation[4] of this algorithm.

[4]`https://github.com/pnorouzi/rl-path-racing`