University of Pennsylvania
ScholarlyCommons

Publicly Accessible Penn Dissertations

2021

Accelerated Risk Assessment And Domain Adaptation For Autonomous Vehicles

Matthew O'kelly
University of Pennsylvania

Follow this and additional works at: https://repository.upenn.edu/edissertations

Part of the Computer Sciences Commons, Electrical and Electronics Commons, and the Robotics Commons

Recommended Citation
O'kelly, Matthew, "Accelerated Risk Assessment And Domain Adaptation For Autonomous Vehicles" (2021). *Publicly Accessible Penn Dissertations*. 3855.
https://repository.upenn.edu/edissertations/3855

This paper is posted at ScholarlyCommons. https://repository.upenn.edu/edissertations/3855
For more information, please contact repository@pobox.upenn.edu.

Accelerated Risk Assessment And Domain Adaptation For Autonomous Vehicles

Abstract

Autonomous vehicles (AVs) are already driving on public roads around the US; however, their rate of deployment far outpaces quality assurance and regulatory efforts. Consequently, even the most elementary tasks, such as automated lane keeping, have not been certified for safety, and operations are constrained to narrow domains. First, due to the limitations of worst-case analysis techniques, we hypothesize that new methods must be developed to quantify and bound the risk of AVs. Counterintuitively, the better the performance of the AV under consideration, the harder it is to accurately estimate its risk as failures become rare and difficult to sample. This thesis presents a new estimation procedure and framework that can efficiently evaluate and AV's risk even in the rare event regime. We demonstrate the approach's performance on a variety of AV software stacks. Second, given a framework for AV evaluation, we turn to a related question: how can AV software be efficiently adapted for new or expanded operating conditions? We hypothesize that stochastic search techniques can improve the naive trial-and-error approach commonly used today. One of the most challenging aspects of this task is that proficient driving requires making tradeoffs between performance and safety. Moreover, for novel scenarios or operational domains there may be little data that can be used to understand the behavior of other drivers. To study these challenges we create a low-cost scale platform, simulator, benchmarks, and baseline solutions. Using this testbed, we develop a new population-based self-play method for creating dynamic actors and detail both offline and online procedures for adapting AV components to these conditions. Taken as a whole, this work represents a rigorous approach to the evaluation and improvement of AV software.

Degree Type
Dissertation

Degree Name
Doctor of Philosophy (PhD)

Graduate Group
Electrical & Systems Engineering

First Advisor
Rahul Mangharam

Subject Categories
Computer Sciences | Electrical and Electronics | Robotics

This dissertation is available at ScholarlyCommons: https://repository.upenn.edu/edissertations/3855

ACCELERATED RISK ASSESSMENT AND DOMAIN ADAPTATION FOR AUTONOMOUS VEHICLES

Matthew O'Kelly

A DISSERTATION

in

Electrical and Systems Engineering

Presented to the Faculties of the University of Pennsylvania

in

Partial Fulfillment of the Requirements for the

Degree of Doctor of Philosophy

2021

Supervisor of Dissertation

Rahul Mangharam, Associate Professor of Electrical and Systems Engineering,
University of Pennsylvania

Graduate Group Chairperson

Victor Preciado, Associate Professor of Electrical and Systems Engineering,
University of Pennsylvania

Dissertation Committee:

Rajeev Alur, Zisman Family Professor of Computer and Information Science,
University of Pennsylvania

Pratik Chaudhari, Assistant Professor of Electrical and Systems Engineering,
University of Pennsylvania

Rahul Mangharam, Associate Professor of Electrical and Systems Engineering,
University of Pennsylvania

Russ Tedrake, Toyota Professor of Electrical Engineering and Computer Science,
Massachusetts Institute of Technology

ACCELERATED RISK ASSESSMENT AND

DOMAIN ADAPTATION FOR AUTONOMOUS VEHICLES

© COPYRIGHT

2021

Matthew Edward O'Kelly

for Nana Liu, who helped me finish, and Morton and Susan O'Kelly, who helped me start

ABSTRACT

ACCELERATED RISK ASSESSMENT AND
DOMAIN ADAPTATION FOR AUTONOMOUS VEHICLES

Matthew O'Kelly

Rahul Mangharam

Autonomous vehicles (AVs) are already driving on public roads around the US; however, their rate of deployment far outpaces quality assurance and regulatory efforts. Consequently, even the most elementary tasks, such as automated lane keeping, have not been certified for safety, and operations are constrained to narrow domains. First, due to the limitations of worst-case analysis techniques, we hypothesize that new methods must be developed to quantify and bound the risk of AVs. Counterintuitively, the better the performance of the AV under consideration, the harder it is to accurately estimate its risk as failures become rare and difficult to sample. This thesis presents a new estimation procedure and framework that can efficiently evaluate and AV's risk even in the rare event regime. We demonstrate the approach's performance on a variety of AV software stacks. Second, given a framework for AV evaluation, we turn to a related question: how can AV software be efficiently adapted for new or expanded operating conditions? We hypothesize that stochastic search techniques can improve the naive trial-and-error approach commonly used today. One of the most challenging aspects of this task is that proficient driving requires making tradeoffs between performance and safety. Moreover, for novel scenarios or operational domains there may be little data that can be used to understand the behavior of other drivers. To study these challenges we create a low-cost scale platform, simulator, benchmarks, and baseline solutions. Using this testbed, we develop a new population-based self-play method for creating dynamic actors and detail both offline and online procedures for adapting AV components to these conditions. Taken as a whole, this work represents a rigorous approach to the evaluation and improvement of AV software.

TABLE OF CONTENTS

CHAPTER 1 : Introduction

1.1. Motivation

Many of the devices integral to our daily lives are imbued with the capacity to observe and engage the physical world. The design and analysis of such systems is characterized by the challenges arising from the interplay of digital computation, networked communication, and continuous evolution of the combined physical state of both the device and environment. Today, the degree of autonomy afforded to such *cyber-physical systems* varies based both on the availability of performance guarantees and the consequences of failures; thus, the use of automation in safety critical settings remains limited. In fact, in many applications the primary barrier to the deployment of highly automated functions is the inability to demonstrate that the software is trustworthy (Wing, 2020).

The development of automated verification (Baier and Katoen, 2008) and synthesis (Gulwani et al., 2017) methods has long been proposed as a means to temper the risks of safety-critical software by mechanizing either its evaluation or production. Verification is the process of establishing that a particular software artifact satisfies its specification. By this definition, a verification method generally certifies whether a model of the program and environment is correct or not. In deterministic methods such as model checking (Clarke, 2008), any specification violation is sufficient to falsify the program's correctness. In contrast to verification, synthesis methods begin with a specification capturing the programmer's intent encoded as a logical formula, a set of input-output examples, or a test environment. The goal is to automatically generate software which satisfies the specification.

1.1.1. Verification

Most verification methods developed for the analysis of cyber-physical systems are concerned with the evaluation of specifications which map the signals produced by the system to a Boolean value which captures whether a particular performance criteria is met. For example, consider the trajectory of poses produced by a mobile ground robot. For each state in the trajectory, the bounding box of the robot can be checked against a map of the environment and the poses of other agents for collision; thus, reducing the trajectory of poses to a binary measure of whether the robot crashed. A deterministic verification method exhaustively searches the space of trajectories to certify that no execution of the system exists in which the robot crashes. If a trajectory is found in which the robot crashes the method terminates and returns the counterexample.

However, the concept of binary correctness can be problematic when applied to cyber-physical systems. In the verification setting it may be impossible to provide a complete specification of the system that separates failures from adversarial inputs of the environment. Currently accepted applications of automation like a caged factory robot or a 747 autopilot rarely encounter this challenge, as the operational domain is highly predictable, bounded, and easy to observe. In this thesis we instead expect cyber-physical systems to interact with complex open worlds inhabited by other uncontrollable decision makers whose states are only partially observable. Given this assumption, we must consider what should be done about adversarial and ambiguous situations such as: a diabetes patient who doesn't accurately track their carbohydrate intake and is dosed with too much insulin by an artificial pancreas, an autonomous car which collides with a wrong-way driver, or a security robot which has been pushed into a fountain by disgruntled patrons. The long tail of scenarios like these all but guarantee that the systems can't be perfectly safe and plague attempts to evaluate autonomy via worst-case analysis.

If the wrong-way driver in the example above is included in the model of the environment against which an autonomous car will be evaluated, then even before the verification process begins we know that the end result will be a counterexample rather than a proof of safety. In this sense, the inability to distinguish between failures and adversarial cases short-circuits the verification process without revealing critical information about the quality of the system. One way to distinguish between executions of the system-under-test that have the same

Boolean value is to evaluate how *robustly* a system execution satisfies a given specification by measuring how close the trajectory gets to a violation. Other mechanisms, which assign blame in the event of a failure, are also useful in differentiating whether the system-under-test upheld social norms or legal responsibilities. While specification robustness (Fainekos and Pappas, 2006) and blame (Shalev-Shwartz et al., 2017) can guide search mechanisms towards critical regions and encode the severity of system failures, neither inherently captures an important facet of the systems performance: the frequency of undesirable or unsafe events.

Why should we be concerned with the frequency of unsafe events? Consider the case of a robo-taxi that will drive you to work each morning. Suppose that it is guaranteed to never have an accident for which it is legally liable; however, everyday it is rear-ended (but not at fault) at a four-way stop on the route. A measurement of the systems safety and quality should consider the frequency of such occurrences. No skilled human driver would have such frequent collisions, and perhaps more importantly no riders would choose to use such an inconvenient and dangerous mode of transportation. In this thesis, we view the problem of system evaluation through the lens of statistical model checking methods, which, unlike deterministic model checking, seek to estimate the probability of failure rather than certify its (non-)existence. The end result is that catastrophic failures (regardless of fault) which are predicted to occur frequently must be addressed because they imply that the system exceeds an acceptable risk threshold.

1.1.2. Synthesis

Thus far, we have assumed that the system-under-test has been simply given to the verifier to evaluate. In reality the development of autonomous systems which can be meaningfully considered for deployment is a time-consuming and expensive task. Suppose an instance of the system has been through the verification and validation process. The second part of this thesis is concerned with how the components of the existing system can be leveraged to solve similar problems in variations of the original operational domain. For example, we would like to update a pacemaker for use in a new country, modify an orbital rocket recovery system to land on a moving barge instead of a launchpad, or adapt a highway autopilot system designed for a sedan for use on a semi-truck.

Fortunately, many cyber-physical systems contain a common, modular core of perception, planning, and control components that can be adapted or reused with relatively small parametric rather than structural modifications. The challenge is that the design and tuning of the system level code is ad-hoc and time-consuming. Moreover, small isolated changes can have far-reaching consequences across modules and manifest as unsafe or undesirable behavior. As a result it is beneficial to evaluate and modify the modules as a system rather than independently. We hypothesize that extensions of inductive synthesis and reinforcement learning techniques have the potential to address the challenges of domain adaptation by jointly-optimizing the performance of existing software components.

Approaches to synthesis can be broadly categorized in terms of the specification type, search space description, and search method (Gulwani et al., 2017). Specifications may be logical formulas, examples of correct input-output behavior, or a reward function. Today, the line between synthesis and reinforcement learning is becoming increasingly blurry. Both reinforcement learning and inductive synthesis techniques leverage the fixed structure of existing software modules (*e.g.* a particular neural network architecture or a program sketch) to constrain the search space. From a reinforcement learning perspective, this approach introduces significant inductive bias into the learning process. Because we suppose that the modules chosen are already proficient at solving related tasks, this bias is well-founded. However, one of the downsides of using arbitrary programs to constrain the search space is that they do not necessarily enjoy useful properties like differentiability, which enable end-to-end training of specialized models like deep neural networks commonly used in supervised learning. Thus, in this work we aim to develop highly parallel stochastic search techniques capable of adapting autonomy components to improve system level performance while maintaining safety.

1.2. Autonomous Vehicles

This work is focused on how concepts from verification and synthesis methods may be extended to assess and mitigate the risk of autonomous vehicles (AVs). AVs are already driving on public roads around the US; however, their rate of deployment far outpaces quality assurance and regulatory efforts. Consequently, even the most elementary tasks, such as automated lanekeeping, have not been certified for safety, and operations are constrained to narrow domains. While the potential benefits of AVs range from lower accident rates to faster commutes and reduced pollution, well-publicized crashes, including at least half-a-dozen fatalities, have already happened. Today, there is no accepted method to quantify and bound the risk that an AV poses to its passengers and to other traffic participants. If unchecked, AV manufacturers could face significant liability precluding the widespread introduction of the technology. Furthermore, if too many accidents occur without clear evidence that the technology is safe, the public excitement regarding AVs may soon turn to distrust. In short, for AV technology to reach the market, we will need both evidence that the probability of AV failure is vanishingly small, and a means to correct or refine the AV system if and when failures are discovered or the operational domain is expanded.

In Section 1.1.1 we hypothesized that measuring the risk of cyber-physical systems like AVs is necessary when absolute safety cannot be guaranteed. Informally, risk measures the exposure to an event which has an associated loss where there exists both some source of randomness and a decision-making capability. The challenge is that evaluating the risk of an AV in the real world is expensive, slow, and dangerous. In response, this thesis develops methods for rare-event simulation in order to estimate, efficiently, the probability of system failure. Utilizing these methods at scale requires additional components. In response we also create a framework for: (1) learning complex generative models which capture the possible behaviors of the other drivers and (2) running many simulations of the AV system in parallel on the cloud.

In Section 1.1.2 we hypothesized that synthesis-based techniques can enable the adaptation of existing autonomy software modules to new operational domains. In the context of AV development, one of the most challenging aspects of this task is that proficient driving requires a balance of performance and safety. In particular, current 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. Just as in the setting of risk-evaluation, it is preferable to leverage simulation for cost, efficiency, and safety reasons. However, for novel scenarios or operational domains there may be little data which can be used to learn a generative model of the other drivers. In this thesis we propose a self-play based methods to learn a diverse set of agent behaviors which can be used both in an offline learning or synthesis setting and for online planning. Through these techniques we can automatically tune AV systems to achieve better performance without sacrificing safety.

1.3. Overview and Contributions

The contributions of this thesis are divided into two parts. In Part I we focus on the task of evaluation, that is *assessing* the risk that an AV poses, both to its own occupants and other drivers, by measuring the frequency of dangerous events. In Part II we consider adapting existing AV planning and control methods to improve their performance (and thus reduce their risk) for the operational environment which they will be deployed. In what follows we give an overview of each chapter and its contributions.

1.3.1. Part I: Accelerated Risk Assessment

Data-driven and learning-based approaches are widely utilized to enable robots and autonomous systems that intelligently interact with unstructured environments. Unfortunately, evaluating the performance of the closed-loop system is challenging, limiting the success of such methods in safety-critical settings. Today, even if we produce autonomous systems better than a human at driving, flying a plane, or performing surgery, we have no tractable way to certify the system's quality. Counterintuitively, the better the performance of the agent under consideration, the harder it is to rigorously compute statistics about it because failures are rare and difficult to sample.

Chapter 2 explores the applicability of a subset of formal methods, namely, verification and falsification, to the problem of evaluating autonomous vehicle safety. We develop a formal approach for defining an AV's operational domain and conduct a series of experiments in order to understand the relative strengths and weaknesses of verification and falsification tools. We find that falsification methods can quickly identify examples of failures of a minimally abstracted AV system model, but they do not guarantee coverage of all failure modes. In contrast, verification methods do provide coverage but require careful consideration of the environment model to avoid trivial failures which are not caused by the AV's design. Moreover, verification methods require significant abstraction of the system-under-test and scale poorly with operational domain complexity (*e.g.* the number of agents). Chapter 2 concludes by motivating the risk-based approach pursued in the remainder of Part I.

In Chapter 3 we formalize the problem of risk assessment for autonomous vehicles and develop a method for creating generative models of the vehicle's operational domain. Motivated by the challenges underlying real-world testing and formal verification, we consider a probabilistic paradigm—which we call a *risk-based framework*—where our goal is to evaluate the *probability of an accident* under a base distribution representing standard traffic behavior. Formally, we let P_0 denote the base distribution that models standard traffic behavior and $X \sim P_0$ be a realization of the simulation (*e.g.* weather conditions and driving policies of other agents). For an objective function $f : \mathcal{X} \to \mathbb{R}$ that measures "safety"—so that low values of $f(x)$ correspond to dangerous scenarios—our goal is to evaluate the probability of a dangerous event

$$p_\gamma := \mathbb{P}_0(f(X) \leq \gamma) \tag{1.1}$$

for some threshold γ. Our risk-based framework is agnostic to the complexity of the ego-policy, and views it as a black-box module. Such abstraction allows, in particular, deep-learning based perception systems that make formal verification methods intractable. As serious accidents are rare (p_γ is small), we view this as a *rare-event simulation* problem; naive Monte Carlo sampling methods require prohibitively many simulation rollouts to generate dangerous scenarios and estimate p_γ. To accelerate safety evaluation, we use adaptive importance-sampling methods to learn alternative distributions P_θ that generate accidents more frequently. The primary contribution of Chapter 3 is a demonstration of a framework for large-scale risk estimation which uses data-driven generative models.

In Chapter 4 we propose a novel method for rare-event simulation. Problem (1.1) is often solved in practice by naive Monte Carlo estimation methods, the simplest of which *explore* the search space via random samples from P_0. These methods are unbiased and easy to parallelize, but they exhibit poor sample complexity. Naive Monte Carlo can be improved by adding an adaptive component *exploiting* the most informative portions of random samples drawn from a sequence of approximating distributions $P_0, P_1, \ldots, P_K$. However, standard adaptive Monte Carlo methods (*e.g.* Cérou and Guyader, 2007), though they may use first-order information on the distributions P_k themselves, fail to use first-order information about f to improve sampling; we explicitly leverage this to accelerate convergence of the estimate through *optimization*. The primary contribution of Chapter 4 is a provably efficient non-parametric method for risk estimation which can leverage first-order information. A major focus of this work is empirical, and accordingly, we demonstrate the superiority of our method over competing techniques in a variety of applications: (i) we perform model comparisons for two learning-based approaches to autonomous navigation, (ii) we evaluate a state-of-the-art autonomous driving stack which has driven over 20 million miles on public roads.

1.3.2. Part II: Domain Adaptation

Given a framework for AV evaluation, we turn to a related question: how can AV software be efficiently adapted for new or expanded operating regimes? We hypothesize that stochastic search techniques can improve the naive trial-and-error approach commonly used today. To study these challenges we first create a low-cost scale autonomous racing platform, simulator, benchmarks, and baseline solutions. While the platform has all the features of a general purpose AV, we use the challenge of autonomous racing to sharpen the tradeoff between performance and safety. Scenarios faced by racing drivers (and autonomous racers) force the development of technology which must operate safely in both nominal conditions and, more importantly, at the limits of vehicle performance.

Chapter 5 details the F1TENTH autonomous racing platform, an open-source evaluation framework for training, testing, and evaluating autonomous systems. With 1/10th-scale low-cost hardware and multiple virtual environments, F1TENTH enables safe and rapid experimentation of AV algorithms even in laboratory research settings. We present three benchmark tasks and baselines in the setting of autonomous racing, demonstrating the flexibility and features of our evaluation environment. The primary contributions of Chapter 5 are simulators that serve as a virtual evaluation environment, enabling rapid implementation and dissemination of algorithms. In addition we develop the low-cost, 1/10th scale vehicle and standardized software infrastructure that supplements our virtual benchmarking tools, enabling the use of a wide range of reinforcement learning algorithms in reality. Finally, we describe three benchmarks in our environment, spanning continuous control and reinforcement learning methods.

Chapter 6 explores the problem adapting an autonomous racecar to a new track by jointly optimizing the racing strategy, planning components, control algorithms, and vehicle parameters. The primary contribution is a toolchain called TunerCar. This toolchain includes target hardware, modular software, a calibrated simulator, and an algorithm which searches for high performance vehicle configurations. We validate our solution with both real and simulated experiments on the AV software stack and 1/10th-scale open-source vehicle described in the previous chapter. We compare the performance of our approach against the existing F1TENTH baselines and to hand-tuned solutions submitted by over 30 international teams, comprised of graduate students working in the field of autonomous vehicles. For all tested tracks, our method provides the lowest lap time.

Chapter 7 investigates the challenge of balancing performance and safety in order to deploy autonomous vehicles in multi-agent environments. Again, we study autonomous racing, 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.

Part I

Accelerated Risk Assessment

CHAPTER 2 : Formal Methods for Autonomous Vehicle Validation and Verification

2.1. Chapter Overview

Elements of this chapter have been adapted from “APEX: Autonomous Vehicle Plan Execution and Verification” in the proceedings of the SAE 2016 and “Computer-aided Design for Safe Autonomous Vehicles” in the proceedings of Resilience Week 2017. These papers were joint work with Houssam Abbas, Shinpei Kato, Shin’ichi Shirashi, Sicun Gao, and Rahul Mangharam.

2.2. Introduction

The electronic design automation industry has a long history of successfully managing the complexity of semiconductor development by providing tools that enable easy design entry, modular design, abstraction, formal equivalency checking between abstractions, automated test generation, formal verification, and the re-use of tests and other artifacts across abstractions. In this chapter we explore the applicability of a subset of these formal techniques, namely, verification and falsification, to the problem of evaluating autonomous vehicle safety.

The novelty of the AV domain is that AVs need to operate in a variety of scenarios (*e.g.* parking and merging). Because the AV will execute different controllers depending on the scenario, it must be verified across a broad cross-section of representative situations. Each scenario may have an infinite variety of instantiations (highways with different curvatures, intersections with different traffic signage); therefore, it is also important to obtain good coverage of a given scenario. Safety-criticality implies that coverage must be rigorously measured or bounded, rather than set by an arbitrary timeout on the duration of verification. In order to concertize the concepts of representative scenarios, coverage, and the necessity of a formal guarantee we begin with a motivating example.

2.2.1. Motivating Example

What type of evidence should we require before deploying an autonomous vehicle? To answer this question, consider the situations presented in Figure 1, all of which are variations on a lane change scenario. Every car in the scenario is characterized by its state, x. In this example, we use a 7-dimensional state. To simulate the scenario, we select initial values for these variables, i.e., an initial state $x(0)$ (which we call a configuration). The initial state can have any value in a bounded set: *e.g.* , in Figure 1, the initial position (s_x, s_y) of the ego vehicle is in $[0, 1] \times [0, 1]$, and the velocity v is in $[24, 33] m/s$. There are two distinct sources of performance variation for an AV within a given scenario: first, the ego vehicle will have to perform a lane change under a variety of initial states. Simulating it under only one initial state is clearly insufficient, because we expect the outcome of the scenario when the two cars start $0.5m$ apart to differ from its outcome when the two cars start $5m$ apart. The second source of variation comes from errors in perception such as localization and velocity estimate. Even if we wish to start the simulation in a particular state, inaccuracies in measurements mean that the car's state cannot be exactly known. So while the algorithms which control the AV assume a given starting state, in reality the car may actually begin from anywhere in a set containing the state estimate. Thus, it is also important to verify that these measurement errors do not cause unsafe situations.

The question then becomes: how many simulations should we perform, and *which* simulations should we perform? Ideally, we would simulate all configurations that produce an unsafe outcome; we will return to this idea in Chapters 3 and 4. For now, we will assume that attaining coverage of failure modes either algorithmically or via domain knowledge is a challenging problem. Even experienced engineers might not think of corner cases, especially given the size of the vehicles configuration space. Moreover, if we naively test the system, as in Figure 1 where we show a lane change scenario which has been simulated a 1000 times, we have no way to know whether the system is safe or if we have just been unlucky and sampled only easy tests. In our example it is the last, non-simulated, situation that reveals the

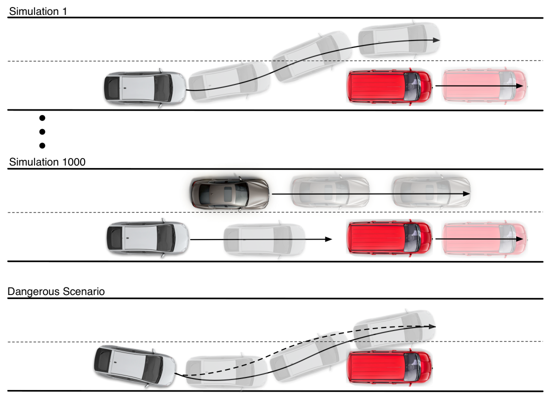


Figure 1: Simulation is not sufficient to fully verify a lane change. After a large number of simulations, the unsafe scenario at the bottom may still not be detected as simulation-based testing is not exhaustive and leaves a verification gap.

collision: the ego vehicle must start with a positive orientation and yaw rate, and attempt to change lanes while the other vehicle is slowing down. It is only through the combination of small errors and the presence of particular environment behavior that the ego vehicle can deviate from the planned reference trajectory in a dangerous manner.

What is clear is that attempts to attain coverage via methods like grid search, where the configurations are sampled from hypercubes of parameters, are not scalable solutions. Suppose we decide to sample only 10 points in the range of every state variable, and each simulation represents 10 seconds of driving. For our 7D model, and with 2 cars, this yields a total of 10^{14} simulations. If each simulation runs in real-time, we require $10 * 10^{14}$ seconds $= 30$ million years of computation with no guarantee that a bug has not been missed. Thus, in both this chapter and the rest of the thesis, we develop and evaluate methods that enable more efficient exploration of search space, leverage massive parallel computation, and seek algorithmic guarantees of failure mode coverage.

2.2.2. Outline

In Section 2.3 we formalize the definitions of safety and robustness that we will use throughout this chapter. Given this definition of safety, Section 2.4 describes a formal model of scenarios from which we can sample closed loop executions of an AV's interaction with its environment or use to perform verification. In Section 2.5 we briefly describe tools for the two methods of safety analysis: verification and falsification. For each tool chain we provide experimental results in Section 2.6 and discuss the relative strengths and weaknesses of each approach. In particular, while falsification methods can quickly find examples of failures of the AV system they do not guarantee coverage of all failure modes. In contrast, verification methods do provide coverage but require careful consideration of the environment model to avoid trivial failures that are not caused by the AV's design. Moreover, verification methods scale poorly as more agents are added and the time horizon of the scenario is extended. Section 2.7 concludes the chapter by proposing a simple modification to our definition of a scenario that enables a re-framing of the safety assessment problem.

2.3. Measuring Safety

The guidance issued by the National Highway Traffic Safety Administration (NHTSA) on the elements of a safety assurance case for AVs (NHTSA, 2016) is a starting point for standardizing the *type* of safety and correctness evidence needed for deployment of AVs. However, it does not prescribe *how* such evidence should be obtained, nor at which point of the design cycle analysis should be performed. Our starting point for the development of evaluation tools is that AV correctness (including safety) is a *spectrum*: *i.e.*, we should be able to order vehicles according to how safe they are relative to each other, and compare one AV's performance across different scenarios. This is routinely done for human-driven cars, which receive safety ratings based on their crash performance. For AVs, such a continuous measure of correctness can and should be obtained *at design time, and measured throughout the design cycle* as the vehicle model is refined and as more components are implemented. Moreover, it is not just a measure of safety, but more generally a measure of how *robustly* the AV satisfies potentially complicated requirements.

To understand the approach, consider the T-Junction scenario in Figure 2. The ego-vehicle, which is the AV under test, must make a right turn while satisfying the following requirements: 1. At all times, stay at least 2 m from any obstacle. 2. If the ego-vehicle is already in the intersection and the approaching vehicle from the left is closer than 3 m, reach a speed of 25 mph within 6 seconds. 3. Either reach the green rectangle within 1 min or stay at the starting position until the road is clear.

These requirements increase in complexity: the first is a static no-collision requirement, the second adds a reactive time-constrained element, and the third adds a purely temporal element. What are meaningful continuous measures of correctness for these three requirements? For requirement 1, a meaningful measure ρ would be the minimum distance between the vehicle and any fixed obstacle over the course of the simulation. For requirement 2, things are more complicated because of the two possibilities. Nonetheless, it is reasonable to say that the correctness measure ρ in this case equals either the minimum distance between the two cars if it is above 3 m (so the minimum speed requirement is irrelevant), otherwise it equals the difference between the maximum car speed and 25 mph over the 6 second window. Intuitively, in the first case the correctness measure indicates how close the other car actually got to colliding, while in the second case what is important is whether the ego-vehicle reacted correctly, and this is measured by how fast it actually got above the minimum threshold of 25mph.

What about the third requirement? Things are even more complicated because of the temporal 'until' component: should the correctness measure reward entering the intersection earlier? Should it differentiate between two different behaviors after the road clears? And what if *all* three requirements are part of the vehicle specification? How do we balance between all of them? It becomes clear that we need a *systematic* way of calculating this correctness measure for *arbitrary* specifications involving reactive, spatio-temporal requirements. Such a systematic measure of correctness is provided by the *robustness function* of Metric Temporal Logic (MTL) specifications (Koymans, 1990).

The tools proposed in this thesis are compatible with requirements as a formula φ in MTL, which is a formal mathematical language for writing temporal specifications. The *robustness* $\rho_\varphi(\mathbf{x})$ of the MTL specification φ is computed relative to a given system execution $\mathbf{x}$. The robustness $\rho_\varphi(\mathbf{x})$ is a real number that measures two things (Fainekos and Pappas, 2009): its sign tells whether $\mathbf{x}$ satisfies the specification ($\rho_\varphi(\mathbf{x}) > 0$) or violates it ($\rho_\varphi(\mathbf{x}) < 0$). Moreover, the trajectory $\mathbf{x}$ can be disturbed by an amount $|\rho_\varphi(\mathbf{x})|$ without changing its truth value (e.g., if it is correct, the disturbed trajectory is also correct). Thus, robustness is a continuous measure of correctness of the AV with the desired properties.

2.4. Scenario Description Language

The *Scenario Description Language* (SDL) allows the user to quickly specify a driving scenario with the following components. A scenario $\mathbf{S}$ consists of a set of agents, A, that includes the ego-vehicle and other vehicles and the road, a set of traffic laws $\mathcal{L}$, a goal Φ to be achieved by the ego-vehicle in this scenario, exit conditions $\mathcal{E}$ that define when a scenario is over (otherwise the verification tools might not know when to terminate), and a set of states $Init$ that captures the initial states of all vehicle agents.

$$\mathbf{S} = (A, \mathcal{L}, \Phi, Init, \mathcal{E})$$

Rather than define these formally, it is best to illustrate them using the T-Junction scenario depicted in Figure 2.

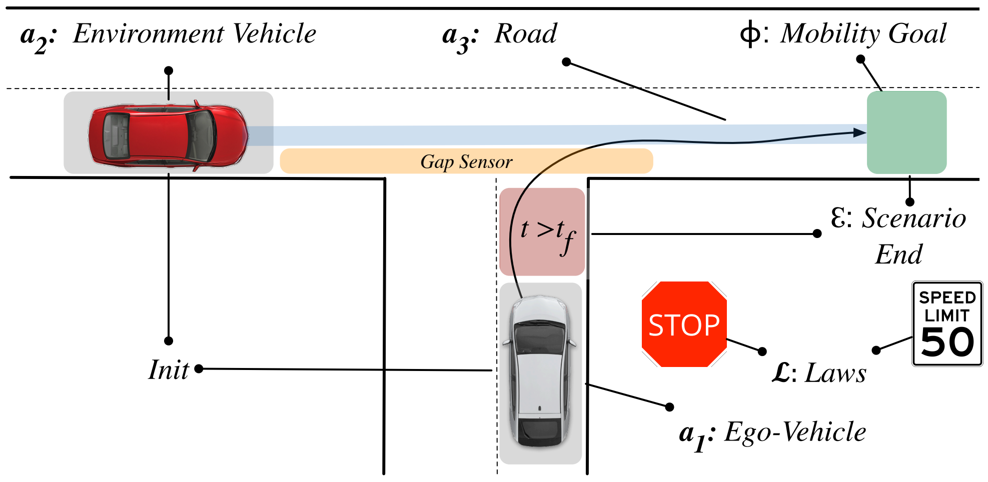


Figure 2: The T-Junction Scenario.

Agents (A): There are 3 agents: the `ego-vehicle` (a_1), `environment-vehicle` (a_2), and the `road` (a_3). It is unrealistic to assume perfect knowledge of the environment vehicle's intentions and dynamics. Thus, the latter must be modeled non-deterministically, e.g. $\dot{x} = v_x, \dot{v}_x \in [0, 1]\frac{m}{s^2}$. Roads are described as finitely-parameterized curves: e.g., straight lines (with parameter: *length*) and cubic splines (parameters: $length, curvatures$). Thus when the scenario is verified, the results are applicable to all modeled behaviors of the environment vehicles and roads, not just one arbitrarily fixed behavior or road.

Law Set ($\mathcal{L}$): The laws are fixed in a given scenario and expressed in Metric Temporal Logic (MTL) (Koymans, 1990). In Figure 2, one law imposes a speed limit of 50 over for the duration T of the scenario: $l_1 = \text{Always}_{[0,T]}(v < 50)$. Other laws encode that the vehicle must not collide, $l_2 = \text{Always}_{[0,T]}(\text{distance}(a_1, a_2) > 0)$, and that the vehicle must stop at the stop sign, $l_3 = \text{Eventually}_{[0,T]}((\text{distance}(a_1, \textit{stop}) < 0) \wedge (v < 0.1))$.

Goal (Φ): The goal Φ of the ego vehicle is always of the form "Ego must reach some region within T time units". In Fig. 2, the goal region is the green rectangle, expressing that the vehicle should turn onto the main road within some bounded time horizon.

Initialization ($Init$): An AV estimates its state x_0 to within some bounded error (because of measurement imprecision), and so it only knows that x_0 is in some set $Init_a$. Thus, it is necessary at verification time to verify that whatever actual value x_0 has in $Init_a$, the scenario will not lead to a collision or to a requirement violation. The set $Init$ is the product of all vehicles' initial sets: $Init = \Pi_{a \in A} Init_a$. The initial sets of the 2 cars are shown in light grey in Figure 2.

Exit Condition ($\mathcal{E}$): Finally, the scenario ends either when the ego vehicle leaves the region defined by the T-Junction (the green region in Figure 2) and proceeds to the next navigation task, or a timeout occurs (*e.g.* the vehicle is stuck in the red region in Figure 2). The SDL enables the user to describe these elements of a scenario in a consistent and structured

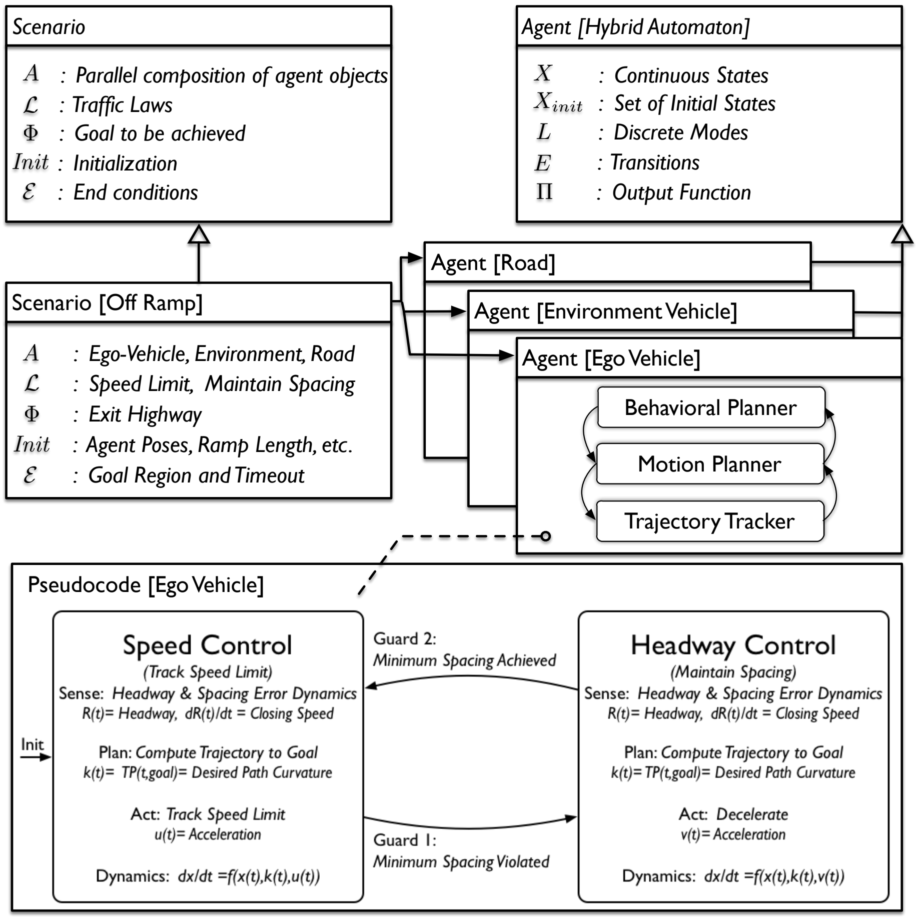

Scenario
A : Parallel composition of agent objects
$\mathcal{L}$: Traffic Laws
Φ : Goal to be achieved
$Init$: Initialization
$\mathcal{E}$: End conditions
Agent [Hybrid Automaton]
X : Continuous States
X_{init} : Set of Initial States
L : Discrete Modes
E : Transitions
Π : Output Function
Scenario [Off Ramp]
A : Ego-Vehicle, Environment, Road
$\mathcal{L}$: Speed Limit, Maintain Spacing
Φ : Exit Highway
$Init$: Agent Poses, Ramp Length, etc.
$\mathcal{E}$: Goal Region and Timeout
Agent [Road]
Agent [Environment Vehicle]
Agent [Ego Vehicle]
Behavioral Planner
Motion Planner
Trajectory Tracker
Pseudocode [Ego Vehicle]
Speed Control
(Track Speed Limit)
Sense: Headway & Spacing Error Dynamics
R(t)= Headway, dR(t)/dt = Closing Speed
Plan: Compute Trajectory to Goal
k(t)= TP(t,goal)= Desired Path Curvature
Act: Track Speed Limit
u(t)= Acceleration
Dynamics: dx/dt =f(x(t),k(t),u(t))
Init
Guard 2:
Minimum Spacing Achieved
Guard 1:
Minimum Spacing Violated
Headway Control
(Maintain Spacing)
Sense: Headway & Spacing Error Dynamics
R(t)= Headway, dR(t)/dt = Closing Speed
Plan: Compute Trajectory to Goal
k(t)= TP(t,goal)= Desired Path Curvature
Act: Decelerate
v(t)= Acceleration
Dynamics: dx/dt =f(x(t),k(t),v(t))

Figure 3: Scenario structure. An unfilled arrow indicates class inheritance. The Off-ramp *scenario automaton* is the product of all *vehicle agents'* automata.

manner, and handles many low-level details like maintaining a global clock, monitoring for important events like scenario end, type and dimensionality checking, and sharing of variables to model one AV perceiving another.

To perform rigorous reachability analysis on the scenario (an approach to verification), it is necessary to map the scenario description to a *formal* internal representation of the the agents in the scenario, i.e. a representation with unambiguous mathematical semantics. While several AV simulators exist with more complex models than presented above (*e.g.* rendering engines, and sensors), the fact that they lack this internal formal representation means that it is not possible to run any formal tools on them. Agents have both discrete switching dynamics in the BP and continuous dynamics in the TP and TT. Thus the appropriate formal model is the *hybrid automaton* (Alur et al., 1995). Every agent in a scenario is modeled as a hybrid automaton, and the overall scenario model is the product of all these automata, as shown in Figure 3. Further details are given in Section 2.6.1.

2.5. Tools

In chapter we investigate two formal techniques for AV evaluation: falsification and verification via bounded reachability analysis. In both approaches, for a given scenario, we wish to understand whether there exist conditions, such as initial positions and velocities of all vehicles, and particular curvatures of the road, that lead to a violation of the (MTL) requirement φ.

First we consider falsification. The scenario description language presented in Section 2.4 enables the translation of the symbolic scenario description to a format which enables trajectories to be sampled by S-TaLiRo (Annapureddy and Fainekos, 2010). S-TaLiRo is a tool for automatic test generation for cyber-physical systems. S-TaLiRo searches X_0 for an x_0 that yields a trajectory $\mathbf{x}$ of minimum robustness $\rho_\varphi(\mathbf{x})$. If $\rho_\varphi(\mathbf{x})$ is negative, then $\mathbf{x}$ is actually a requirement violation. The results are then returned to the designer to visualize and debug. If a violating x_0 exists, then S-TaLiRo will find it in the long run with probability approaching 1. In practice, S-TaLiRo can find errors more quickly than verification methods, as will be shown in the experiments. It is important to stress that the complexity and hybrid nature of an AV scenario requires testing tools, like S-TaLiRo, that go beyond traditional randomized testing for software. As explained in Section 2.3, S-TaLiRo computes the robustness of the scenario, which allows us to estimate how badly a requirement is violated (or how well it is satisfied) enabling low robustness samples to both guide the selection of interesting test cases, and also be reported for further analysis.

The scenario description language can also be translated to hybrid automaton in a format that can be processed by dReach (Kong et al., 2015b), which answers the verification question *exhaustively.* Namely: if dReach returns that the scenario is SAFE, then no initial condition exists for which the model violates the specification, and if it returns that it is δ-UNSAFE, it provides an example of a violation for debugging. If a scenario is δ-*UNSAFE*, then this means that a δ-sized perturbation of the scenario's trajectories can reach a slightly expanded unsafe set. In fact there is a strong theoretical connection between δ-satisfiability and the robustness of the scenario (*cf.* Abbas et al., 2017); however, for brevity we omit further discussion here. In practice, such a system should be considered as unsafe, and its design made more robust. Unlike S-TaLiRo, which might be interrupted before finding a violation that does exist, dReach's answer is definitive, but it usually runs for much longer and scales poorly with the complexity of the scenario being evaluated. The properties of the falsification and exhaustive verification methodologies described above are complementary, and it what follow we describe and leverage their individual strengths.

2.6. Experiments

In this section we briefly introduce and expand on the lane change scenario introduced in Section 2.2 and check its safety using a bounded time reachability analysis. Next, we consider the more complex T-Junction scenario described in Section 2.4 and apply both falsification and reachability tools to analyze its safety.

2.6.1. Case Study: Simple Lane Change Controller

The following example describes a lane change scenario in the context of a mission and mobility goals. We seek to verify both that all possible individual trajectories which occur in the *execution* of the plan are safe and that the mobility goal is achieved. We consider two variants of the lane change scenario shown in Figure 4. In *Scenario 1* we will demonstrate a dangerous condition that could have been missed under testing or simulation. In *Scenario 2* we will show how a refinement in the *requirements* on the perception system as well as a more conservative behavioral controller can lead to a provably safe maneuver.

We first outline the architecture of the ego vehicle software. The planning and control of the vehicle is hierarchical in nature. Each successive layer performs a task over a shorter time horizon. At the top level a mission planner is given a mobility goal. Such a goal is typically expressed as a (location, destination) pair. Given this pair the mission planner finds an optimal (or feasible) route through the road network. In the next layer, the behavioral planner makes local decisions about how to navigate the road network. For example, if the mission planner informs the behavioral planner that at the next intersection it will need to turn left, the behavioral planner will use a set of rules to determine that the ego vehicle must be in the left lane. The behavior planner then provides a sequence of waypoints, or intermediary destinations, to the lower-level local planner. Finally, the local planner produces a trajectory that connects the vehicle's current pose to the target pose at the next waypoint. Here pose refers to the combined position, heading, and velocity of the vehicle. Specifically, given a goal pose relative to the vehicle's current pose, the local planner computes a set of candidate smooth trajectories that can lead to the goal pose or near it, then selects a single trajectory and sends it to the vehicle. The vehicle then uses a PID controller to track the selected trajectory. Our objective is to verify whether a model (see O'Kelly et al., 2016, for additional details) of the AV system described above successfully completes a lane change without crashing for a given set of initial conditions.

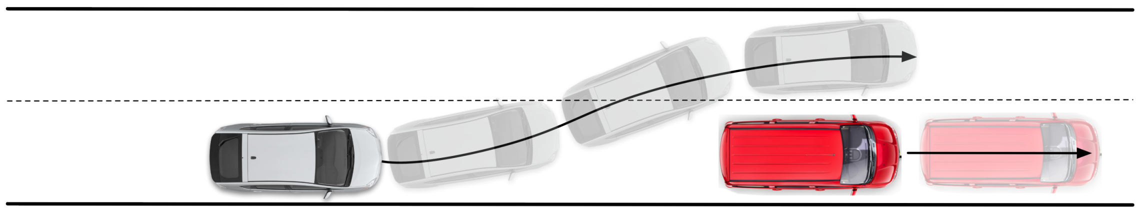
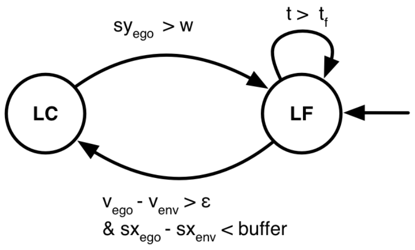


Figure 4: A lane change scenario. This scenario is unsafe for certain inter-vehicle spacings and velocities. Reachability analysis is used to identify these conditions and update requirements for the state estimation system and vehicle controllers.

Scenario 1 (A simple lane change and goal)**.** *As shown in Figure 4, the ego vehicle is driving in the right lane of a unidirectional two lane road network. Another car is driving in front of the ego vehicle at a lower speed. We include the extreme case where the environmental vehicle stops. We highlight that when there is significant uncertainty regarding the ego vehicle's orientation and that it may deviate (initially) from the reference trajectory while the tracking* *controller recovers.*

Table 1 describes the parameterization of the scenario and Figure 5 details the behavior controller, where LC means “Lane Change" and LF means “Lane Follow". When the ego vehicle approaches the lead car with a positive difference in speed it enters the LC mode and tracks a lane change trajectory. When the lane change has been completed the ego vehicle returns to lane following mode. Using dReach we *attempt* to show that there is no execution of the system which can lead to a collision. However, because the system is incorrectly designed dReach returns δ-UNSAFE and a counterexample.



$sy_{ego} > w$
$t > t_f$
LC
LF
$v_{ego} - v_{env} > \varepsilon$
$\& \ sx_{ego} - sx_{env} < buffer$

Figure 5: An automaton describing a simplistic behavior planner for lane changes

The counterexample is generated by a configuration of the scenario in which the environment vehicle stops in the lane and the ego vehicle's initial state estimate is poor. The combination of the sudden stopping behavior of the lead car and large initial trajectory tracking error cause the ego vehicle to clip the rear left bumper of the other vehicle.

Using this counterexample to inform a refinement of the scenario, we adjust the behavior controller to require a greater separation between the ego and environment vehicle. In addition we posit an abstract improvement to the state estimation system such that the ego vehicles initial pose is more accurate.

Scenario 2 (A more conservative behavioral controller)**.** *We begin with Scenario 1. In order to ensure the forward safety of the vehicle we propose a small modification to the behavioral controller of the vehicle, and furthermore require that the ego vehicle's state estimation system return estimates with less uncertainty. Namely, we first increase the size of the buffer* *between the ego vehicle and environment, so that the ego vehicle is forced to initiate a lane change maneuver earlier. Secondly, we also decrease the size of the state estimation error. Speed, v, now starts anywhere in* $[10.9, 11]$ *and lateral position, s_y, starts in [0.0,0.05].*

With these changes, dReal returns SAFE, meaning that no trajectory of the system violates the constraints. The result of each scenario's verification run is provided in Table 1. To perform verification, we employ dReach version 3.15.10.02 on a Mac OSX laptop with Intel(R) Core i7(R) 2.60GHz CPU and 16 GB memory.

Table 1: Lane change scenario verification results

Symbol	Scenario 1	Scenario 2
Lane Width, w	3.7	3.7
Buffer, B	15	20
Numerical Tolerance, δ	0.1	0.1
Speed, v_{ego}	[10.8, 11.1]	[10.8, 11]
Lead Vehicle Speed, v_{env}	[0.0,11.1]	[0.0,11.1]
Longitudinal Position, $s_{x_{ego}}$	[0.0, 0.5]	[0.0, 0.5]
Lateral Position, $s_{y_{ego}}$	[0.0, 0.1]	[0.0, 0.05]
Orientation, Ψ	[0.0, 0.1]	[0.0, 0.1]
Verification Time (s)	30.821	373.924
Result	δ-UNSAFE	SAFE

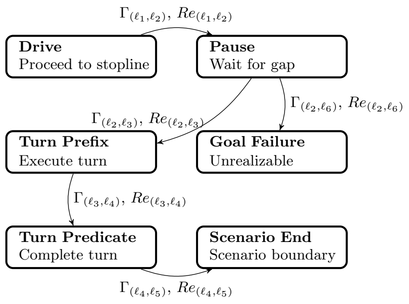
There are several important facts to note about these verification results. First, the effects of perception errors enter the model of the AV in an abstract way via non-determinism. Both the complexity of the perception components themselves (*e.g.* large neural networks), and the challenge of creating a formal model of the image formation process (*e.g.* real-time rendering engines) limit the use of verification tools on AV models which include perception. Second, in order to verify the lane change scenario it was necessary to make the ego vehicle behave more conservatively. In Part II of this thesis we explore tradeoffs like this in a principled manner. Third, as the number of modes in the behavior control layer grows the complexity of bounded reachability analysis can grow exponentially because the tool must explore each possible mode sequence. The discrete modes in the lane change scenario are extremely simple limiting the number of possible mode sequences. Empirically (in this and other experiments), we have observed two regimes for verification. In the first regime the computation time behaves like a random variable drawn from a bi-modal distribution with one mode for unsafe scenarios and another for safe scenarios. In the second regime, scenarios of sufficient complexity time out without ever returning an answer. In one instance a we allowed a verification run to continue for over a month with no progress.

2.6.2. Case Study: T-Junction

The second case study investigates a more complex T-Junction scenario, illustrated in Figure 2. It demonstrates the challenges of modeling the environment vehicles without being overly restrictive or overly permissive. Specifically, it reveals that the non-deterministic dynamics of the environment vehicle may be under-constrained, allowing it to rear-end the ego-vehicle, and causing the verification process to terminate with a counterexample.

Scenario 3 (T-Junction)**.** *The T-Junction instance encodes a driving scenario in which the ego vehicle reaches a stop sign and must turn right onto a two lane bidirectional road. The oncoming traffic is not required to stop. Thus, the ego vehicle must first come to a stop and then execute a sequence of trajectories such that it reaches the goal state of the scenario without colliding with any other object or traffic.*

Table 2 describes the parameterization of the scenario, and Figure 6 details the behavior controller. The scenario contains three agents: an ego vehicle, an environment vehicle, and a T-Junction road network with a stop sign. The ego vehicle begins in mode `Drive` (ℓ_1), traveling towards the stop sign obeying relevant traffic laws. When it reaches the region where the roads connect it enters mode `Pause` (ℓ_2) and comes to a stop. Once there is a sufficiently large gap in the traffic the ego vehicle begins the turn right (mode `Turn Prefix`, ℓ_3). Finally, when it has reached the main road it switches to mode `Turn Predicate` (ℓ_4) and aligns itself with the centerline. We begin our analysis of the T-Junction scenario by running the falsification tool S-TaLiRo with the goal of quickly finding and understanding specification violations.


$\Gamma_{(\ell_1,\ell_2)}$, $Re_{(\ell_1,\ell_2)}$
Drive
Proceed to stopline
Pause
Wait for gap
$\Gamma_{(\ell_2,\ell_3)}$, $Re_{(\ell_2,\ell_3)}$
$\Gamma_{(\ell_2,\ell_6)}$, $Re_{(\ell_2,\ell_6)}$
Turn Prefix
Execute turn
Goal Failure
Unrealizable
$\Gamma_{(\ell_3,\ell_4)}$, $Re_{(\ell_3,\ell_4)}$
Turn Predicate
Complete turn
Scenario End
Scenario boundary
$\Gamma_{(\ell_4,\ell_5)}$, $Re_{(\ell_4,\ell_5)}$

State	*Transition*	*Transition*
Drive (ℓ_1)	**Guard** $\Gamma_{(\ell_1,\ell_2)}$: $s_x \geq s_{x_{stop}}$ **Reset** $Re_{(\ell_1,\ell_2)}$: $\mathbf{I}'_{stop} = 1, t' = 0$ **Next State:** Pause	**Guard**: NA **Reset**: NA **Next State:** NA
Pause (ℓ_2)	**Guard** $\Gamma_{(\ell_2,\ell_3)}$: $(t > t_{pause}) \wedge (d_{gap} > d_{min})$ **Reset** $Re_{(\ell_2,\ell_3)}$: $\mathbf{I}'_{drive} = 0, \mathbf{I}'_{stop} = 0, t' = 0$ **Next State:** Turn Prefix	**Guard**: $\Gamma_{(\ell_2,\ell_6)}$ $(t > t_f) \wedge (\ell = 2)$ **Reset**: NA **Next State:** Goal Failure
Turn Prefix (ℓ_3)	**Guard** $\Gamma_{(\ell_3,\ell_4)}$: $s_y < s_{f_{y_1}}$ **Reset** $Re_{(\ell_3,\ell_4)}$: $s'_{x_0} = s_x, s'_{y_0} = s_y, s'_{ego} = 0$ $s'_{x_{goal}} = wp_{x_1}, s_{y_{goal}} = wp_{y_1}$ $\mathbf{I}'_{pre} = 0, \mathbf{I}'_{pred} = 1$ **Next State:** Turn Predicate	**Guard**: NA **Reset**: NA **Next State:** NA
Turn Predicate (ℓ_4)	**Guard** $\Gamma_{(\ell_4,\ell_5)}$: $s_y < s_{f_{y_2}}$ **Reset** $Re_{(\ell_4,\ell_5)}$: $s'_{x_0} = s_x, s'_{y_0} = s_y, s'_{ego} = 0$ $s'_{x_{goal}} = wp_{x_2}, s_{y_{goal}} = wp_{y_2}$ $\mathbf{I}'_{pred} = 1, t' = 0$ **Next State:** Scenario Complete	**Guard**: NA **Reset**: NA **Next State:** NA

Figure 6: T-Junction: scenario automaton (left) and corresponding guards and resets (right).

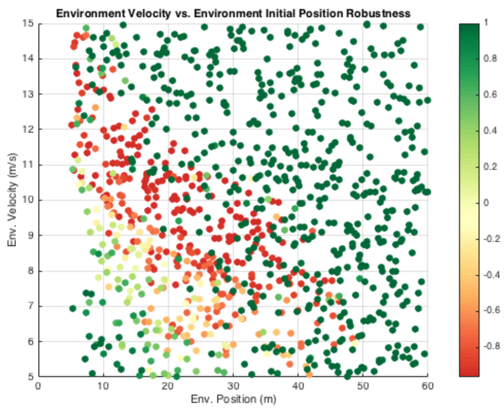


Figure 7: Analysis of samples generated by robust testing of the T-Junction scenario that the behavioral controller design fails in cases where the environment vehicle starts relatively far away at a velocity similar to the speed limit and accelerates into the back of the ego-vehicle (1,000 runs)

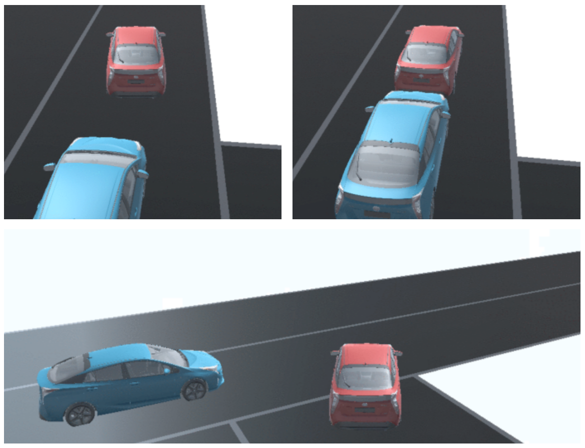


Figure 8: Scenario 3. *Top*: collision due to under-constrained environment vehicle. *Bottom*: collision due to incorrect transitions.

Table 2: Verification Results for the T-Junction Scenario. δ-UNSAFE means a δ-sized perturbation leads to collisions

State Variable	Description	Controller Error	Corrected Controller	Additional Uncertainty
$s_{x_{env}}$	Environment X-position m	[8,60]	[8,60]	[8,60]
v_{env}	Environment Velocity	[5,15]	[5,15]	[5,15]
$\dot{v}_{env} \in F(\ell)$	Environment Velocity Noise	[-0.5,0.5]	[-0.5,0.5]	[-0.5,0.5]
$\dot{s}_{x_{env}} \in F(\ell)$	Environment X-Position Noise	[0, 0]	[0,0]	[-0.5,0.5]
Runtime (s)	2.6 GHz Intel Core i7, 16 GB RAM	83.622	41.985	542.344
Result	$\delta = 0.1$	δ-UNSAFE	SAFE	SAFE

Figure 7 shows sampled robustness values as a function of the initial environment vehicle position and velocity. The figure illustrates that incorrect behaviors (yellow-to-red dots) are interspersed in-between correct behavior (green dots). Thus, the testing of complex systems cannot simply be a matter of testing the so-called corner cases or extreme configurations. In contrast, the specification-guided testing methodology incorporated in S-TaLiRo is able to efficiently identify such failure modes. The failure modes found by S-TaLiRo can be traced back to the dynamics of the environment: the latter could accelerate behind the ego vehicle and rear-end it (see the visualization in the top panel of Figure 8). In an accident of this nature, the liability usually but not always, falls on the trailing vehicle's operator rather than the ego vehicle being tested. For the purposes of verifying this scenario we will assume that this behavior is undesirable, but in Chapters 3 and 4 we develop methods which do not require such subjective judgments of blame.

This undesirable environment behavior can be excised by adding a new exit condition, which is triggered before the environment vehicle rear-ends the ego vehicle, to the scenario automaton (the composition of all agents' hybrid automata). Following this update, another failure mode is identified by S-TaLiRo (see the bottom panel of Figure 8). In this case, the ego vehicle fails to yield to an accelerating vehicle. While dReach is capable of identifying this failure mode in approximately 84 seconds (Table 2, 3rd column), S-TaLiRo produces many variations of the crash in only 8 seconds. In order to correct this failure, the guard in `Pause` is refined to ensure that there is a large enough gap for the ego vehicle to complete the maneuver.

Upon updating the ego vehicle's controller, dReach exhaustively verified the scenario to be SAFE in 41.98 seconds (Table 2 column 'Corrected Controller'.) Additionally, dReach certified the scenario as SAFE even with added non-determinism in the environment vehicle's dynamics. Still there are serious limitations to scaling verification due to its computational complexity; as expected, additional T-Junction experiments with larger initial sets or more agents all timed out without returning any results.

2.7. Conclusions

In summary, for safety and correctness certification, AV manufacturers will have to provide rigorous and transparent guarantees (Arnold and Scheutz, 2016) that the AV will perform correctly under a number of scenarios, regardless of initial configuration. Where applicable, formal methods can provide such rigorous guarantees. Thus, in this work we developed a tool chain for performing safety assessments of AVs via both verification and falsification techniques. The case studies show that scenarios which capture AV operational domains can be analyzed as hybrid systems; however, they also serve to demonstrate the limitations of such approaches.

For the AV verification problem, formal methods are subject to fundamental computational barriers. This fact means that current tools are limited to very short time-horizons, during which a crash might not arise, or to handling individual components (Nilsson et al.), discretized dynamics (Humphrey et al., 2014), or highly abstract models (Loos et al., 2011) all of which may result in overly conservative conclusions (Pavlic et al., 2011) and doubtful applicability to the physical car.

More specifically, a primary concern of all model-based design methods is that any evidence gathered is a statement about the model, which may differ significantly from the real system. Although this reality-gap is not exclusive to the verification-based methods, it is exacerbated in the verification setting. Put plainly, the actual artifacts which constitute AV software and

AV simulation environments are not written in a formalism suitable for verification tools. While it may technically possible to express a rendering engine, optimization routine, or sensor model as a hybrid system, the discrete modes of the scenario object would be so complex that verification is unrealistic. Thus, abstraction of key AV components, particularly perception, trajectory generation, and localization systems, is more than just convenience: it is necessary for any practical verification effort.

In the falsification regime, modeling decisions still result in a gap between the evidence generated by the toolchain and performance in reality; however, the nature and cause of this gap is quite different. Falsification approaches only require that we can sample an execution of the system; the structure of the system model itself is not integral to the analysis method, as it is in verification. Thus, there is no need to abstract complex software modules in the simulator or the AV stack; the real implementation can still be used. The primary difficultly is instead the fidelity of the model for the "physical" portion of the system, *e.g.* the vehicle dynamics, behaviors of human actors, and the textures and geometry of the world. The "cyber" portion of the model, that is the software controlling the system under test, is largely untouched. Therefore, in terms of modeling effort and model expressivity there are clear benefits to the falsification-based family of approaches. Thus, in the remainder of this thesis, maintaining the ability to consider the system-under-test as a black-box from which simulated executions may be sampled will be a requirement of methods which we propose.

Despite the need for abstraction in the verification regime, it has many favorable characteristics. When a verification method terminates, we know either that the model satisfies its specification, or that there exists a counterexample. In contrast falsification-based methods can make no such assurance, and the computational budgets which define their run-time are arbitrary. While the tools we utilize in this chapter for verification do not "simulate" the system-under-test explicitly, precluding a measurement of efficiency in terms of number of simulations, we do know that they will terminate. Therefore, if we are to seek methods which can analyze black-box system models (as in falsification) we would like to add an additional requirement that the number of simulations necessary to terminate the run be quantifiable.

Still, the verification and falsification approaches we have presented lack a key capability. In both regimes we do not recover the likelihood of a counterexample. The reasons are two-fold: first, the representations of the system-environment model are impoverished, non-determinism is used to express uncertainty about executions of the system rather than defining a probability space. From a formal modeling perspective the required changes are small. Instead of considering a hybrid automaton, we can define the underlying structure of the system-under-test to be a hybrid automaton with parametric uncertainty (Wang et al., 2015). While we will not explicitly write models in this formalism in the remainder of this thesis, the underlying assumption is that we are able to sample executions from such an object.

Updating the model to include a probabilistic description of the initial conditions of the system does little to change the situation if the verifier or falsifier are still tasked with searching for the existence of a failure. In this case, positing a uniform distribution over initial conditions would have no different result than Gaussian. Instead, we must also change the problem we are trying to solve from: “Does there exist a failure?” to “What is the probability of a failure?”. It is from this set of desiderata that we begin Chapter 3.

CHAPTER 3 : Risk-based Framework

3.1. Chapter Overview

This chapter is adapted from "Scalable End-to-end Autonomous Vehicle Testing via Rare-event Simulation", which appeared in the Proceedings of Neural Information Processing Systems 2018 as joint work with Aman Sinha, Hongseok Namkoong, John Duchi, and Russ Tedrake.

3.2. Introduction

Recent breakthroughs in deep learning have accelerated the development of autonomous vehicles (AVs); many research prototypes now operate on real roads alongside human drivers. While advances in computer-vision techniques have made human-level performance possible on narrow perception tasks such as object recognition, several fatal accidents involving AVs underscore the importance of testing whether the perception and control pipeline—when considered as a *whole system*—can safely interact with humans. Unfortunately, testing AVs in real environments, the most straightforward validation framework for system-level input-output behavior, requires prohibitive amounts of time due to the rare nature of serious accidents (Shalev-Shwartz et al., 2017). Concretely, a recent study (Kalra and Paddock, 2016) argues that AVs need to drive "hundreds of millions of miles and, under some scenarios, hundreds of billions of miles to create enough data to clearly demonstrate their safety." Alternatively, formally verifying an AV algorithm's "correctness" (Kwiatkowska et al., 2011; Althoff and Dolan, 2014; Seshia et al., 2015; O'Kelly et al., 2016) is difficult since all driving policies are subject to crashes caused by other drivers (Shalev-Shwartz et al., 2017). It is unreasonable to ask that the policy be safe under *all* scenarios. Unfortunately, ruling out scenarios where the AV should not be blamed is a task subject to logical inconsistency, combinatorial growth in specification complexity, and subjective assignment of fault.

Motivated by the challenges underlying real-world testing and formal verification, we consider a probabilistic paradigm—which we call a *risk-based framework*—where the goal is to evaluate the *probability of an accident* under a base distribution representing standard traffic behavior. By assigning learned probability values to environmental states and agent behaviors, our risk-based framework considers performance of the AV's policy under a data-driven model of the world. To efficiently evaluate the probability of an accident, we implement a photo-realistic and physics-based simulator that provides the AV with perceptual inputs (*e.g.* video and range data) and traffic conditions (*e.g.* other cars and pedestrians). The simulator allows parallelized, faster-than-real-time evaluations in varying environments (*e.g.* weather, geographic locations, and aggressiveness of other cars).

Formally, we let P_0 denote the base distribution that models standard traffic behavior and $X \sim P_0$ be a realization of the simulation (*e.g.* weather conditions and driving policies of other agents). For an objective function $f : \mathcal{X} \to \mathbb{R}$ that measures "safety"—so that low values of $f(x)$ correspond to dangerous scenarios—our goal is to evaluate the probability of a dangerous event

$$p_\gamma := \mathbb{P}_0(f(X) \le \gamma) \tag{3.1}$$

for some threshold γ. Our risk-based framework is agnostic to the complexity of the ego-policy and views it as a black-box module. Such an approach allows, in particular, deep-learning based perception systems that make formal verification methods intractable.

An essential component of this approach is to estimate the base distribution P_0 from data; we use public traffic data collected by the US Department of Transportation (US Department of Transportation – FHWA, 2008). While such datasets do not offer insights into how AVs interact with human agents—this is precisely why we design our simulator—they illustrate the range of standard human driving behavior that the base distribution P_0 must model. We use imitation learning (Russell, 1998; Ross and Bagnell, 2010; Ross et al., 2011; Ho and Ermon, 2016; Baram et al., 2017) to learn a generative model for the behavior (policy) of environment vehicles; unlike traditional imitation learning, we train an ensemble of models to characterize a distribution of human-like driving policies.

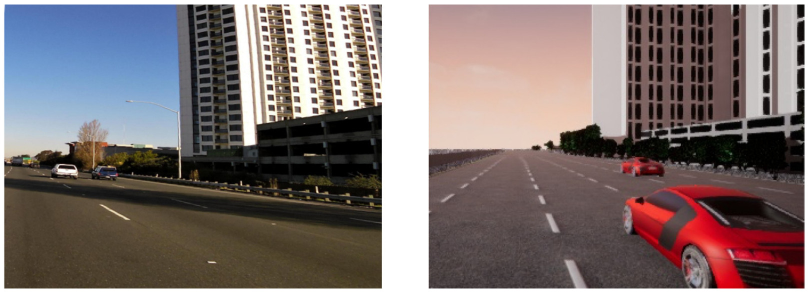


Figure 9: Multi-lane highway driving on I-80: (left) real image, (right) rendered image from simulator

As serious accidents are rare (p_γ is small), we view this as a *rare-event simulation* (Asmussen and Glynn, 2007) problem; naive Monte Carlo sampling methods require prohibitively many simulation rollouts to generate dangerous scenarios and estimate p_γ. To accelerate safety evaluation, we use adaptive importance-sampling methods to learn alternative distributions P_θ that generate accidents more frequently. Specifically, we use the cross-entropy algorithm (Rubinstein and Kroese, 2004) to iteratively approximate the optimal importance sampling distribution. In contrast to simple classical settings (Rubinstein and Kroese, 2004; Zhao et al., 2018) which allow analytic updates to P_θ, our high-dimensional search space requires solving convex optimization problems in each iteration (Section 3.3). To address numerical instabilities of importance sampling estimators in high dimensions, we carefully design search spaces and perform computations in logarithmic scale. Our implementation produces 2-20 times as many rare events as naive Monte Carlo methods, independent of the complexity of the ego-policy.

In addition to accelerating evaluation of p_γ, learning a distribution P_θ that *frequently* generates realistic dangerous scenarios $X_i \sim P_\theta$ is useful for engineering purposes. The importance-sampling distribution P_θ not only efficiently samples dangerous scenarios, but also ranks them according to their likelihoods under the base distribution P_0. This capability enables a deeper understanding of failure modes and prioritizes their importance to improving the ego-policy.

As a system, our simulator allows fully distributed rollouts, making our approach orders of magnitude cheaper, faster, and safer than real-world testing. Using the asynchronous messaging library ZeroMQ (Hintjens, 2013), our implementation is fully-distributed among available CPUs and GPUs; our rollouts are up to 30P times faster than real time, where P is the number of processors. Combined with the cross-entropy method's speedup, we achieve 10-300P speedup over real-world testing.

In what follows, we describe components of our open-source toolchain, a photo-realistic simulator equipped with our data-driven risk-based framework and cross-entropy search techniques. The toolchain can test an AV as a *whole system*, simulating the driving policy of the ego-vehicle by viewing it as a black-box model. The use of adaptive-importance sampling methods motivates a unique simulator architecture (Section D.5) which allows real-time updates of the policies of environment vehicles. In Section 3.5, we test our toolchain by considering an end-to-end deep-learning-based ego-policy (Bojarski et al., 2016) in a multi-agent highway scenario. Figure 9 shows one configuration of this scenario in the real world along with rendered images from the simulator, which uses Unreal Engine 4 (Games, 2015). Our experiments show that we accelerate the assessment of rare-event probabilities with respect to naive Monte Carlo methods as well as real-world testing. We believe our open-source framework is a step towards a rigorous yet scalable platform for evaluating AV systems, with the broader goal of understanding how to reliably deploy deep-learning systems in safety-critical applications.

3.3. Rare-event simulation

To motivate our risk-based framework, we first argue that formally verifying correctness of a AV system is infeasible due to the challenge of defining "correctness." Consider a scenario where an AV commits a traffic violation to avoid collision with an out-of-control truck approaching from behind. If the ego-vehicle decides to avoid collision by running through a red light with no further ramifications, is it "correct" to do so? The "correctness" of the policy depends on the extent to which the traffic violation endangers nearby humans and whether any element of the "correctness" specification explicitly forbids such actions. That is, “correctness” as a binary output is a concept defined by its exceptions, many elements of which are subject to individual valuations (Bonnefon et al., 2016).

Instead of trying to verify correctness, we begin with a continuous measure of safety $f : \mathcal{X} \to \mathbb{R}$, where $\mathcal{X}$ is space of traffic conditions and behaviors of other vehicles. The prototypical example in this paper is the minimum time-to-collision (TTC) (see Appendix A.1 for its definition) to other environmental agents over a simulation rollout. Rather than requiring safety for all $x \in \mathcal{X}$, we relax the deterministic verification problem into a probabilistic one where we are concerned with the probability under standard traffic conditions that $f(X)$ goes below a safety threshold. Given a distribution P_0 on $\mathcal{X}$, our goal is to estimate the rare event probability $p_\gamma := P_0(f(X) \le \gamma)$ based on simulated rollouts $f(X_1), \ldots, f(X_n)$. As accidents are rare and p_γ is near 0, we treat this as a rare-event simulation problem; see (Bucklew, 2013; Asmussen and Glynn, 2007, Chapter VI) for an overview of this topic.

First, we briefly illustrate the well-known difficulty of naive Monte Carlo simulation when p_γ is small. From a sample $X_i \overset{\text{i.i.d.}}{\sim} P_0$, the naive Monte Carlo estimate is

$$\widehat{p}_{N,\gamma} := \frac{1}{N} \sum_{i=1}^{N} \mathbf{1}\left\{f(X_i) \le \gamma\right\}.$$

As p_γ is small, we use relative accuracy to measure our performance, and the central limit theorem implies the relative accuracy is approximately

$$\left|\frac{\widehat{p}_{N,\gamma}}{p_\gamma} - 1\right| \overset{\text{dist}}{\approx} \sqrt{\frac{1-p_\gamma}{Np_\gamma}}\,|Z| + o(1/\sqrt{N}) \;\text{ for } Z \sim \mathsf{N}(0,1).$$

For small p_γ, we require a sample of size $N \gtrsim 1/(p_\gamma \epsilon^2)$ to achieve ϵ-relative accuracy, and if $f(X)$ is light-tailed, the sample size must grow exponentially in γ.

Cross-entropy method As an alternative to a naive Monte Carlo estimator, we consider (adaptive) importance sampling (Asmussen and Glynn, 2007), and we use a model-based optimization procedure to find a good importance-sampling distribution. The optimal importance-sampling distribution for estimating p_γ has the conditional density $p^\star(x) = \mathbf{1}\{f(x) \le \gamma\}\, p_0(x)/p_\gamma$, where p_0 is the density function of P_0: as $p_0(x)/p^\star(x) = p_\gamma$ for all x satisfying $\mathbf{1}\{f(x) \le \gamma\}$, the estimate $\widehat{p}^\star_{N,\gamma} := \frac{1}{N}\sum_{i=1}^N \frac{p_0(X_i)}{p^\star(X_i)}\mathbf{1}\{f(X_i) \le \gamma\}$ is exact. This sampling scheme is, unfortunately, *de facto* impossible, because we do not know p_γ. Instead, we use a parameterized importance sampler P_θ and employ an iterative model-based search method to modify θ so that P_θ approximates $P^\star$.

The cross-entropy method (Rubinstein and Kroese, 2004) iteratively tries to find $\theta^\star \in \operatorname{argmin}_{\theta\in\Theta} D_{\text{kl}}\left(P^\star \| P_\theta\right)$, the Kullback-Leibler projection of $P^\star$ onto the class of parameterized distributions $\mathcal{P} = \{P_\theta\}_{\theta\in\Theta}$. Over iterations k, we maintain a surrogate distribution $q_k(x) \propto \mathbf{1}\{f(x) \le \gamma_k\}\, p_0(x)$ where $\gamma_k \ge \gamma$ is a (potentially random) proxy for the rare-event threshold γ, and we use samples from P_θ to update θ as an approximate projection of Q onto $\mathcal{P}$. The motivation underlying this approach is to update θ so that P_θ upweights regions of $\mathcal{X}$ with low objective value (*i.e.* unsafe) $f(x)$. We fix a quantile level $\rho \in (0,1)$—usually we choose $\rho \in [0.01, 0.2]$—and use the ρ-quantile of $f(X)$ where $X \sim P_{\theta_k}$ as γ_k, our proxy for the rare event threshold γ (see Homem-de Mello (2007) for alternatives). We have the additional challenge that the ρ-quantile of $f(X)$ is unknown, so we approximate it using i.i.d. samples $X_i \sim P_{\theta_k}$. Compared to applications of the cross-entropy method (Rubinstein and Kroese, 2004; Zhao et al., 2018) that focus on low-dimensional problems permitting analytic updates to θ, our high-dimensional search space requires solving convex optimization problems in each iteration. To address numerical challenges in computing likelihood ratios in high-dimensions, our implementation carefully constrains the search space and we compute likelihoods in logarithmic scale.

We now rigorously describe the algorithmic details. First, we use natural exponential families as our class of importance samplers $\mathcal{P}$.

Definition 1. *The family of density functions $\{p_\theta\}_{\theta\in\Theta}$, defined with respect to base measure μ, is a* natural exponential family *if there exists a sufficient statistic Γ such that $p_\theta(x) = \exp(\theta^\top\Gamma(x) - A(\theta))$ where $A(\theta) = \log\int_{\mathcal{X}}\exp(\theta^\top\Gamma(x))d\mu(x)$ is the log partition function and*

Algorithm 1 Cross-Entropy Method

```
1: Input: Quantile $\rho \in (0,1)$, Stepsizes $\{\alpha_k\}_{k\in\mathbb{N}}$, Sample sizes $\{N_k\}_{k\in\mathbb{N}}$, Number of iterations $K$
2: Initialize: $\theta_0 \in \Theta$
3: for $k = 0, 1, 2, \ldots, K-1$ do
4:     Sample $X_{k,1}, \ldots, X_{k,N_k} \overset{\text{i.i.d.}}{\sim} P_{\theta_k}$
5:     Set $\gamma_k$ as the minimum of $\gamma$ and the $\rho$-quantile of $f(X_{k,1}), \ldots, f(X_{k,N_k})$
6:     $\theta_{k+1} = \text{argmax}_{\theta\in\Theta} \left\{\alpha_k \theta^\top D_{k+1} + (1-\alpha_k)\theta^\top \nabla A(\theta_k) - A(\theta)\right\}$
```

$\Theta := \{\theta \mid A(\theta) < \infty\}$.

Given this family, we consider idealized updates to the parameter vector θ_k at iteration k, where we compute projections of a mixture of Q_k and P_{θ_k} onto $\mathcal{P}$

$$\begin{aligned}\theta_{k+1} &= \operatorname*{argmin}_{\theta\in\Theta} D_{\text{kl}}\left(\alpha_k Q_k + (1-\alpha_k)P_{\theta_k} \| P_\theta\right)\\ &= \operatorname*{argmax}_{\theta\in\Theta} \left\{\alpha_k \mathbb{E}_{Q_k}[\log p_\theta(X)] + (1-\alpha_k)\mathbb{E}_{\theta_k}[\log p_\theta(X)]\right\}\\ &= \operatorname*{argmax}_{\theta\in\Theta} \left\{\alpha_k \theta^\top \mathbb{E}_{Q_k}[\Gamma(X)] + (1-\alpha_k)\theta^\top \nabla A(\theta_k) - A(\theta)\right\}. \end{aligned} \tag{3.2}$$

The term $\mathbb{E}_{Q_k}[\Gamma(X)]$ is unknown in practice, so we use a sampled estimate. For

$$X_{k,1}, \ldots, X_{k,N_k} \overset{\text{i.i.d.}}{\sim} P_{\theta_k},$$

let γ_k be the ρ-quantile of $f(X_{k,1}), \ldots, f(X_{k,N_k})$ and define

$$D_{k+1} := \frac{1}{N_k}\sum_{i=1}^{N_k} \frac{q_k(X_{k,i})}{p_{\theta_k}(X_{k,i})}\Gamma(X_{k,i}) = \frac{1}{N_k}\sum_{i=1}^{N_k} \frac{p_0(X_{k,i})}{p_{\theta_k}(X_{k,i})}\mathbf{1}\left\{f(X_{k,i}) \le \gamma_k\right\}\Gamma(X_{k,i}). \tag{3.3}$$

Using the estimate D_{k+1} in place of $\mathbb{E}_{Q_k}[\Gamma(X)]$ in the idealized update (3.2), we obtain Algorithm 1. To select the final importance sampling distribution from Algorithm 1, we choose θ_k with the lowest ρ-quantile of $f(X_{k,i})$. We observe that this choice consistently improves performance over taking the last iterate or Polyak averaging. Letting θ_{ce} denote the parameters for the importance sampling distribution learned by the cross-entropy method, we sample $X_i \overset{\text{i.i.d.}}{\sim} P_{\theta_{\text{ce}}}$ and use $\widehat{p}_{N,\gamma} := \frac{1}{N}\sum_{i=1}^N \frac{p_0(X_i)}{p_{\theta_{\text{ce}}}(X_i)}\mathbf{1}\left\{f(X_i) \le \gamma\right\}$ as our final importance-sampling estimator for p_γ.

In the context of our rare-event simulator, we use a combination of Beta and Normal distributions for P_θ. The sufficient statistics Γ include (i) the parameters of the generative model of behaviors that our imitation-learning schemes produce and (ii) the initial poses and velocities of other vehicles, pedestrians, and obstacles in the simulation. Given a current parameter θ and realization from the model distribution P_θ, our simulator then (i) sets the parameters of the generative model for vehicle policies and draws policies from this model, and (ii) chooses random poses and velocities for the simulation. Our simulator is one of the largest-scale applications of cross-entropy methods.

3.4. Simulation framework

Two key considerations in our risk-based framework influence design choices for our simulation toolchain: (1) learning the base distribution P_0 of nominal traffic behavior via data-driven modeling, and (2) testing the AV as a *whole system*. We now describe how our toolchain achieves these goals.

3.4.1. Data-driven generative modeling

While our risk-based framework (cf. Section 3.3) is a concise, unambiguous measure of system safety, the rare-event probability p_γ is only meaningful insofar as the base distribution P_0 of road conditions and the behaviors of other (human) drivers is estimable. Thus, to implement our risk-based framework, we first learn a base distribution P_0 of nominal traffic behavior. Using the highway traffic dataset NGSim (US Department of Transportation – FHWA, 2008), we train policies of human drivers via imitation learning (Russell, 1998; Ross and Bagnell, 2010; Ross et al., 2011; Ho and Ermon, 2016; Baram et al., 2017). Our data consists of videos of highway traffic (US Department of Transportation – FHWA, 2008), and our goal is to create models that imitate human driving behavior even in scenarios distinct from those in the data. We employ an ensemble of generative adversarial imitation learning (GAIL) (Ho and Ermon, 2016) models to learn P_0. Our approach is motivated by the observation that reducing an imitation-learning problem to supervised learning—where we simply use expert data to predict actions given vehicle states—suffers from poor performance in regions of the state space not encountered in data (Ross and Bagnell, 2010; Ross et al., 2011). Reinforcement-learning techniques have been observed to improve generalization performance, as the imitation agent is able to explore regions of the state space in simulation during training that do not necessarily occur in the expert data traces.

Generically, GAIL is a minimax game between two functions: a discriminator D_ϕ and a generator G_ξ (with parameters ϕ and ξ respectively). The discriminator takes in a state-action pair (s, u) and outputs the probability that the pair came from real data, $\mathbb{P}(\text{real data})$. The generator takes in a state s and outputs a conditional distribution $G_\xi(s) := \mathbb{P}(u \mid s)$ of the action u to take given state s. In our context, $G_\xi(\cdot)$ is then the (learned) policy of a human driver given environmental inputs s. Training the generator weights ξ occurs in a reinforcement-learning paradigm with reward $-\log(1 - D_\phi(s, G_\xi(s)))$. We use the model-based variant of GAIL (MGAIL) (Baram et al., 2017) which renders this reward fully differentiable with respect to ξ over a simulation rollout, allowing efficient model training. GAIL has been validated by Kuefler et al. (2017) to realistically mimic human-like driving behavior from the NGSim dataset across multiple metrics. These include the similarity of low-level actions (speeds, accelerations, turn-rates, jerks, and time-to-collision), as well as higher-level behaviors (lane change rate, collision rate, hard-brake rate, etc).

Our importance sampling and cross-entropy methods use not just a single instance of model parameters ξ, but rather a distribution over them to form a generative model of human driving behavior. To model this distribution, we use a (multivariate normal) parametric bootstrap over a trained ensemble of generators $\xi^i,\ i = 1, \ldots, m$. Our models ξ^i are high-dimensional ($\xi \in \mathbb{R}^d,\ d > m$) as they characterize the weights of large neural networks, so we employ the graphical lasso (Friedman et al., 2008) to fit the inverse covariance matrix for our ensemble. This approach to modeling uncertainty in neural-network weights is similar to the bootstrap approach of Osband et al. (2016). Other approaches include using dropout for inference (Gal and Ghahramani, 2016) and variational methods (Graves, 2011; Blundell et al., 2015; Kingma et al., 2015).

While several open source driving simulators have been proposed (Dosovitskiy et al., 2017a; Shah et al., 2017; Quiter and Ernst, 2018), our problem formulation requires unique features to allow sampling from a continuous distribution of driving policies for environmental agents. Conditional on each sample of model parameters ξ, the simulator constructs a (random) rollout of vehicle behaviors according to G_ξ. Unlike other existing simulators, ours is designed to efficiently execute and update these policies as new samples ξ are drawn for each rollout.

3.4.2. System architecture

The second key characteristic of our framework is that it enables black-box testing the AV as a *whole system*. Flaws in complex systems routinely occur at poorly specified interfaces between components, as interactions between processes can induce unexpected behavior. Consequently, solely testing subcomponents of an AV control pipeline separately is insufficient (Abbas et al., 2019). Moreover, it is increasingly common for manufacturers to utilize software and hardware artifacts for which they do not have any whitebox model (Heinecke et al., 2004; Cheah et al., 2016). We provide a concise but extensible language-agnostic interface to our benchmark world model so that common AV sensors such as cameras and LIDAR can provide the necessary inputs to induce vehicle actuation commands.

Our simulator is a distributed, modular framework, which is necessary to support the inclusion of new AV systems and updates to the environment-vehicle policies. A benefit of this design is that simulation rollouts are simple to parallelize. In particular, we allow instantiation of multiple simulations simultaneously, without requiring that each include the entire set of components. For example, a desktop may support only one instance of Unreal Engine but could be capable of simulating 10 physics simulations in parallel; it would be impossible to fully utilize the compute resource with a monolithic executable wrapping all engines together. Our architecture enables instances of the components to be distributed on heterogeneous GPU compute clusters while maintaining the ability to perform meaningful analysis locally on commodity desktops. In Appendix A.1, we detail our scenario specification, which describes how Algorithm 1 maps onto our distributed architecture.

3.5. Experiments

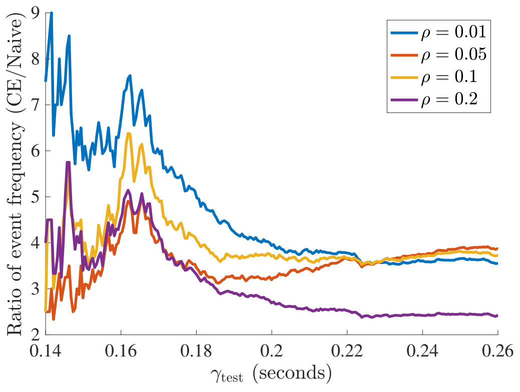
In this section, we demonstrate our risk-based framework on a multi-agent highway scenario. As the rare-event probability of interest p_γ gets smaller, the cross-entropy method learns to sample more rare events compared to naive Monte Carlo sampling; we empirically observe that the cross-entropy method produces 2-20 times as many rare events as its naive counterpart. Our findings hold across different ego-vehicle policies, base distributions P_0, and scenarios.

To highlight the modularity of our simulator, we evaluate the rare-event probability p_γ on two different ego-vehicle policies. The first is an instantiation of an imitation learning (non-vision) policy which uses LIDAR as its primary perceptual input. Secondly, we investigate a vision-based controller (vision policy), where the ego-vehicle drives with an end-to-end highway autopilot network (Bojarski et al., 2016), taking as input a rendered image from the simulator (and LIDAR observations) and outputting actuation commands. See Appendix A.2 for a summary of network architectures used.

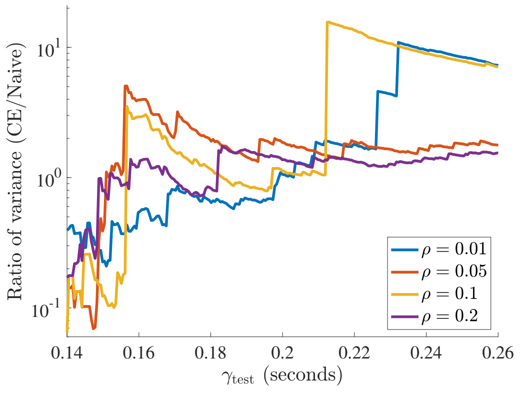
We consider a scenario consisting of six agents, five of which are considered part of the environment. The environment vehicles' policies follow the distribution learned in Section 3.4.1. All vehicles are constrained to start within a set of possible initial configurations consisting of pose and velocity, and each vehicle has a goal of reaching the end of the approximately 2 km stretch of road. Fig. 9 shows one such configuration of the scenario, along with rendered images from the simulator. We create scene geometry based on surveyors' records and photogrammetric reconstructions of satellite imagery of the portion of I-80 in Emeryville, California where the traffic data was collected (US Department of Transportation – FHWA, 2008).

Simulation parameters We detail our postulated base distribution P_0. Letting m denote the number of vehicles, we consider the random tuple $X = (S, T, W, V, \xi)$ as our simulation parameter where the pair $(S, T) \in \mathbb{R}_+^{m\times 2}$ indicates the two-dimensional positioning of each vehicle in their respective lanes (in meters), W the orientation of each vehicle (in degrees), and V the initial velocity of each vehicle (in meters per second). We use $\xi \in \mathbb{R}^{404}$ to denote the weights of the last layer of the neural network trained to imitate human-like driving behavior. Specifically, we set $S \sim 40\text{Beta}(2, 2) + 80$ with respect to the starting point of the road, $T \sim 0.5\text{Beta}(2, 2) - 0.25$ with respect to the lane's center, $W \sim 7.2\text{Beta}(2, 2) - 3.6$ with respect to facing forward, and $V \sim 10\text{Beta}(2, 2) + 10$. We assume $\xi \sim \mathcal{N}(\mu_0, \Sigma_0)$, with the mean and covariance matrices learned via the ensemble approach outlined in Section 3.4.1. The neural network whose last layer is parameterized by ξ describes the policy of environment vehicles; it takes as input the state of the vehicle and LIDAR observations of the surrounding environment (see Appendix A.2 for more details). Throughout this section, we define our measure of safety $f : \mathcal{X} \to \mathbb{R}$ as the minimum time-to-collision (TTC) over the simulation rollout. We calculate TTC from the center of mass of the ego vehicle; if the ego-vehicle's body crashes into obstacles, we end the simulation before the TTC can further decrease (see Appendix A.1 for details).

Cross-entropy method Throughout our experiments, we impose constraints on the space of importance samplers (adversarial distributions) for feasibility. Numerical stability considerations predominantly drive our hyperparameter choices. For model parameters ξ, we also constrain the search space to ensure that generative models G_ξ maintain reasonably realistic human-like policies (recall Sec. 3.4.1). For S, T, W, and V, we let $\{\text{Beta}(\alpha, \beta) : \alpha, \beta \in [1.5, 7]\}$ be the model space over which the cross-entropy method searches, scaled and centered appropriately to match the scale of the respective base distributions. We restrict the search space of distributions over $\xi \in \mathbb{R}^{404}$ by searching over $\{\mathsf{N}(\mu, \Sigma_0) : \|\mu - \mu_0\|_\infty \leq .01\}$, where (μ_0, Σ_0) are the parameters of the base (bootstrap)



(a) Ratio of number of rare events vs. threshold



(b) Ratio of variance vs. threshold

Figure 10: The ratio of (a) number of rare events and (b) variance of estimator for p_γ between cross-entropy method and naive MC sampling for the non-vision ego policy. Rarity is inversely proportional to γ, and, as expected, we see the best performance for our method over naive MC at small γ.

Search Algorithm	$\gamma_{\text{test}} = 0.14$	$\gamma_{\text{test}} = 0.15$	$\gamma_{\text{test}} = 0.19$	$\gamma_{\text{test}} = 0.20$
Naive 1300K	(12.4±3.1)e-6	(80.6±7.91)e-6	(133±3.2)e-5	(186±3.79)e-5
Cross-entropy 100K	(19.8±8.88)e-6	(66.1 ± 15)e-6	(108± 9.51)e-5	(164 ± 14)e-5
Naive 100K	(20±14.1)e-6	(100± 31.6)e-6	(132±11.5)e-5	(185±13.6)e-5

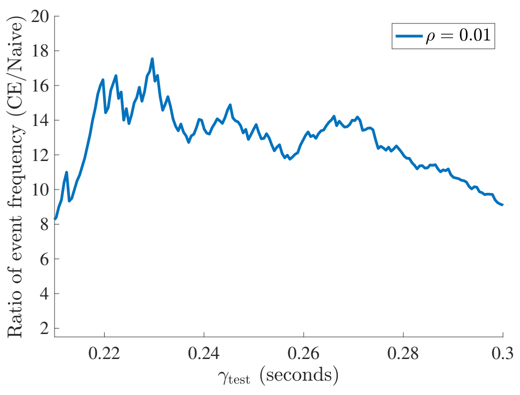
Table 3: Estimate of rare-event probability p_γ (non-vision ego policy) with standard errors. For the cross-entropy method, we show results for the learned importance sampling distribution with $\rho = 0.01$.

distribution. For our importance sampling distribution P_θ, we use products of the above marginal distributions. These restrictions on the search space mitigate numerical instabilities in computing likelihood ratios within our optimization routines, which is important for our high-dimensional problems.

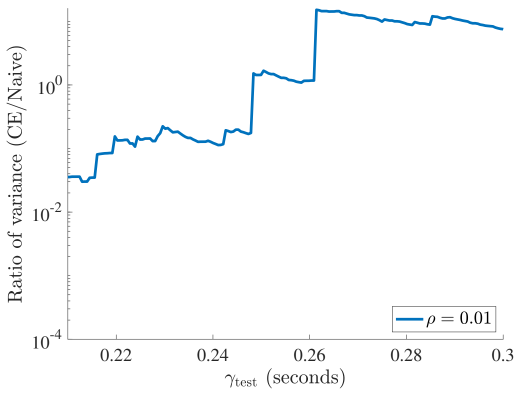
We first illustrate the dependence of the cross-entropy method on its hyperparameters. We choose to use a non-vision ego-vehicle policy as a test bed for hyperparameter tuning, since this allows us to take advantage of the fastest simulation speeds for our experiments. We focus on the effects (in Algorithm 1) of varying the most influential hyperparameter, $\rho \in (0, 1]$, which is the quantile level determining the rarity of the observations used to compute the importance sampler θ_k. Intuitively, as ρ approaches 0, the cross-entropy method learns importance samplers P_θ that up-weight unsafe regions of $\mathcal{X}$ with lower $f(x)$, increasing the frequency of sampling rare events (events with $f(X) \leq \gamma$). In order to avoid overfitting θ_k as $\rho \to 0$, we need to increase N_k as ρ decreases. Our choice of N_k is borne out of computational constraints as it is the biggest factor that determines the run-time of the cross-entropy method. Consistent with prior works (Rubinstein and Kroese, 2004; Hu and Hu, 2009), we observe empirically that $\rho \in [0.01, 0.2]$ is a good range for the values of N_k deemed feasible for our computational budget ($N_k = 1000 \sim 5000$). We fix the number of iterations at $K = 100$, number of samples taken per iteration at $N_k = 5000$, step size for updates at $\alpha_k = 0.8$, and $\gamma = 0.14$. As we see below, we consistently observe that the cross-entropy method learns to sample significantly more rare events, despite the high-dimensional nature ($d \approx 500$) of the problem.

To evaluate the learned parameters, we draw $n = 10^5$ samples from the importance sampling distribution to form an estimate of p_γ. In Figure 10, we vary ρ and report the relative performance of the cross-entropy method compared to naive Monte Carlo sampling. Even though we set $\gamma = 0.14$ in Algorithm 1, we evaluate the performance of all models with respect to multiple threshold levels γ_{test}. We note that as ρ approaches 0, the cross-entropy method learns to frequently sample increasingly rare events; the cross-entropy method yields 3-10 times as many dangerous scenarios, and achieves 2-16 times variance reduction depending on the threshold level γ_{test}. In Table 3, we contrast the estimates provided by naive Monte Carlo and the importance sampling estimator provided by the cross-entropy method with $\rho = 0.01$; to form a baseline estimate, we run naive Monte Carlo with $1.3 \cdot 10^6$ samples. For a given number of samples, the cross-entropy method with $\rho = 0.01$ provides more precise estimates for the rare-event probability $p_\gamma \approx 10^{-5}$ over naive Monte Carlo.

We now leverage the tuned hyperparameter ($\rho = 0.01$) for our main experiment: evaluating the probability of a dangerous event for the vision-based ego policy. We find that the hyperparameters for the cross-entropy method generalize, allowing us to produce good importance samplers for a very different policy without further tuning. Based on our computational budget (with our current implementation, vision-based simulations run about 15 times slower



(a) Ratio of number of rare events vs. threshold



(b) Ratio of variance vs. threshold

Figure 11: The ratio of (a) number of rare events and (b) variance of estimator for p_γ between cross-entropy method and naive MC sampling for the vision-based ego policy.

Search Algorithm	$\gamma_{\text{test}} = 0.22$	$\gamma_{\text{test}} = 0.23$	$\gamma_{\text{test}} = 0.24$	$\gamma_{\text{test}} = 0.25$
Cross-entropy 50K	(5.87±1.82)e-5	(13.0± 2.94)e-5	(19.0 ± 3.14)e-5	(4.52 ± 1.35)e-4
Naive 50K	(11.3±4.60)e-5	(20.6±6.22)e-5	(43.2±9.00)e-5	(6.75±1.13)e-4

Table 4: Estimate of rare-event probability p_γ (non-vision ego policy) with standard errors. For the cross-entropy method, we show results for the learned importance sampling distribution with $\rho = 0.01$.

than simulations with only non-vision policies), we choose $K = 20$ and $N_k = 1000$ for the cross-entropy method to learn a good importance sampling distribution for the vision-based policy (although we also observe similar behavior for N_k as small as 100). In Figure 11, we illustrate again that the cross-entropy method learns to sample dangerous scenarios more frequently (Figure 11a)—up to 18 times that of naive Monte Carlo—and produces importance sampling estimators with lower variance (Figure 11b). As a result, our estimator in Table 4 is better calibrated compared to that computed from naive Monte Carlo.

Qualitative analysis We provide a qualitative interpretation for the learned parameters of the importance sampler. For initial velocities, angles, and positioning of vehicles, the importance sampler shifts environmental vehicles to box in the ego-vehicle and increases the speeds of trailing vehicles by 20%, making accidents more frequent. We also observe that the learned distribution for initial conditions have variance 50% smaller than that of the base distribution, implying concentration around adversarial conditions. Perturbing the policy weights ξ for GAIL increases the frequency of risky high-level behaviors (lane-change rate, hard-brake rate, etc.). An interesting consequence of using our definition of TTC from the center of the ego vehicle (cf. Appendix A.1) as a measure of safety is that dangerous events $f(X) \leq \gamma_{\text{test}}$ (for small γ_{test}) include frequent sideswiping behavior, as such accidents result in smaller TTC values than front- or rear-end collisions. See the Appendix for a reference to supplementary videos that exhibit the range of behavior across many levels γ_{test}. The modularity of our simulation framework easily allows us to modify the safety objective to an alternative definition of TTC or even include more sophisticated notions of safety, *e.g.* temporal-logic specifications or implementations of responsibility-sensitive safety (RSS) (Shalev-Shwartz et al., 2017; Roohi et al., 2018).

3.6. Related work and conclusions

Given the complexity of AV software and hardware components, it is unlikely that any single method will serve as an oracle for certification. Many existing tools are complementary to our risk-based framework. In this section, we compare and contrast representative results in testing, verification, and simulation.

AV testing generally consists of three paradigms. The first, largely attributable to regulatory efforts, uses a finite set of basic competencies (*e.g.* the Euro NCAP Test Protocol (Schram et al., 2013)); while this methodology is successful in designing safety features such as airbags and seat-belts, the non-adaptive nature of static testing is less effective in complex software systems found in AVs. Alternatively, real-world testing—deployment of vehicles with human oversight—exposes the vehicle to a wider variety of unpredictable test conditions. However, as we outlined above, these methods pose a danger to the public and require prohibitive number of driving hours due to the rare nature of accidents (Kalra and Paddock, 2016). Simulation-based falsification (in our context, simply finding any crash) has also been successfully utilized (Tuncali et al., 2016); this approach does not maintain a link to the likelihood of the occurrence of a particular event, which we believe to be key in acting to prioritize and correct AV behavior.

Formal verification methods (Kwiatkowska et al., 2011; Althoff and Dolan, 2014; Seshia et al., 2015; O'Kelly et al., 2016) have emerged as a candidate to reduce the intractability of empirical validation. A verification procedure considers whether the system can *ever* violate a specification and returns either a proof that there is no such execution or a counterexample. Verification procedures require a white-box description of the system (although it may be abstract), as well as a mathematically precise specification. Due to the impossibility of certifying safety in *all* scenarios, these approaches (Shalev-Shwartz et al., 2017) require further specifications that assign blame in the case of a crash. Such assignment of blame is impossible to completely characterize and relies on subjective notions of fault. Our risk-based framework allows one to circumvent this difficulty by only using a measure of safety that does not assign blame (e.g. TTC) and replacing the specifications that assign blame with a probabilistic notion of how likely the accident is. While this approach requires a learned model of the world P_0—a highly nontrivial statistical task in itself—the adaptive importance sampling techniques we employ can still efficiently identify dangerous scenarios even when P_0 is not completely accurate. Conceptually, we view verification and our framework as complementary; they form powerful tools that can evaluate safety *before* deploying a fleet for real-world testing.

Even given a consistent and complete notion of blame, verification remains highly intractable from a computational standpoint. Efficient algorithms only exist for restricted classes of systems in the domain of AVs, and they are fundamentally difficult to scale. Specifically, AVs—unlike previous successful applications of verification methods to application domains such as microprocessors (Baier and Katoen, 2008)—include both continuous and discrete dynamics. This class of dynamics falls within the purview of hybrid systems (Lygeros, 2004), for which exhaustive verification is largely undecidable (Henzinger et al., 1995).

Verifying individual components of the perception pipeline, even as standalone systems, is a nascent, active area of research (see (Arora et al., 2014; Cohen et al., 2016; Bartlett et al., 2017) and many others). Current subsystem verification techniques for deep neural networks (Huang et al., 2017; Katz et al., 2017; Tjeng and Tedrake, 2017) do not scale to state-of-the-art models and largely investigate the robustness of the network with respect to small perturbations of a single sample. There are two key assumptions in these works; the label of the input is unchanged within the radius of allowable perturbations, and the resulting expansion of the test set covers a meaningful portion of possible inputs to the network. Unfortunately, for realistic cases in AVs it is likely that perturbations to the state of the world which in turn generates an image *should* change the label. Furthermore, the combinatorial nature of scenario configurations casts serious doubt on any claims of coverage.

In our risk-based framework, we replace the complex system specifications required for formal verification methods with a model P_0 that we learn via imitation-learning techniques. Generative adversarial imitation learning (GAIL) was first introduced by Ho and Ermon (2016) as a way to directly learn policies from data and has since been applied to model human driving behavior by Kuefler et al. (2017). Model-based GAIL (MGAIL) is the specific variant of GAIL that we employ; introduced by Baram et al. (2017), MGAIL's generative model is fully differentiable, allowing efficient model training with standard stochastic approximation methods.

The cross-entropy method was introduced by Rubinstein (2001) and has attracted interest in many rare-event simulation scenarios (Rubinstein and Kroese, 2004; Kroese et al., 2013). More broadly, it can be thought of as a model-based optimization method (Hu and Hu, 2009, 2011; Hu et al., 2012; Zabinsky, 2013; Hu et al., 2014; Zhou and Hu, 2014). With respect to assessing safety of AVs, the cross-entropy method has recently been applied in simple lane-changing and car-following scenarios in two dimensions (Zhao, 2016; Zhao et al., 2018). Our work significantly extends these works by implementing a photo-realistic simulator that can assess the deep-learning based perception pipeline along with the control framework. We address the development of improved rare-event simulation methods in Chapter 4.

To summarize, a fundamental tradeoff emerges when comparing the requirements of our risk-based framework to other AV assessment techniques, such as real-world testing or formal verification. Real-world testing endangers the public but is still in some sense a gold standard. Verified subsystems provide evidence that the AV should drive safely even if the estimated distribution shifts, but verification techniques are limited by computational intractability as well as the need for both white-box models and the completeness of specifications that assign blame (*e.g.* (Shalev-Shwartz et al., 2017)). In turn, our risk-based framework is most useful when the base distribution P_0 is accurate, but even when P_0 is misspecified, our adaptive importance sampling techniques can still efficiently identify dangerous scenarios, especially those that may be missed by verification methods assigning blame. Our framework offers significant speedups over real-world testing and allows efficient evaluation of black-box AV input/output behavior, providing a powerful tool to aid in the design of safe AVs.

CHAPTER 4 : Gradient-guided Bridge Sampling

4.1. Chapter Overview

Aspects of this chapter have been adapted from "Neural Bridge Sampling for Evaluating Safety-Critical Autonomous Systems" which appeared in the Proceedings of Neural Information Processing Systems 2020 as joint work with Aman Sinha, John Duchi, and Russ Tedrake

4.2. Introduction

Data-driven and learning-based approaches have the potential to enable robots and autonomous systems that intelligently interact with unstructured environments. Unfortunately, evaluating the performance of the closed-loop system is challenging, limiting the success of such methods in safety-critical settings. Even if we produce a deep reinforcement learning agent better than a human at driving, flying a plane, or performing surgery, we have no tractable way to certify the system's quality. Thus, currently deployed safety-critical autonomous systems are limited to structured environments that allow mechanisms such as PID control, simple verifiable protocols, or convex optimization to enable guarantees for properties like stability, consensus, or recursive feasibility (see *e.g.* (Doyle et al., 2013; Borrelli et al., 2017)). The stylized settings of these problems and the limited expressivity of guaranteeable properties are barriers to solving unstructured, real-world tasks such as autonomous navigation, locomotion, and manipulation.

The goal of this paper is to *efficiently* evaluate complex systems that lack safety guarantees and/or operate in unstructured environments. We assume access to a simulator to test the system's performance. Given a distribution $X \sim P_0$ of simulation parameters that describe typical environments for the system under test, our governing problem is to estimate the probability of an adverse event

$$p_\gamma := \mathbb{P}_0(f(X) \leq \gamma). \tag{4.1}$$

The parameter γ is a threshold defining an adverse event, and $f : \mathcal{X} \to \mathbb{R}$ measures the safety of a realization x of the agent and environment (higher values are safer). In this work, we assume P_0 is known; the system-identification and generative-modeling literatures (*e.g.* (Åström and Eykhoff, 1971; Papamakarios et al., 2019)) provide several approaches to learn or specify P_0. A major challenge for solving problem (4.1) is that the better an agent is at performing a task (*i.e.* the smaller p_γ is), the harder it is to confidently estimate p_γ—one rarely observes events with $f(x) \leq \gamma$. For example, when P_0 is light-tailed, the sample complexity of estimating p_γ using naive Monte Carlo samples grows exponentially (Bucklew, 2013).

Problem (4.1) is often solved in practice by naive Monte Carlo estimation methods, the simplest of which *explore* the search space via random samples from P_0. These methods are unbiased and easy to parallelize, but they exhibit poor sample complexity. Naive Monte Carlo can be improved by adding an adaptive component *exploiting* the most informative portions of random samples drawn from a sequence of approximating distributions $P_0, P_1, \ldots, P_K$. However, standard adaptive Monte Carlo methods (*e.g.* (Cérou and Guyader, 2007)), though they may use first-order information on the distributions P_k themselves, fail to use first-order information about f to improve sampling; we explicitly leverage this to accelerate convergence of the estimate through *optimization*.

Naive applications of first-order optimization methods in the estimation problem (4.1)—for example biasing a sample in the direction $-\nabla f(x)$ to decrease $f(x)$—also require second-order information to correct for the distortion of measure that such transformations induce. Consider the change of variables formula for distributions $\rho(y) = \rho(g^{-1}(y)) \cdot |\det J_{g^{-1}}(y)|$ where $y = g(x)$. When $g(x)$ is a function of the gradient $\nabla f(x)$, the volume distortion $|\det J_{g^{-1}}(y)|$ is a function of the Hessian $\nabla^2 f(x)$. Hessian computation, if even defined, is unacceptably expensive for high-dimensional spaces $\mathcal{X}$ and/or simulations that involve the time-evolution of a dynamical system; our approach avoids any Hessian computation. In contrast, gradients $\nabla f(x)$ can be efficiently computed for many closed-loop systems (Abbas et al., 2014; Pant et al., 2017; Yaghoubi and Fainekos, 2018; Leung et al.) or through the use of surrogate methods (Williams, 1992; Deisenroth and Rasmussen, 2011; Duchi et al., 2015; Baram et al., 2017).

To that end, we propose *gradient-guided bridge sampling*, a technique that combines *exploration, exploitation*, and *optimization* to efficiently solve the estimation problem (4.1). Specifically, we consider a novel Markov-chain Monte Carlo (MCMC) scheme that moves along an adaptive ladder of intermediate distributions P_k (with corresponding unnormalized densities $\rho_k(x)$ and normalizing constants $Z_k := \int_{\mathcal{X}} \rho_k(x)dx$). This MCMC scheme iteratively transforms the base distribution P_0 to the distribution of interest $P_0 I\{f(x) \leq \gamma\}$. Gradient-guided bridge sampling adaptively balances exploration in the search space (via $\nabla \log \rho_0$) against optimization (via ∇f), while avoiding Hessian computations. Our final estimate $\hat{p}_\gamma$ is a function of the ratios Z_k/Z_{k-1} of the intermediate distributions P_k, the so-called "bridges" (Bennett, 1976; Meng and Wong, 1996).

Contributions and outline Section 4.3 presents our method, while Section 4.4 provides guarantees for its statistical performance and overall efficiency. A major focus of this work is empirical, and accordingly, Section 4.5 empirically demonstrates the superiority of gradient-guided bridge sampling over competing techniques in autonomous vehicle performance evaluation applications: (i) we perform model comparisons for two learning-based approaches to autonomous navigation (ii) we efficiently evaluate the failure rate of a real AV system which has driven over 20 million miles on public roads.

4.2.1. Related Work

Safety evaluation Several communities (Corso et al., 2020) have attempted to evaluate the closed-loop performance of cyber-physical, robotic, and embodied agents both with and without learning-based components. Existing solutions are predicated on the definition of the evaluation problem: verification, falsification, or estimation. In this paper we consider a method that utilizes interactions with a gradient oracle in order to solve the estimation problem (4.1). In contrast to our approach, the verification community has developed tools (*e.g.* (Kong et al., 2015a; Chen et al., 2013; Althoff, 2015)) to investigate whether any adverse or unsafe executions of the system exist. Such methods can certify that failures are impossible, but they require that the model is written in a formal language (a barrier for realistic systems), and they require whitebox access to this formal model. Falsification approaches (*e.g.* Esposito et al., 2004; Donzé, 2010; Annpureddy et al., 2011; Zutshi et al., 2014; Dreossi et al., 2019; Qin et al., 2019) attempt to find *any* failure cases for the system (but not the overall probability of failure). Similar to our approach, some falsification approaches (*e.g.* Abbas et al., 2014; Yaghoubi and Fainekos, 2018) utilize gradient information, but their goal is to simply minimize $f(x)$ rather than solve problem (4.1). Adversarial machine learning is closely related to falsification; the key difference is the domain over which the search for falsifying evidence is conducted. Adversarial examples (*e.g.* (Madry et al., 2017; Katz et al., 2017; Sinha et al., 2017; Tjeng and Tedrake, 2017)) are typically restricted to a p-norm ball around a point from a dataset, whereas falsification considers all possible in-distribution examples. Both verification and falsification methods provide less information about the system under test than estimation-based methods: they return only whether or not the system satisfies a specification. When the system operates in an unstructured environment (*e.g.* driving in an urban setting), the mere existence of failures is trivial to demonstrate (Shalev-Shwartz et al., 2017). Several authors (*e.g.* O'Kelly et al., 2018; Webb et al., 2018) have proposed that it is more important in such settings to understand the overall frequency of failures as well as the relative likelihoods of different failure modes, motivating our approach.

Rare-event simulation methods When sampling rare events and estimating their probability, there are two main branches of related work: parametric adaptive importance sampling (AIS) (Marshall, 1954; Oh and Berger, 1992) and nonparametric sequential Monte Carlo (SMC) techniques (Doucet et al., 2001; Del Moral et al., 2006). Both of these literatures are advanced forms of variance reduction techniques, and they are complementary to standard methods such as control variates (Rubinstein and Marcus, 1985; Hesterberg and Nelson, 1998). Parametric AIS techniques, such as the cross-entropy method (Rubinstein and Kroese, 2004), postulate a family of distributions for the optimal importance-sampling distribution. They iteratively perform heuristic optimization procedures to update the sampling distribution. SMC techniques perform sampling from a sequence of probability distributions defined nonparametrically by the samples themselves. The SMC formalism encompasses particle filters, birth-death processes, and smoothing filters (Del Moral, 2004).

Our method employs bridge sampling (Bennett, 1976; Meng and Wong, 1996), which is closely related to other SMC techniques such as umbrella sampling (Chen et al., 2012), multilevel splitting (Bréhier et al., 2015; Cérou and Guyader, 2007), and path sampling (Gelman and Meng, 1998). The operational difference between these methods is in the form of the intermediate distribution used to calculate the ratio of normalizing constants. Namely, the optimal umbrella sampling distribution is more brittle than that of bridge sampling (Chen et al., 2012). Multilevel splitting employs hard barriers through indicator functions, whereas our approach relaxes these hard barriers with smoother exponential barriers. Finally, Path sampling generalizes bridge sampling by taking discrete bridges to a continuous limit; however, this approach is difficult to implement in an adaptive fashion.

4.3. Proposed approach

As we note in Section 3.2, naive Monte Carlo measures probabilities of rare events inefficiently. Instead, we consider a sequential Monte Carlo (SMC) approach: we decompose the rare-event probability p_γ into a chain of intermediate quantities, each of which is tractable to compute with standard Monte Carlo methods. Specifically, consider K distributions P_k with corresponding (unnormalized) probability densities ρ_k and normalizing constants $Z_k := \int_{\mathcal{X}} \rho_k(x)dx$. Let ρ_0 correspond to the density for P_0 and $\rho_\infty(x) := \rho_0(x)I\{f(x) \leq \gamma\}$ be the (unnormalized) conditional density for the region of interest. Then, we consider the following decomposition:

$$p_\gamma := \mathbb{P}_0(f(X) \leq \gamma) = \mathbb{E}_{P_K}\left[\frac{Z_K}{Z_0}\frac{\rho_\infty(X)}{\rho_K(X)}\right], \qquad \frac{Z_K}{Z_0} = \prod_{k=1}^{K}\frac{Z_k}{Z_{k-1}}. \tag{4.2}$$

Algorithm 2 Gradient-guided bridge sampling

```
**Input:** $N$ samples $x_i^0 \overset{\text{i.i.d.}}{\sim} P_0$, MCMC steps $T$, step size $\alpha \in (0,1)$, stop condition $s \in (0,1)$
Initialize $k \leftarrow 0$, $\beta_0 \leftarrow 0$, $\log(\hat{p}_\gamma) \leftarrow 0$
**while** $\frac{1}{N}\sum_i I\{f(x_i^k) \le \gamma\} < s$ **do**
    $\beta_{k+1} \leftarrow$ solve problem (4.6)
    **for** $i = 1$ to $N$, in parallel
        $x_i^{k+1} \overset{\text{i.i.d.}}{\sim} \text{Mult}(\{\rho_{k+1}(x_i^k)/\rho_k(x_i^k)\})$  // multinomial resampling
    **for** $t = 1$ to $T$
        **for** $i = 1$ to $N$, in parallel
            $x_i^{k+1} \leftarrow \text{SplitHMC}(x_i^k, \theta_k)$  // Appendix B.1
    $\log(\hat{p}_\gamma) \leftarrow \log(\hat{p}_\gamma) + \log(Z_{k+1}/Z_k)$  //bridge estimate (4.4)
    $k \leftarrow k+1$
$\log(\hat{p}_\gamma) \leftarrow \log(\hat{p}_\gamma) + \log(\frac{1}{N}\sum_i I\{f(x_i^k) \le \gamma\})$
```

Although we are free to choose the intermediate distributions arbitrarily, we will show below that our estimate for each ratio Z_k/Z_{k-1} and thus p_γ is accurate insofar as the distributions sufficiently overlap (a concept we make rigorous in Section 4.4). Thus, the intermediate distributions act as bridges that iteratively steer samples from P_0 towards P_K. One special case is the multilevel splitting approach (Kahn and Harris, 1951; Bréhier et al., 2015; Webb et al., 2018; Norden et al., 2019), where $\rho_k(x) := \rho_0(x)I\{f(x) \le L_k\}$ for levels $\infty =: L_0 > L_1 \ldots > L_K := \gamma$. In this paper, we introduce an exponential tilting barrier (Siegmund, 1976)

$$\rho_k(x) := \rho_0(x) \exp\left(\beta_k \left[\gamma - f(x)\right]_-\right), \tag{4.3}$$

which allows us to take advantage of gradients $\nabla f(x)$. Here we use the "negative ReLU" function defined as $[x]_- := -[-x]_+ = xI\{x < 0\}$, and we assume that the measure of non-differentiable points, *e.g.* where $\nabla f(x)$ does not exist or $f(x) = \gamma$, is zero (see Appendix B.1 for a detailed discussion of this assumption). We set $\beta_0 := 0$ and adaptively choose $\beta_k > \beta_{k-1}$. The parameter β_k tilts the distribution towards the distribution of interest: $\rho_k \to \rho_\infty$ as $\beta_k \to \infty$. In what follows, we describe an MCMC method that combines exploration, exploitation, and optimization to draw samples $X_i^k \sim P_k$. We then show how to compute the ratios Z_k/Z_{k-1} given samples from both P_{k-1} and P_k. Finally, we describe an adaptive way to choose the intermediate distributions P_k. Algorithm 2 summarizes the overall approach.

MCMC with an exponential barrier Gradient-based MCMC techniques such as the Metropolis-adjusted Langevin algorithm (MALA) (Rossky et al., 1978; Roberts and Stramer, 2002) or Hamiltonian Monte Carlo (HMC) (Duane et al., 1987; Neal, 2012) use gradients $\nabla \log \rho_0(x)$ to efficiently explore the space $\mathcal{X}$ and avoid inefficient random-walk behavior (Durmus et al., 2017; Chen et al., 2019). Classical mechanics inspires the HMC approach: HMC introduces an auxiliary random momentum variable $v \in \mathcal{V}$ and generates proposals by performing Hamiltonian dynamics in the augmented state-space $\mathcal{X} \times \mathcal{V}$. These dynamics conserve volume in the augmented state-space, even when performed with discrete time steps (Leimkuhler and Reich, 2004).

By including the barrier $\exp\left(\beta_k \left[\gamma - f(x)\right]_-\right)$, we combine exploration with optimization; the magnitude of β_k in the barrier modulates the importance of ∇f (optimization) over $\nabla \log \rho_0$ (exploration), two elements of the HMC proposal (see Appendix B.1 for details). We discuss the adaptive choice for β_k below. Most importantly, we avoid any need for Hessian computation because the dynamics conserve volume. As Algorithm 2 shows, we perform MCMC as follows: given N samples $x_i^{k-1} \sim P_{k-1}$ and a threshold β_k, we first resample using their importance weights (exploiting the performance of samples that have lower function value than others) and then perform T HMC steps. In this paper, we implement split HMC (Shahbaba et al., 2014) which is convenient for dealing with the decomposition of $\log \rho_k(x)$ into $\log \rho_0(x) + \beta_k[\gamma - f(x)]_-$ (see Appendix B.1 for details).

Estimating Z_k/Z_{k-1} via bridge sampling Bridge sampling (Bennett, 1976; Meng and Wong, 1996) allows estimating the ratio of normalizing constants of two distributions by rewriting

$$E_k := \frac{Z_k}{Z_{k-1}} = \frac{Z_k^B/Z_{k-1}}{Z_k^B/Z_k} = \frac{\mathbb{E}_{P_{k-1}}[\rho_k^B(X)/\rho_{k-1}(X)]}{\mathbb{E}_{P_k}[\rho_k^B(X)/\rho_k(X)]}, \qquad \widehat{E}_k = \frac{\sum_{i=1}^N \rho_k^B(x_i^{k-1})/\rho_{k-1}(x_i^{k-1})}{\sum_{i=1}^N \rho_k^B(x_i^k)/\rho_k(x_i^k)}, \quad (4.4)$$

where ρ_k^B is the density for a bridge distribution between P_{k-1} and P_k, and Z_k^B is its associated normalizing constant. We employ the geometric bridge $\rho_k^B(x) := \sqrt{\rho_{k-1}(x)\rho_k(x)}$. In addition to being simple to compute, bridge sampling with a geometric bridge enjoys the asymptotic performance guarantee that the relative mean-square error scales inversely with the Bhattacharyya coefficient, $G(P_{k-1}, P_k) = \int_{\mathcal{X}} \sqrt{\frac{\rho_{k-1}(x)}{Z_{k-1}} \frac{\rho_k(x)}{Z_k}} dx \in [0,1]$ (see Appendix B.2 for a proof). This value is closely related to the Hellinger distance, $H(P_{k-1}, P_k) = \sqrt{2 - 2G(P_{k-1}, P_k)}$. In Section 4.4, we analyze the ramifications of this fact on the overall convergence of our method.

Adaptive intermediate distributions Because we assume no prior knowledge of the system under test, we exploit previous progress to choose the intermediate β_k online; this is a key difference to our approach compared to other forms of sequential Monte Carlo (*e.g.* (Neal, 2001, 2005)) which require a predetermined schedule for β_k. We define the quantities

$$a_k := \textstyle\sum_i^N I\{f(x_i^k) \le \gamma\}/N, \;\; b_k(\beta) := \sum_{i=1}^N \exp\left((\beta - \beta_k)[\gamma - f(x_i^k)]_-\right)/N. \tag{4.5}$$

The first is the fraction of samples that have achieved the threshold. The second is an importance-sampling estimate of E_{k+1} given samples $x_i^k \sim P_k$, written as a function of β. For fixed fractions $\alpha, s \in (0,1)$ with $\alpha < s$, β_{k+1} solves the following optimization problem:

$$\text{maximize } \beta \;\text{ s.t. } \{b_k(\beta) \ge \alpha, \;\; a_k/b_k(\beta) \le s\}. \tag{4.6}$$

Since $b_k(\beta)$ is monotonically decreasing and $b_k(\beta) \ge a_k$, this problem can be solved efficiently via binary search. The constant α tunes how quickly we enter the tails of P_0 (smaller α means fewer iterations), whereas s is a stop condition for the last iteration. Choosing β_{k+1} via (4.6) yields a crude estimate for the ratio Z_{k+1}/Z_k as α (or a_{K-1}/s for the last iteration). The bridge-sampling estimate $\widehat{E}_{k+1}$ corrects this crude estimate once we have samples from the next distribution P_{k+1}.

4.4. Performance analysis

We can write the empirical estimator of the function (4.2) as

$$\hat{p}_\gamma = \prod_{k=1}^{K} \widehat{E}_k \frac{1}{N} \sum_{i=1}^{N} \frac{\rho_\infty(x_i^K)}{\rho_K(x_i^K)}, \tag{4.7}$$

where $\widehat{E}_k$ is given by the expression (4.4). We provide guarantees for both the time complexity of running Algorithm 2 (*i.e.* the iterations K) as well as the overall mean-square error of $\hat{p}_\gamma$. For simplicity, we provide results for the asymptotic (large N) and well-mixed MCMC (large T) limits. Assuming these conditions, we have the following:

Proposition 1. *Let $K_0 := \lfloor \log(p_\gamma)/\log(\alpha) \rfloor$. Then, for large N and T, $s \geq 1/3$, and $p_\gamma < s$, the total number of iterations in Algorithm 2 approaches $K \overset{\text{a.s.}}{\to} K_0 + I\{p_\gamma/\alpha^{K_0} < s\}$. Furthermore, the asymptotic relative mean-square error $\mathbb{E}[(\hat{p}_\gamma/p_\gamma - 1)^2]$ is*

$$\frac{2}{N}\sum_{k=1}^{K}\left(\frac{1}{G(P_{k-1},P_k)^2} - 1\right) - \frac{2}{N}\sum_{k=1}^{K-1}\left(\frac{G(P_{k-1},P_{k+1})}{G(P_{k-1},P_k)G(P_k,P_{k+1})} - 1\right) + \frac{1-s}{sN} + o\left(\frac{1}{N}\right). \tag{4.8}$$

In particular, if the inverse Bhattacharyya coefficients are bounded such that $\frac{1}{G(P_{k-1},P_k)^2} \leq D$ (with $D \geq 1$), then the asymptotic relative mean-square error satisfies $\mathbb{E}[(\hat{p}_\gamma/p_\gamma - 1)^2] \leq 2KD/N$.

See Appendix B.2 for the proof. We provide some remarks about the above result. Intuitively, the first term in the bound (4.8) accounts for the variance of $\widehat{E}_k$. The denominator of $\widehat{E}_{k-1}$ and numerator of $\widehat{E}_k$ both depend on x_i^k; the second sum in (4.8) accounts for the covariance between those terms. Furthermore, the quantities in the bound (4.8) are all empirically estimable, so we can compute the mean-square error from a single pass of Algorithm 2. In particular,

$$G(P_{k-1},P_k)^2 = \frac{Z_k^B}{Z_{k-1}}\frac{Z_k^B}{Z_k}, \qquad \frac{G(P_{k-1},P_{k+1})}{G(P_{k-1},P_k)G(P_k,P_{k+1})} = \frac{Z_k^C}{Z_k}\frac{Z_k}{Z_k^B}\frac{Z_k}{Z_{k+1}^B}, \tag{4.9}$$

where $Z_k^C/Z_k = \mathbb{E}_{P_k}\left[\rho_k^B(X)\rho_{k+1}^B(X)/\rho_k(X)^2\right]$. The last term in the bound (4.8) is the relative variance of the final Monte Carlo estimate $\sum_i I\{f(x_i^K) \leq \gamma\}/N$.

Overall efficiency The statistical efficiency outlined in Proposition 1 is pointless if it is accompanied by an overwhelming computational cost. We take the atomic unit of computation to be a query of the simulator, which returns both evaluations of $f(x)$ and $\nabla f(x)$; we assume other computations to be negligible compared to simulation. As such, the cost of Algorithm 2 is $N(1+KT)$ evaluations of the simulator.

Our method can exploit two further sources of efficiency. First, we can employ surrogate models for gradient computation and/or function evaluation during the T MCMC steps. For example, using a surrogate model for a fraction $d \leq 1 - 1/T$ of the MCMC iterations reduces the factor T to $T_s := (1-d)T$ in the overall cost. Surrogate models have an added benefit of making our approach amenable for simulators that do not provide gradients. The second source of efficiency is parallel computation. Given C processors, the factor N in the cost drops to $N_c := \lceil N/C \rceil$.

The overall efficiency of the estimator (4.7)—relative error multiplied by cost (Hammersley and Handscomb, 1964)—depends on p_γ as $\log(p_\gamma)^2$. In contrast, the standard Monte Carlo estimator has cost N to produce an estimate with relative error $\frac{1-p_\gamma}{p_\gamma N}$. Thus, the relative efficiency gain for our estimator (4.7) over naive Monte Carlo is $O(1/(p_\gamma \log(p_\gamma)^2))$: the efficiency gains over naive Monte Carlo increase as p_γ decreases.

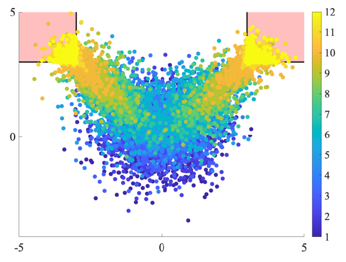
4.5. Experiments

We evaluate our approach on a variety of autonomous vehicle policies, showcasing its use in efficiently evaluating the safety of autonomous systems. We begin with a synthetic problem to illustrate the methodology concretely as well as highlight the pitfalls of using gradients naively. Then we consider an example of using gradient-guided bridge sampling as a tool for engineering design in high-dimensional settings comparing two algorithms which solve the OpenAI Gym CarRacing environment (Klimov, 2016). In particular, we demonstrate how our method may be used to compare the frequency of extreme failures in order to differentiate between two reinforcement learning policies which achieve similar average reward. As our concluding experiment we identify and estimate the probability of failure modes of a real autonomous vehicle system which has driven 20 million miles on public roads. In Part III we highlight the use of the framework and methods developed in this chapter for evaluation problems beyond AV evaluation.

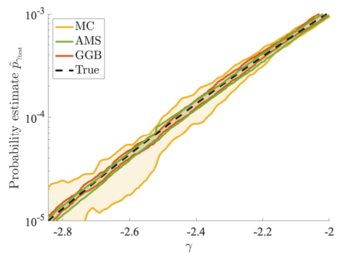
We compare our method, gradient-guided bridge sampling (GGB), with naive Monte Carlo (MC) and adaptive multilevel splitting (AMS) (Bréhier et al., 2015; Webb et al., 2018; Norden et al., 2019). All methods are given the same computational budget as measured by evaluations of the simulator. This varies from 50,000-100,000 queries to run Algorithm 2 as determined by p_γ (see Appendix B.3 for details of each experiment's hyperparameters). Despite running Algorithm 2 with a given γ, we evaluate estimates $\hat{p}_{\gamma_{\text{test}}}$ for all $\gamma_{\text{test}} \geq \gamma$. Note that, larger γ_{test} require fewer queries to evaluate $\hat{p}_{\gamma_{\text{test}}}$ (as Algorithm 2 terminates early). Thus, we adjust the number of MC queries accordingly for each γ_{test}. Independently, we calculate the ground-truth values $p_{\gamma_{\text{test}}}$ for the non-synthetic problems using a fixed, large number of MC queries.

Synthetic problem We consider the two-dimensional function $f(x) = -\min(|x_{[1]}|, x_{[2]})$, where $x_{[i]}$ is the i^{th} dimension of $x \in \mathbb{R}^2$. We let $\gamma = -3$ and $P_0 = \mathcal{N}(0, I)$ (for which $p_\gamma = 3.6 \cdot 10^{-6}$). Note that $\nabla^2 f(x) = 0$ almost everywhere, yet $\nabla f(x)$ has negative divergence in the neighborhoods of $x_{[2]} = |x_{[1]}|$. Indeed, gradient descent collapses $x_i \sim P_0$ to the lines $x_{[2]} = |x_{[1]}|$, and the ill-defined nature of the Hessian makes it unsuitable to track volume distortions. Thus, simple gradient-based transformations used to find adversarial examples (*e.g.* minimize $f(x)$) should not be used for estimation in the presence of non-smooth functions, unless volume distortions can be quantified.

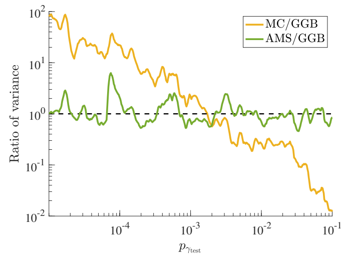
Figure 4.12(a) shows the region of interest in pink and illustrates the gradual warping of ρ_0 towards ρ_∞ over iterations of Algorithm 2. Figures 4.12(b) and 4.12(c) indicate that all adaptive methods outperform MC for $p_{\gamma_{\text{test}}} < 10^{-3}$. For larger $p_{\gamma_{\text{test}}}$, the overhead of the adaptive methods renders MC more efficient (Figure 4.12(c)). The linear trend of the



(a) Samples colored by iteration

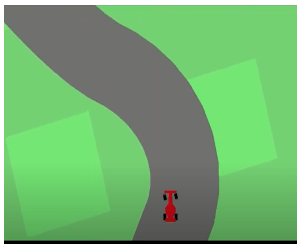


(b) $\hat{p}_{\gamma_{\text{test}}}$ vs. γ_{test}

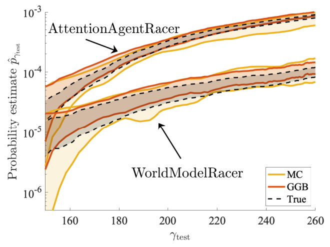


(c) Ratio of variance vs. $p_{\gamma_{\text{test}}}$

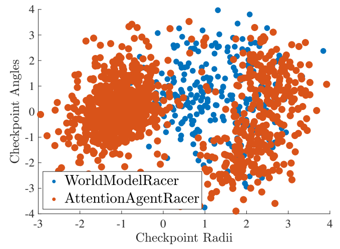
Figure 12: Experiments on a synthetic problem. 10 trials are used to calculate the 99% confidence intervals in (b) and variance ratios in (c). All adaptive methods perform similarly in this well-conditioned search space.



(a) The environment



(b) Failure rates



(c) Failure modes

Figure 13: CarRacing experiments. MC cannot distinguish between the policies below $\gamma_{\text{test}} = 160$. GGB's high-confidence estimates enable model comparisons at extreme limits of failure. Low-dimensional visualization of the failure modes shows that the algorithms fail in distinct ways.

yellow MC/GGB line in Figure 4.12(c) aligns with the theoretical efficiency gain discussed in Section 4.4. Finally, due to the simplicity of the search space and the landscape of $f(x)$, the benefits of gradients are not drastic. Specifically, as shown in Figure 4.12(c), all adaptive methods have similar confidence in their estimates except at very small $p_{\gamma_{\text{test}}} < 10^{-5}$. The next example showcases the benefits of gradients in a more complicated search space.

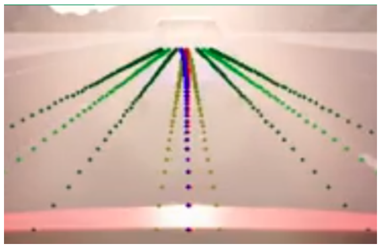
Car racing The CarRacing environment (Figure 4.13(a)) is a challenging reinforcement-learning task with a continuous action space and pixel observations. Similar observation spaces have been proposed for real autonomous vehicles (*e.g.* (Bansal et al., 2018; Luo et al., 2018; Wang et al., 2019a)). We compare two recent approaches, AttentionAgentRacer (Tang et al., 2020) and WorldModelRacer (Ha and Schmidhuber, 2018a) that have similar average performance: they achieve average rewards of 903 ± 49 and 899 ± 46 respectively (mean

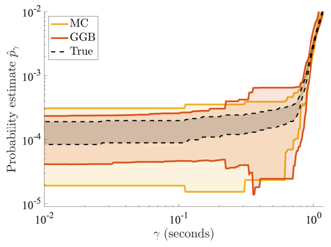
$\pm$ standard deviation over 2 million trials). Both systems utilize one or more deep neural networks to plan in image-space, so neither has performance guarantees. We evaluate the probability of getting small rewards ($\gamma = 150$).

The 24-dimensional search space P_0 parametrizes the generation of the racing track (details are in Appendix B.3). This environment does not easily provide gradients due to presence of a rendering engine in the simulation loop. Instead, we fit a Gaussian process surrogate model to compute $\nabla f(x)$ (see Appendix B.3). As these experiments are extremely expensive (taking up to 1 minute per simulation), we only use 2 million naive Monte Carlo samples to compute the ground-truth failure rates. Figure 4.13(b) shows that, even though the two models have very similar average performance, their catastrophic failure curves are distinct. Furthermore, MC is unable to distinguish between the policies below rewards of 160 due to its high uncertainty, whereas GGB clearly shows that WorldModelRacer is superior. Note that, because even the ground-truth has non-negligible uncertainty with 2 million samples, we only report the variance component of relative mean-square error in Table 5.

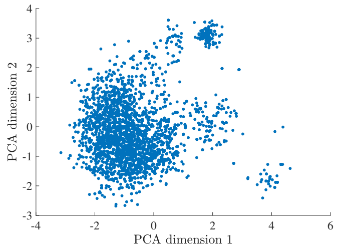
We visualize the modes of failure (defined by $\gamma_{\text{test}} = 225$) via PCA in Figure 4.13(c). The dominant eigenvectors involve large differentials between radii and angles of consecutive checkpoints that are used to generate the racing tracks. AttentionAgentRacer has two distinct modes of failure, whereas WorldModelRacer has a single mode.

OpenPilot The OpenPilot environment (Figure 4.14(a)) integrates the TestPilot fork (Norden et al., 2019) of the OpenPilot driver assistance system with the Carla simulator (Dosovitskiy et al., 2017b). We utilize the same search space as (Norden et al., 2019) to evaluate the performance of the OpenPilot system on highway driving tasks. In particular we sample initial positions for 6 vehicles (including the system-under-test) on a 1 KM section of road which includes both straight and curved portions. In addition each vehicle is given an initial velocity and orientation. The environment vehicles also select a driving policy from a distribution of agent models. Unlike the previous example which makes decisions based on an abstract model of the environment, OpenPilot extracts features from images produced by a


(a) The environment



(b) Failure rates



(c) Failure modes

Figure 14: OpenPilot experiments.

Table 5: Relative mean-square error $\mathbb{E}[(\hat{p}_\gamma/p_\gamma - 1)^2]$ over 10 trials

	Synthetic	AttentionAgentRacer	WorldModelRacer	TestPilot
MC	0.9456	1.0866	0.9508	1.9613
AMS	0.0138	1.0211	0.8177	0.9331
GGB	0.0098	0.9030	0.7837	0.8758
p_γ	$1.0 \cdot 10^{-5}$	$\approx 2.5 \cdot 10^{-5}$	$\approx 9.5 \cdot 10^{-6}$	$\approx 1.4 \cdot 10^{-4}$

front-facing dash camera; thus, weather conditions can significantly affect its performance. In order to capture such effects we also sample cloud cover, sun angle, precipitation, and ground precipitation parameters. See Appendix A of (Norden et al., 2019) for details.

As in the previous experiment, each simulation of the system is expensive taking approximately 20 seconds; moreover, generating simulated camera images and executing the models contained in the perception component of OpenPilot requires the utilization of GPU server instances. Evaluating the system naively is not only time-consuming, but also expensive. The ground truth failure rate is estimated by sampling 150,000 simulations of the combined system. As in Chapter 3 we use the minimum TTC over the simulation as the safety metric.

The results, see Figure 4.14(b) and Table 5, demonstrate that AMS and GGB both significantly reduce the relative mean-squared error of the failure rate estimate when compared to MC with equal computation budget. In Figure 4.14(c) we also demonstrate that GGB (like AMS, see (Norden et al., 2019)) is able to identify four distinct failure modes with no prior knowledge of their existence (unlike the parametric approach presented in Chapter 3).

4.6. Conclusion

There is a growing need for rigorous evaluation of safety-critical systems which contain components without formal guarantees (*e.g.* deep neural networks). Scalably evaluating the safety of such systems in the presence of rare, catastrophic events is a necessary component in enabling the development of trustworthy high-performance systems. Our proposed method, gradient-guided bridge sampling, employs three concepts—exploration, exploitation, and optimization—in order to evaluate system safety with provable statistical and computational efficiency. We demonstrate the performance of our method on a variety of reinforcement-learning and robotic systems, highlighting its use as a tool for continuous integration and rapid engineering design. In future work, we intend to investigate how efficiently sampling rare failures—like we propose here for *evaluation*—could also enable the *automated repair* of safety-critical reinforcement-learning agents.

4.7. Current state-of-the art

Following the completion of this work, Sinha et al. (2020) extended the methods described in this chapter. Although, the theoretical efficiency of the method is unchanged, significant empirical performance was gained. In particular, the accuracy of bridge sampling depends on the overlap between intermediate distributions P_k. Simply increasing the number of intermediate distributions is inefficient, because it requires running more simulations. Instead, Sinha et al. (2020) employ a technique known as *warping*, where intermediate distributions map to a common reference distribution (Voter, 1985; Meng and Schilling, 2002).

Specifically, Sinha et al. (2020) use normalizing flows (Rezende and Mohamed, 2015; Kingma et al., 2016; Papamakarios et al., 2017, 2019), which efficiently transform arbitrary distributions to standard Gaussians through a series of deterministic, invertible functions. Recently, Hoffman et al. (2019) explored the benefits of using normalizing flows for reparametrizing distributions within MCMC; the method proposed in Sinha et al. (2020) encompasses this benefit and extends it to the SMC setting.

Given, this modification, Sinha et al. (2020) show that the cost of Algorithm 2 is $N(1 + KT)$ evaluations of the simulator without warping and $N(1 + KT) + 2KN$ with warping. Thus, the relative burden of warping is minimal, because training the normalizing flows to minimize $D_{\text{KL}}(Q_k \| \mathcal{N}(0, I))$ requires no extra simulations. In contrast, directly minimizing $D_{\text{KL}}(Q_{k-1} \| Q_k)$ would require extra simulations at each training step.

The results of Sinha et al. (2020) show that, qualitatively, if the system-under-test has irregular failure modes with pathological curvature, HMC is difficult for AMS and GGB (Betancourt, 2017). Quantitatively, the effect can be observed in poor mixing of the chain and adversely affects the performance of AMS and GGB.

Part II

Domain Adaptation

CHAPTER 5 : F1TENTH

5.1. Chapter Overview

This chapter is adapted from "F1TENTH: An Open-source Evaluation Environment for Continuous Control and Reinforcement Learning" which appeared in the Post-proceedings of Neural Information Processing Systems 2019 as joint work with Hongrui Zheng, Dhruv Karthik, and Rahul Mangharam.

5.2. Introduction

Unlike supervised learning tasks, the minimal evaluation environment for reinforcement learning-based (RL) approaches to robotics must include more than a labeled dataset. We propose that RL evaluation environments should incorporate: (1) accurate and efficient simulators to reduce the time and cost of testing, (2) accessible hardware to correlate simulation results with real-world performance, and (3) adaptable benchmarks which include strong baselines supported by competition to drive innovation and reduce overfitting. While others (*e.g.* Brockman et al., 2016) have described a subset of these requirements, their efforts classify the validation, realism, and accuracy of simulation tools, and availability of hardware as future goals rather than necessary components. In response, we create an open-source RL evaluation environment (simulator, hardware, benchmarks, baselines, and competitions) based on a scaled autonomous vehicle situated in a racing environment.

Accurate simulation is a necessary component of any *practical* evaluation system, as real-world evaluations are dangerous, expensive, and time consuming. A number of other studies note the sensitivity of RL-based agents to simulation details (*cf.* Lewis et al., 2019; Henderson et al., 2018). Indeed, like all simulators, ours fails to capture agent vehicle dynamics with perfect fidelity. However, we follow classical system identification methods to actively quantify and minimize this discrepancy with respect to performance on a real hardware platform. Additionally, we propose simple methods to more accurately simulate a

2D LIDAR, IMU and cameras in a photorealistic environment.

Hardware-based experiments must support simulation-based evaluation in order to ensure that an approach performs well in reality. As noted by (Yang et al., 2020), the hardware platform used must not limit potential solutions. However, similar benchmarks such as (Balaji et al., 2019) impose sensor configuration and training method restrictions, whereas others (Goldfain et al., 2019) are too expensive to be used widely. In contrast, we offer a full-fledged Open-AI Gym API for controlling a real world vehicle *with* the ability to redefine *step*/*reward* functions in an intuitive manner for numerous sensor configurations. Furthermore, running experiments does not require a special track, as our carefully tuned SLAM packages use the onboard LIDAR to create maps of tracks (*e.g.* a corridor loop), and seamlessly use those maps within simulation. Finally, our hardware is open-source, documented, affordable, and supported by an active community.

In addition to simulation and hardware environments we also define a set of **benchmark tasks** by which to standardize evaluations. The context of these tasks, autonomous racing, aids in the definition of evaluation metrics; unlike day-to-day driving, each episode has a clear winner and is likely to expose the agent to the most difficult regimes of the dynamics. Importantly, the benchmarks naturally include both single agent and multi-agent formulations which are of interest to multiple communities.

Contributions and Organization: In Section 5.3, we compare our simulation environment and hardware support with existing platforms. In Section 5.4, we describe in detail the simulators that serve as a virtual evaluation environment, enabling rapid implementation and dissemination of algorithms in an environment where reproducibility is enforced. In Section 5.5, we describe the low-cost, 1/10th scale vehicle and standardized software infrastructure that supplements our virtual benchmarking tools, enabling the use of a wide range of RL algorithms in reality. In Section 5.6 and 5.7, we describe three benchmarks in our environment, spanning continuous control and RL methods.

5.3. Related work

It is still unknown how deep RL will be used in autonomous vehicles; however, a recent survey (Kiran et al., 2020) highlights numerous promising directions. Nevertheless, the goal of this work is not to describe autonomous driving methods, but rather to build tools which enable a community wide effort. As such, one closely related body of literature includes autonomous vehicle simulators (*e.g.* Shah et al., 2017; Dosovitskiy et al., 2017a; Quiter and Ernst, 2018; LG Electronics, Inc., 2019). Although these tools are widely used in deep RL research, none specifically include a corresponding vehicle platform. More general simulation environments used within the RL community (*e.g.* Todorov et al., 2012; Coumans and Bai, 2016) utilize multi-body physics. In contrast, our simulator is narrow; it uses simple car-like vehicle dynamics described as continuous ODEs. While our domain of interest is targeted, the task is still challenging due to the non-holonomic behavior of the robot, the use of high-dimensional measurements, and interactions with external dynamic agents.

Other related work includes open-source implementations of full-scale autonomous vehicle stacks (Baidu Apollo Team, 2017; Kato et al., 2018). Our approach differs because we prioritize the use of scaled, inexpensive hardware that does not require special permission or insurance for operation. Of course, there are now many scaled autonomous vehicle projects available to the community (*cf.* Fishberg et al., 2019; Gonzales et al.; Roscoe, 2019; Srinivasa et al., 2019; Goldfain et al., 2019). The F1TENTH hardware is most similar to RACECAR, and in fact shares some low-level drivers; however, F1TENTH differs in its simulation capabilities which are more easily extended beyond pedagogical settings. Our goal of creating a standardized evaluation system is shared with (Balaji et al., 2019). While the DeepRacer platform has a lower upfront cost, the lack of sensing modalities such as LIDAR, non-standard hardware acceleration for neural network components, and closed nature of the simulation system limit the use of DeepRacer in research settings.

5.4. Simulation Tools

In this section we describe the containerized simulation engine for the 1/10th scale vehicle platform of Section 5.5. We note that simulated environments include a *reality gap* which is characterized by the discrepancy between the performance of the system obtained in simulation relative to the deployed performance. We describe how our simulator design can mitigate and control factors which influence the reality gap.

5.4.1. Simulator design

The simulator can operate in two modes: (1) a ROS-based mode designed for software-in-the-loop (SIL) testing and (2) an OpenAI Gym (Brockman et al., 2016) mode designed for distributed training. The OpenAI Gym simulator mode supports the additional option to generate photo-realistic camera observations which enables the testing and training of a computer vision pipeline.

The SIL-oriented simulator is a plug-and-play replacement for the 1/10th scale platform itself. The response of the vehicle to control commands is captured using the single track model described in (Althoff et al., 2017). The simulator also replaces the sensors used on the actual vehicle (see Section 5.5). Synthetic sensors mimic odometry information from the electronic speed controller and laser scans from LIDAR. The same message types and ROS publisher subscriber mechanisms are used to communicate the sensor observations. The primary use-case for this simulation mode is debugging shortly before deployment as implementation errors related to ROS and the vehicles interface can be found safely.

The training-oriented simulator is created with the RL community's needs in mind. In contrast to the SIL mode, training mode explicitly steps time with the given action input in order to create deterministic rollouts. Since the simulator is capable of simulating multiple vehicles and their sensor observations, the API is designed such that all agents are stepped simultaneously. Determinism has two benefits; first, experiments are repeatable, a feature useful both for debugging and in service of our stated goal of creating fair and accurate evaluations. Second, the explicit stepping enables the physics engine to take advantage of faster than real-time execution (up to a 6x speedup per-worker on commodity cloud instances). In addition, we provide a containerized executable of the simulator which includes a ZeroMQ (Hintjens, 2013) interface implementing the MapReduce pattern (Dean and Ghemawat, 2008) for scaling distributed training algorithms. Finally, we implement a bridge to the vehicle's ROS interface which enables easy deployment after training has completed, see Figure 5.16(c). Unlike other work (Balaji et al., 2019) our system is open source and can be deployed on any cloud-service.

Depending on the sensor payload carried on board the vehicle it may be necessary for the simulator to simulate a camera. Thus, the training-oriented simulator can also provide synthetic images; our implementation utilizes Unreal Engine 4 which we access via a plugin, UnrealCV (Qiu et al., 2017). We discuss the specifics of our modeling choices via an example environment which mimics a track instance used for experiments (see Figure 15) in Section 5.4.2. A benchmark and baseline implementation for the training mode simulator with synthetic image generation is described in Section 5.7.2.

5.4.2. Addressing the reality gap

In this section we consider two sources of error in simulated evaluations of vehicle performance. The first source is the vehicle dynamics model itself. The simulator provides variable vehicle model parameters including: mass, center of mass, moment of inertia, dynamic friction coefficient between the four tires and the environment surface, cornering stiffness, and maximum acceleration/deceleration rates. In order to address the reality gap from a system dynamics perspective, the simulator contains parameters for the vehicle and test environment identified (see Figure 5.16(b)) using the methods outlined in (O'Kelly et al., 2019). The second source of error, sensor model fidelity, especially the synthetic camera is a more pernicious factor. We utilize natural image statistics (Kundu, 2016), a measure of the spatial complexity and color content of an image, to quantify the distributional shift between simulation and reality. In order to minimize the gap, we adjust the renderer's lighting tem-

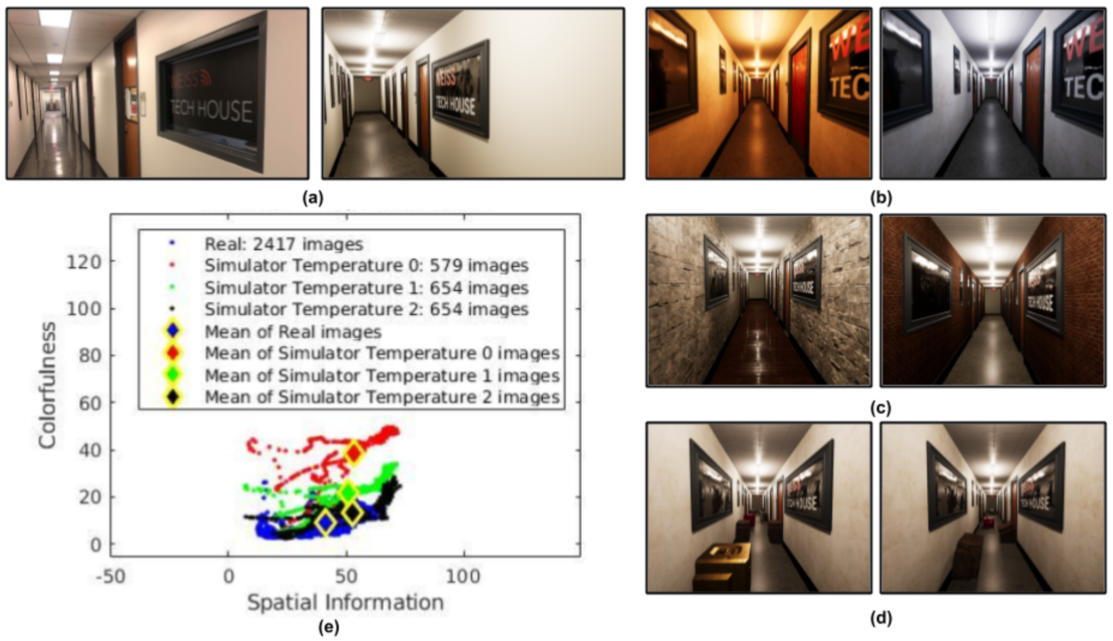
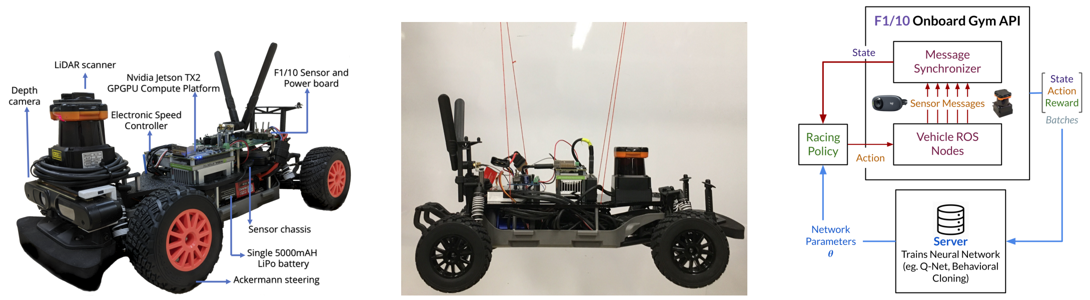


Figure 15: (a) Left: reality, right: simulation (b) DR via rendering parameters (Temperature) (c) DR via texture replacement (d) DR via static obstacle generation (e) Tuning the rendering parameters (temperature) with natural image statistics

perature. In addition, our plugin implements a variety of domain randomization techniques (Tobin et al., 2017) which include renderer settings, texture replacement, and procedural generation of obstacles in order to improve generalization in the real-world . Finally, issues of the environment geometry mismatch are addressed by including a state-of-the-art SLAM front end which enables the capture of maps (Hess et al., 2016). Figure 15.(a) compares the same hallway in reality and in the simulation; (b), (c), and (d) in Figure 15 shows different strategies of domain randomization (DR), and Figure 15.(e) details the statistics of real images collected on the vehicle relative to a variety of simulation settings.

5.5. Hardware Specification and Middleware

The hardware platform, shown in Figure 5.16(a), consists of two halves: a widely available 1/10-th scale RC car modified to accommodate autonomous control inputs as well as a sensor and compute platform which enables autonomous decision making. The first half, known as the base chassis has 1. A high torque brushless DC motor. 2. An electronic speed



(a) Vehicle components (b) System Identification (c) Onboard Gym API

Figure 16: Major components of the F1TENTH platform and SystemID procedure.

controller (VESC) based on an open source design (Vedder, 2015) to convert speed input into motor RPM. 3. A servo motor for controlling the Ackermann steering, 4. A Lithium Polymer (LiPo) battery pack.

The second half is called the sensor and compute platform; it consists of the sensor payload, power management board, and the on-board computer.

The power management board provides a stable voltage source for the compute unit and its peripherals since the battery voltage varies as the vehicle is operated. The sensor suite includes a Hokuyo UST-10LX LIDAR, selected for its durability and high maximum scan frequency of 40Hz. The platform includes options to mount an Intel RealSense D435i which provides RGBD images and an integrated a IMU. Odometry is provided by the VESC. The default compute unit is the Jetson TX2, and there is additional support for the Xavier and Nano variant of the Jetson. The TX2 is mounted on a carrier board to reduce the form factor.

Finally, the platform also includes an open source middleware stack comprising: a driver that provides the interface between on-board ROS nodes and the hardware in order to obtain sensor information and actuate the vehicle; configuration files that define the lower level motor actuation and odometry estimation; and an OpenAI Gym interface as shown in Figure 5.16(c) that enables quasi-deterministic stepping and resetting of the physical car.

5.6. Benchmark Tasks and Competitions

The autonomous driving task encompasses a subset of general navigation and locomotion problems. Through the lens of autonomous racing (AR) we further partition the navigation problem into three distinct settings of increasing complexity. Note that our benchmark environments allow the use of alternate rewards for training, but submitted solutions are measured by a simple, sparse reward: lap time.

Benchmark 1: Static Kinodynamic Planning: The first benchmark environment involves a known race-track and a single vehicle; the objective of the autonomous vehicle is to navigate the track in a minimal amount of time; the task is considered successful only if the trajectory is collision free. Many planning algorithms assume that it is possible to observe the vehicle's full state and identify accurate polyhedral representations of obstacles. In practice, an accurate map and localization system can justify the assumption in this task. We describe a baseline solution based on evolution strategies implemented in Section 5.7.1.

Benchmark 2: Online Kinodynamic Planning: The importance of hardware-based benchmarks (at least as motivation for assumptions) is made more clear if we consider a relaxation of **Benchmark 1**; the map or equivalently the location of obstacles (still polyhedral) is not known prior to runtime. The agent must detect and avoid static obstacles in real time. Furthermore, due to occlusions and sensor properties the vehicle's pose restricts a precise representation of the full map. We describe an imitation learning-based baseline for this problem in Section 5.7.2.

Benchmark 3: Multi-agent Racing: Another advantage of the AR setting is that it naturally extends to multi-agent robotics problems. In the third benchmark task we consider two or more agents driving simultaneously on the same map. We detail a suite of control theoretic baselines in Section 5.7.3.

We curate hardware-tested baseline solutions to the described benchmarks by organizing biannual competitions (Mangharam et al., 2018a,b, 2019a,b). Each competition includes races (**Benchmark 1,3**) and qualification rounds (**Benchmark 2**). We support entrants by providing tools (described in this paper) and teaching material (Agnihotri et al., 2020).

5.7. Baseline Solutions

5.7.1. Benchmark 1: Kinodynamic planning

In **Benchmark 1**, the environment consists of the racetrack encoded as a map, a random starting position, and a reward function which measures lap-time. The agent, through interactions with the environment, tunes a parameterized model of its dynamics, strategy, and controller, with the objective of finding an assignment of these parameters that minimizes the lap-time. We employ the covariance matrix adaptation evolutionary strategy (CMA-ES) (Hansen et al., 2003) optimizer due to the non-smooth objective function landscape. This baseline straddles continuous control and RL methods; the parameters of the vehicle's model predictive controller are tuned based on the results of rollouts which evaluate a candidate solution's quality. A detailed discussion of the learning problem and approach is given in Chapter 6.

The results comparing simulated lap times and the actual lap times are shown in Table 6. In all the experiments, the lap times measured on hardware rollouts are higher than the simulated lap times. Several factors contribute to this discrepancy: the friction coefficient between the driving surface and the tires is not constant, the steering delay is time-varying, and the accuracy of localization varies depending on the vehicle's location in the map. Each of these factors causes the ego-vehicle to deviate from the optimal trajectory found in simulation. Importantly, however, the local planning and feedback control methods which are presented in greater detail within Section 5.7.3 allow the solutions found in simulation to remain feasible in the real-world. Moreover, the difference in lap times between sim and real is small, 0.61% for the test track, demonstrating the accuracy of the simulator.

Table 6: Best Solution Lap Times in Sim vs. Real

Method	Lap Time (s, sim)	Lap Time (s, real)
TunerCar	16.2376 ± 0.2161	16.19
Pure Pursuit	17.2307 ± 0.0620	17.00

5.7.2. Benchmark 2: Online kinodynamic planning

In **Benchmark 2**, a random number of static obstacles are dispersed around the track which the agent will navigate. The agent does not have prior knowledge of the static obstacles' positions in the map before being deployed. Our baseline implementation for this benchmark illustrates a method to transfer a reactive LIDAR based obstacle avoidance policy to a vision based policy via self-supervised imitation learning (SSIL). The oracle policy, based on the follow the gap method (FGM) (Sezer and Gokasan, 2012) uses the most recent LIDAR scan to avoid the nearest obstacle and steer toward the largest gap. We define I as the stack of input images, S as the snapshot of the current laser scan, a_o as the steering angle output from the oracle policy at time t, and a_v as the steering angle output generated by the vision-based agents at time t. We utilize the network architecture described in (Bojarski et al., 2016) and trained on tuples (I, a_o). Given a trajectory $(I_0, S_0, a_{v,0}), (I_1, S_1, a_{v,1}) \dots (I_T, S_T, a_{v,T})$ we use S_t and FGM to relabel each $a_{v,t}$ to $a_{o,t}$. The relabeled trajectories are used to iteratively correct, via DAgger (Ross et al., 2011), the policy as the vehicle explores the state space.

In the simulated training phase, every rollout occurs in a domain randomized environment where obstacle placement, simulator asset textures, and lighting temperature are varied. Following the simulation-based training phase; the vehicle is deployed in the environment and trained in the same manner *in situ*. After the agent is capable of completing the real track without any collision, it is evaluated in the same environment with the learned vision only policy. We show the resulting average lap times of the imitation policy and the oracle policy on the same Philadelphia Track in 5.7.1 in Table 7. SSIL DAgger produces an average lap time 5% slower than that of the FGM Oracle Policy, indicating that the vision-based policy mimics the LIDAR based policy after sufficient DAgger iterations.

Future baselines should explore the ability of imitation learning approaches to exceed the performance of oracle policies.

Table 7: Performance of SSIL and DAgger on Benchmark 2

Method	Avg. Lap time (s, sim)	Avg. Lap time (s, real)
SSIL DAgger	22.625 ± 0.1092	26.71
FGM (Oracle)	21.55 ± 0.0967	18.24

5.7.3. Benchmark 3: Multi-agent racing

For **Benchmark 3** we evaluate reactive, randomized, and model predictive approaches to multi-agent racing. The reactive method uses the same FGM policy described in Section 5.7.2. The probabilistic planning method is based on rapidly-exploring random trees (RRT) (LaValle, 1998). The RRT planner randomly samples points in the workspace of the vehicle and then expands a tree to find a path between the car's current position and goal position. We also demonstrate a variant of RRT, RRT* (Karaman and Frazzoli, 2011), which rebuilds the tree structure based on the path traversal cost such that the selected path asymptotically approach the global minimum cost. The solutions provided by RRT and RRT* are not inherently dynamically feasible; thus, an additional trajectory tracking controller (Coulter, 1992) is utilized. The model predictive control approach, lattice planning, and has been demonstrated on full scale autonomous vehicles (Urmson et al., 2008). The formulation of the MPC trajectory generation component in the planner is described in (McNaughton, 2011). Online, the lattice planner samples a set of goal poses in the ego-vehicle's moving reference frame, for each sample a trajectory, parameterized a cubic polynomial, which drives the vehicle from its current state to the goal is computed. Given the set of sampled trajectories, each is evaluated for feasibility and progress. This process repeats in a receding horizon fashion.

Table 8 reports the average lap times, crash rate, and win rate of baseline solutions on **Benchmark 3**. Unlike the baseline provided in the previous section, the algorithms investigated all originate from motion planning and continuous control. In these methods the opponent is treated as a dynamic obstacle rather than an adversarial agent. Thus, although all the baselines have win rate higher than 50%, they all have non-zero crash rates due to their relatively unsophisticated prediction mechanisms as compared to RL and game-theoretic approaches (*e.g.* Liniger and Lygeros, 2019). Further work, presented in Chapter 7, extends the multi-agent benchmark to balance safety and performance within our evaluation environment.

Table 8: Performance of Local Planners on Multi-agent benchmark

Method	**Avg. Lap time (s)**	**Crash Rate**	**Win Rate**
FGM	18.647 ± 2.65	50%	50%
RRT	17.14 ± 2.11	45%	55%
RRT*	13.91 ± 0.64	35%	65%
Lattice Planner	15.59 ± 1.22	30%	70%

5.8. Conclusion

This chapter details the F1TENTH platform, an open-source evaluation framework with virtual environments and a low-cost hardware counterpart which enables safe and rapid experimentation suitable for laboratory research settings. We present three benchmark tasks and baselines in the setting of autonomous racing, demonstrating the flexibility and features of our evaluation environment. A limitation of this work is the relatively small sample size of algorithms included in our baseline examples; however, the goal of releasing this environment is to generate a wider variety of community submitted solutions. In what follows we utilize the platform developed in this chapter in order to address the domain adaptation problem. In Chapter 6 we present a solution for Benchmark 1, and in Chapter 7 we present a solution for Benchmark 3.

CHAPTER 6 : TunerCar

6.1. Chapter Overview

This chapter is adapted from "TunerCar: A Superoptimization Toolchain for Autonomous Racing," which appeared in the Proceedings of the International Conference on Robotics and Automation 2020 as joint work with Hongrui Zheng, Achin Jain, Joseph Auckley, Kim Luong, and Rahul Mangharam.

6.2. Introduction

Since its inception, racing has been a key driver of new technology in the automotive industry. Some domains, such as powertrain engineering, are obvious beneficiaries of racing development. However, the goals of racing – fast and aggressive driving – are seemingly orthogonal to the safety-oriented specifications of the autonomous vehicle industry. Nevertheless, scenarios faced by racing drivers (and autonomous racers) force the development of technology which must operate safely in both nominal conditions and, more importantly, at the limits of vehicle performance.

The objective of developing an optimal autonomous racer is motivated by the desire to create safe and reusable core autonomy components, namely vehicle and environment agnostic planning and control software. Racing, in this context, is a mechanism to create a competitive environment where the quality of the chosen vehicle configuration has a clear measure –

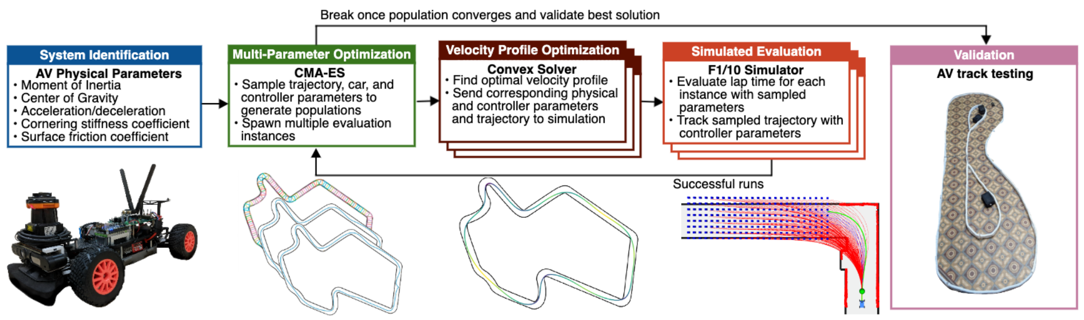

Break once population converges and validate best solution
System Identification
AV Physical Parameters
• Moment of Inertia
• Center of Gravity
• Acceleration/deceleration
• Cornering stiffness coefficient
• Surface friction coefficient
Multi-Parameter Optimization
CMA-ES
• Sample trajectory, car, and controller parameters to generate populations
• Spawn multiple evaluation instances
Velocity Profile Optimization
Convex Solver
• Find optimal velocity profile
• Send corresponding physical and controller parameters and trajectory to simulation
Simulated Evaluation
F1/10 Simulator
• Evaluate lap time for each instance with sampled parameters
• Track sampled trajectory with controller parameters
Validation
AV track testing
Successful runs

Figure 17: The TunerCar toolchain which jointly optimizes racing strategy, planning methods, control algorithms, and vehicle parameters for an autonomous racecar.

lap time. While nominal conditions may be handled even with poorly integrated components (*e.g.* a pure pursuit controller which works even when accidentally used on the wrong robot (Coulter, 1992)), racing conditions severely punish sub-optimal vehicle dynamics, racing strategy (path and speed selection) and controller parameters.

This paper introduces the notion that component reuse and adaptation is analogous to creating a compiler-like tool that targets computational, physical, and external environmental details of a robot's operational domain. In general, the goal of a compiler is to validate and then transform a source program from one language to another (usually lower-level *e.g.* assembly) which is suitable for the target domain (Aho et al., 1986). Modern optimizing compilers (Lattner, 2002) also seek to improve the performance of the transformed program. To concretize the analogy, we define the source program as a parameterized description of the vehicle dynamics, tracking controllers, and local planning method which we wish to transform to perform safely and efficiently in the operational environment represented by a map, physical laws, and sensing capabilities.

We propose a solution to this parameter search problem which is related to the concept of *superoptimization* (Schkufza et al., 2013; Massalin, 1987; Solar-Lezama, 2013; Liang et al., 2010), a technique that searches the space of equivalent and correct programs for the most performant instance rather than applying a sequence of optimization passes which attack specific performance bottlenecks. Despite the success of superoptimization approaches in narrow domains (Vasilache et al., 2018; Ragan-Kelley et al., 2013), viewing the autonomous vehicle as the compilation target creates entirely new issues due to the cyber-physical nature of the platform. First, no simulator or model can perfectly emulate the target thus creating a noisy performance measure and potentially falsifying correctness claims. Second, addressing this reality gap by instead executing the proposed program transformations on the vehicle is dangerous, slow, and expensive. The vehicle may get damaged due to aggressive strategies, tests cannot proceed faster than real-time, and evaluations cannot be easily parallelized. In response to these challenges, our solution provides a validated, deterministic, and parallel simulation environment as well as a method of adapting derived strategies to reality via an efficient online optimization component.

This work has three primary contributions. First, we provide a toolchain for superoptimization of autonomous racers called TunerCar (see Figure 17). This toolchain includes target hardware, modular software, and a calibrated simulator. Second, we describe a methodology for *tuning* a high-dimensional set of hyperparameters spanning control, planning, dynamics, and strategy. Finally, we validate our solution on an AV software stack and 1/10th-scale open-source vehicle described in Chapter 5.

In what follows, we demonstrate that TunerCar achieves improvement in lap time relative to other approaches tested given a fixed computation budget, and that our solution exceeds the performance of crowd-sourced expert solutions by up to 21%. Section 6.3 describes the problem setup and optimization pipeline. Section 6.4 places our solution in context with previous approaches. Section 6.5 describes the modular autonomy stack, simulator, and hardware utilized for large-scale experiments detailed in Section 6.6.

6.3. Methodology

6.3.1. Problem statement

Our goal is to determine the best vehicle parameters, racing strategy, and controller settings that minimize lap time for a given racing track (also referred to as map). Mathematically, we define the objective function as the lap time, $f(\Theta) : \mathbb{R}^n \to \mathbb{R}$. Here lap time is computed by simulating the system producing a trajectory. The search space parametrized by Θ is the concatenation of three components – vehicle dynamics, racing strategy, and controller. Note that in this work we utilize a simulator (black-box) where the resulting scalar measure is deterministic given an assignment of parameters $\theta \in \Theta$ but may be noisy relative to *in situ* executions. We do not make any other assumptions such as the existence of a gradient. Our approach is shown in Figure 17. In what follows we define a parameterization of the autonomous racer, describe an evaluation criteria, and detail a method to sample and optimize realizations of the parameters.

6.3.2. Search space

The physical parameters Θ_p are defined as the mass, m, the location of the center of gravity in the longitudinal direction, l_g, the friction coefficient, μ_s, the height of the center of gravity, h_g, front cornering stiffness, C_{α_f}, and rear cornering stiffness, C_{α_r}. We limit the range of these parameters such that they are physically achievable without major modifications. The nominal values and their ranges are based on system identification performed on the actual vehicle, for further details see Appendix C.0.1.

The strategy parameters Θ_s are defined by the nominal path (x, y) and the velocity profile (v_x, v_y); the size varies depending on the track, for example, $\Theta_s \in \mathbb{R}^{260}$ on *Philadelphia Track A*. The path and velocity profile combined represent the largest portion of the search. To improve the convergence of our proposed optimization method, we note that a deterministic assignment of the optimal velocity profile is possible given a path x, y, reducing the size of the search space by 1/3. Specifically, the optimal velocity profile can be computed in polynomial time by solving a convex minimum-time parametric optimization problem. We refer the reader to (Lipp and Boyd, 2014) and Appendix C.0.2 for full details of the minimum time path traversal formulation.

Finally, the driver parameters Θ_d represent aspects of the control algorithms used. Manipulating these parameters affects how the vehicle tracks the raceline. There are three components: the waypoint lookahead gain g_w, the planning horizon d_l, and the speed gain v_g (a precise definition is given in Section 6.5.3).

6.3.3. Evaluation criteria

For a given set of parameters, θ, i.e. waypoints with velocities, vehicle dynamics and controllers, we calculate the lap time, $f(\theta)$, using our simulator detailed in Section 6.5. When the simulator receives a new sample, the parameters are updated, the environment is reset, and the virtual vehicle attempts to traverse the track. The result, simulated lap time, is used as the objective function which sorts a set of samples (or population) into quantiles. Solutions are determined to be feasible if the trajectory does not intersect with the boundaries of the track (accounting for the car's length and width). Infeasible solutions are rejected and replaced at the sampling stage described in the next subsection.

6.3.4. Optimization

Our proposed optimization problem is high-dimensional, non-convex, non-smooth, and lacks a closed-form expression of the dynamics. Thus, we use gradient-free black-box optimization techniques. A potential pitfall of heuristic optimization methods applied to this class of problem is the inherent tension between exploration of the search space and exploitation of solutions due to the potential existence of local minima. We utilize a method known as covariance matrix adaptation evolution strategies (CMA-ES), (Hansen and Ostermeier, 2001); our implementation is described in Algorithm 3. CMA-ES is known to perform well in challenging regimes with many local minima because it explicitly balances exploration with hill-climbing. CMA-ES adapts the covariance matrix (increases the L^2 norm) of the proposal distribution when the top performers of the population are far from the rest of the population, and conversely narrows the spread of the proposal distribution as the population becomes less diverse. Additionally, this approach also allows massively parallel evaluation of solutions, the most computationally expensive aspect of this pipeline.

As described in Algorithm 3, we first randomly initialize a population of feasible solutions using uniform sampling. Once the population has been evaluated, the top $\mu\lambda$ solutions are isolated. In Section 6.6, we detail the results of a grid search to identify high-performing, generalizable settings for μ and the population size, λ. The top quantile of solutions is used to fit a multivariate Gaussian distribution from which the next generation of samples is drawn. We repeat this process, terminating when the L^2-norm of the covariance matrix, C^g, is less than $\epsilon = 0.01$ (see Section 6.6 and (Hansen and Ostermeier, 2001)).

Algorithm 3 Covariance matrix adaptation evolution strategy (CMA-ES) (Hansen and Ostermeier, 2001)

```
1: input: population size $\lambda$, parent number $\mu$
2: randomly initialize population: $\theta_k$ for $k = 1, ..., \lambda$
3: calculate population means $\hat{\theta}^g$ of the parameters
4: while termination criterion not met do
5:     evaluate current population $\theta^g$ using $f(\theta)$
6:     sort $\theta$ from smallest to largest objective, $f(\theta)$
7:     isolate top $K = \mu\lambda$ individuals: $\theta_k$ for $k = 1, .., K$
8:     estimate the covariance matrix $C^{(g+1)}$ using $\hat{\theta}^g$
9:     calculate the means $\hat{\theta}^{(g+1)}$ of $\theta_k$ for $k = 1, .., \mu$
10:    sample a new population of $\lambda$ individuals via $\hat{\theta}^{(g+1)}$ and $C^{(g+1)}$
11: end while
```

6.4. Related Work

This paper presents an approach to heterogeneous component integration and optimization suitable for robotic systems that interact with a physical environment. A complete discussion of the general superoptimization and compiler literature has been omitted. However, specific approaches such as program sketching, population-based training, and program synthesis are of interest to the robotics community. We also consider a narrower view of this work relative to numerous methods for optimizing specific vehicle components and software within the racecar engineering community.

Sketching (Solar-Lezama, 2013) utilizes fragments of programs which capture macroscopic details about the structure of the solution in order to synthesize the low-level details. In contrast, program synthesis (Piterman et al., 2006) attempts to construct a correct program from scratch using only formal specifications. Neither approach inherently considers optimality, only correctness. Examples of sketching (Campos et al., 2019) and program synthesis (Liu et al., 2013) techniques applied to robotics generally fail to address the gap between models of robotic systems and realizable interactions between the program and the world. Even still, a variety of computational complexity and undecidability results (Alur et al., 2000) remain a barrier to applying these methods to realistic systems. Another closely related approach, population-based training (PBT) (Jaderberg et al., 2017) has recently been utilized to search for efficient neural network designs; however, earlier work (Cheney et al., 2014) focused only on physical robot design, a subset of the problem addressed in this paper. Likewise, (Ha and Schmidhuber, 2018b) directly addresses the synthesis of a controller and planner interface but does not address the real-world applicability of the approach, specifically with respect to the (lack-of) realism in the simulator and vehicle dynamics. Like (Ha and Schmidhuber, 2018b) the approach presented in Section 6.3 utilizes a black-box optimization method, CMA-ES (Hansen, 2016).

In general, race car optimization has historically been divided into two main categories: vehicle parameter optimization and racing trajectory optimization. The former uses a fixed track and fixed trajectory while varying the car parameters while the latter uses a fixed track and a fixed car while varying the trajectories. In (Kasprzak et al., 1998), 500 pairs of roll stiffness and weight distribution values are evaluated using Pareto minimum analysis to optimize for the fastest lap time. More recently, (Hacker et al., 2000; Castellani and Franceschini, 2003; Hayward, 2007), feature expanded design spaces, parallel computation, and utilize simple genetic algorithms.

In raceline optimization, several approaches have been proposed. An iterative two-step algorithm that separates the longitudinal and lateral control components is explored in (Kapania et al., 2016). Alternatively, in this paper, we use a population-based guided search using CMA-ES to determine the best raceline. In this process, we use the result from (Lipp and Boyd, 2014) that calculates both lateral and longitudinal forces required to minimize time to traverse a *fixed trajectory.* Concurrent work uses Bayesian optimization in place of CMA-ES (Jain and Morari, 2020). An alternative approach (Liniger et al., 2015), attempts to separate the autonomous racecar performance into two steps. First an approximate solution to the high-level path planning problem is computed then the lap time is minimized using a model predictive controller (MPC) in a receding horizon fashion. Similarly (Rosolia and Borrelli, 2019) formulates raceline optimization as an MPC problem using a learned dynamics model. Like these works we also perform system identification

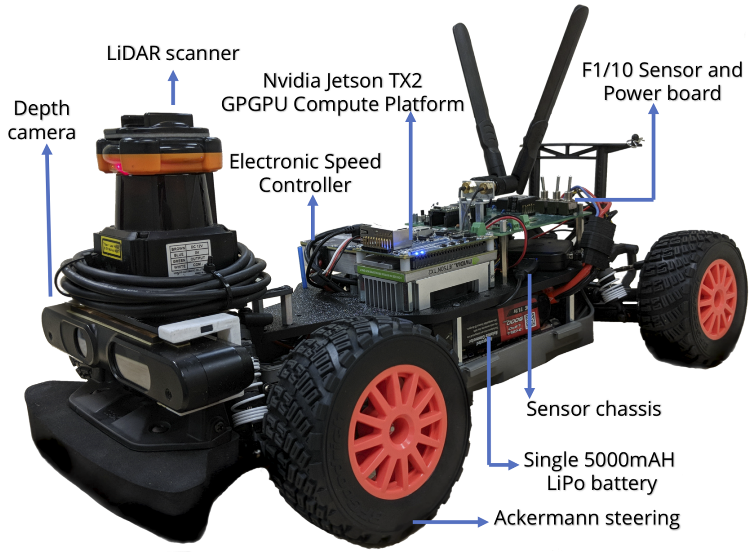

LiDAR scanner
Depth camera
Nvidia Jetson TX2 GPGPU Compute Platform
Electronic Speed Controller
F1/10 Sensor and Power board
Sensor chassis
Single 5000mAH LiPo battery
Ackermann steering

Figure 18: Target 1/10th scale vehicle.

and incorporate model predictive control; however, we generalize the raceline optimization problem to explicitly consider the system as a whole.

6.5. Simulator and Hardware

TunerCar includes a complete set of system components which are utilized to demonstrate the approach. In this section we outline the vehicle hardware, simulator, and vehicle software. The target hardware, an open-source 1/10th scale autonomous vehicle created by the authors, is mapped to a validated, deterministic simulator capable of modeling both the dynamics and sensors included on the vehicle. The simulator includes a wrapper which enables a distributed approach to optimization of the source program with low communication overhead. In addition, the toolchain provides performance oriented implementations of the core algorithms (see Section 6.5.3) and a method to update their parameters.

6.5.1. Hardware

The vehicle, shown in Figure 18, is designed around a ready-to-run RC car chassis. A power board used to manage the onboard compute and sensors as well as mounting plate design are provided. All aspects – hardware, mechanical design, software – of these additional parts are open source. Computation occurs on the onboard NVIDIA TX2, a modern embedded system on a chip (SoC) which contains a conventional multicore ARM CPU in addition to a power-efficient GPU. The main sensor is a planar laser scanner (or LIDAR) which can capture range measurements. The LIDAR enables the vehicle to localize, estimate odometry, and create maps. Due to the operating environment (typically corridors with few features), we supplement the LIDAR measurements with additional odometry information from the electronic speed controller (Vedder, 2015). An optional RGB-D camera provides additional sensing modalities, but is not used in these experiments.

6.5.2. Simulator

The simulator uses a lightweight 2D physics engine written in C++ which implements the single track vehicle model described in (Althoff et al., 2017). System identification was performed to determine vehicle parameters – mass, center of mass moment of inertia, friction coefficient, cornering stiffness, and maximum acceleration/deceleration rates, Appendix C.0.1 contains further details. The moment of inertia was estimated using the bifilar (two-wire) pendulum method (Soule and Miller, 1934). Tire parameters were found using the PAC2002 Magic-formula model. A force scale was used to measure the kinetic friction coefficient between the rubber tires and linoleum floor as the vehicle was dragged laterally at a constant velocity. In addition to modeling vehicle dynamics, the simulator detects collisions between the vehicle and obstacles in the environment in linear-time using a pre-computed lookup table of range measurements (Felzenszwalb and Huttenlocher, 2012). This method is also used to simulate the vehicle's LIDAR.

The simulator differs significantly from existing ROS-based tools. To create deterministic rollouts, the C++ executable is wrapped in the OpenAI Gym (Brockman et al., 2016) environment which explicitly steps time when all inputs have been computed. A further benefit of this approach is the ability to take advantage of faster than real-time execution. On commodity hardware, simulations proceed at approximately 6x real-time. Finally, the simulation environment provides a method for the members of the vehicle dynamics parameter class to be modified by TunerCar at the beginning of each rollout. Last, the simulation

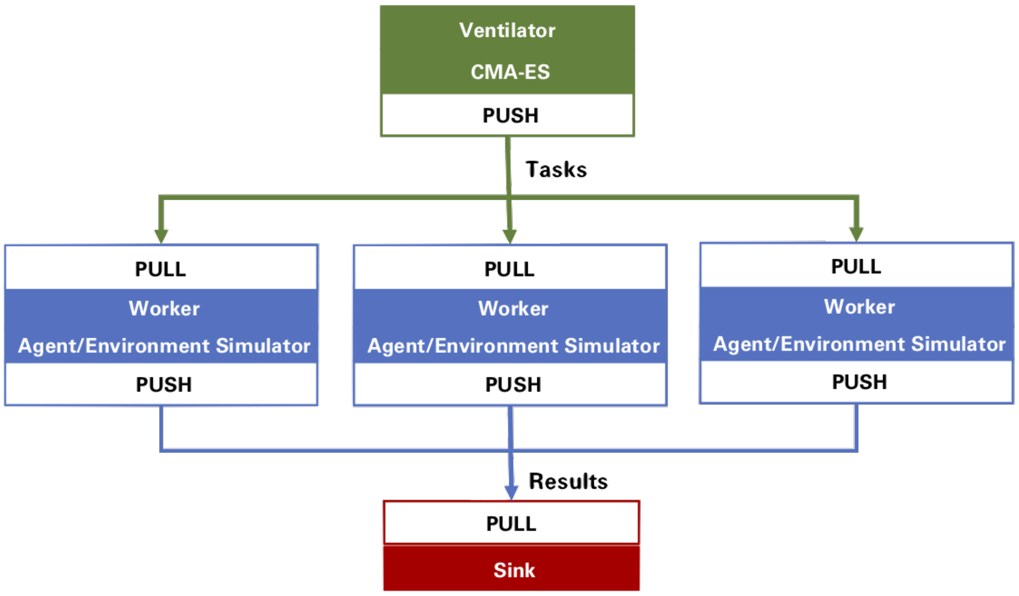


Figure 19: MapReduce pattern of the search implementation.

toolchain include a front end for generating random maps and pre-processing occupancy grid maps created using SLAM.

The TunerCar toolchain employs a MapReduce (Dean and Ghemawat, 2008) messaging pattern implemented using ZeroMQ (Hintjens, 2013), a high-performance asynchronous messaging library. Figure 19 shows many 'workers' spawned in parallel, each equipped with a simulator, velocity optimizer, and full vehicle stack. Samples from the CMA-ES search node are then broadcast to a pool of worker nodes. On receipt of a new sample, the worker instantiates vehicle and environment parameters and simulates a rollout to compute the lap time. In order to update the searcher's sampling distribution, the worker simply transmits the lap time and task index to a sink node which collects the worker results and synchronizes the search epochs.

6.5.3. Vehicle Software

Some algorithms are considered adaptable; others such as the simultaneous localization and mapping package (SLAM) are only used offline to create a model of the environment and, thus, are not tuned. The complete set of non-adaptable algorithms and software includes

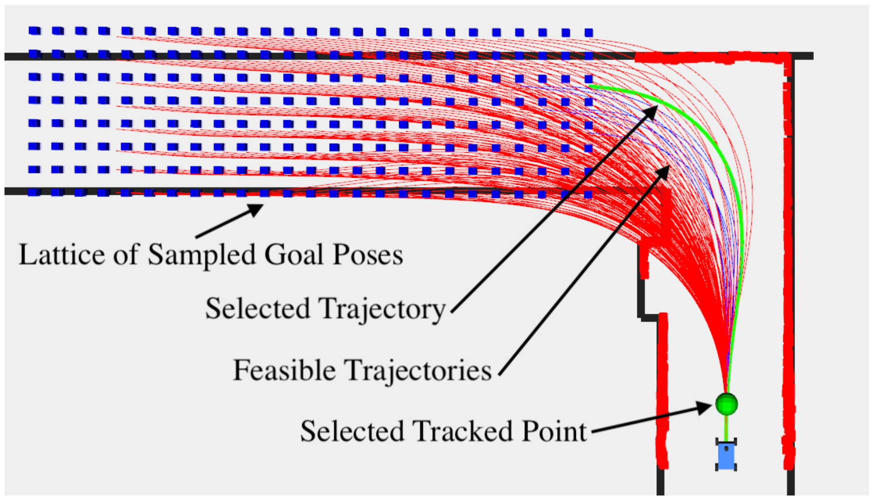


Figure 20: Adjusting the horizon of the path planner shifts the lattice of samples (blue markers), the green marker represents the pure pursuit trackers selected lookahead distance.

Google Cartographer (Hess et al., 2016) for SLAM, a particle filter for pure localization (Walsh and Karaman, 2017), and a behavior controller which manages communication between the planning nodes and the motor controller.

TunerCar compiles optimizes versions of two processes on the vehicle: a geometric path tracking controller based on pure pursuit (Coulter, 1992; Snider, 2009) and a path planning module which utilizes a sampling-based non-linear model predictive control (Howard, 2009; McNaughton, 2011). In addition TunerCar supplies a reference raceline and adjusts the vehicle's physical configuration.

Path Tracker

The goal of path tracker is to compute steering inputs which allow the autonomous vehicle to follow a sequence of waypoints defined in the map frame. Performance of a path tracker is measured by the feasibility of the inputs, heading error, and lateral offset between the path and vehicle's position (Snider, 2009). We utilize the pure pursuit path tracker (Coulter, 1992), a geometric method which has been shown to be effective if properly tuned for the expected operating conditions. In particular, the lookahead distance, which is used to select a point on the path ahead of the vehicle significantly affects the performance. Too small of a lookahead and the vehicle oscillates; too large and corners are cut. Thus, the lookahead distance of the path tracker computed as $g_w d_l$ (where the tracking lookahead gain is $g_w < 1$ to ensure the tracking lookahead doesn't exceed the path planning horizon, d_l), is included in the search space as a tuneable parameter.

Path Planner

The goal of the path planner is to generate kinematically and dynamically feasible trajectories that can take the vehicle from its current pose to a sampled set of goal poses, see Figure 20. Each trajectory is represented as a set of cubic spirals, $p = [s, a, b, c, d]$ where s is the total arc length of the trajectory and (a, b, c, d) are equispaced knot points encoding the path curvature. For each trajectory, gradient descent is used to find spline parameters which minimize the error between the goal pose and the calculated end point given by a forward simulation of the vehicle dynamics. Real-time performance of the system is improved by creating a dense lookup table of pre-computed goal, solution pairs as described in (Urmson et al., 2008). Thus, online, once a goal point has been chosen, all that remains is to sample N equidistant points corresponding to the spline parameters stored in the lookup-table and check them against an occupancy grid for feasibility. In order to adapt the method to the operational domain the horizon of the planner is adjusted to account for track geometry.

6.6. Experiments

In our experiments we explore the assignment of hyperparameters of the CMA-ES algorithm as well as the effectiveness of the method in both simulation and reality. As noted in Section 6.3, CMA-ES requires only three hyperparameters: ϵ, the threshold of the L^2 norm of the covariance matrix for the sampling distribution, λ, the population size, and μ, the quantile of samples to retain between epochs. Figure 21 shows an experiment in which ϵ is determined. We repeat this experiment on multiple hyperparameters selections, observing that the lowest lap time does not decrease significantly after the L^2 norm reaches the determined threshold for all experiments. Figure 22 shows the performance of the CMA-ES population for a selection of quantiles μ and population sizes λ. We observe that the best final minimum lap time is achieved as λ increases and μ decreases. We omit results from some combinations of hyperparameters for the sake of clarity in Figures 21 and 22. Based on these experiments we select a μ of 0.01 and λ of 10000 for best overall performance in terms of lap time (here, convergence rate is less of a concern due to the offline nature of our approach).

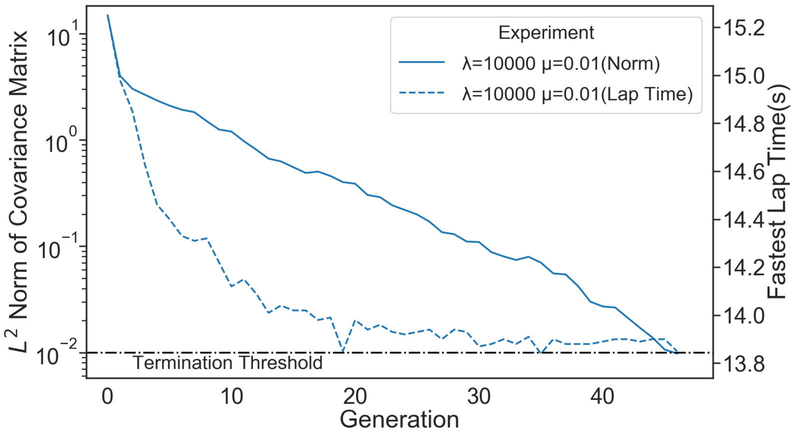


Figure 21: Convergence of L^2 norm of covariance matrix to determine the threshold ϵ. The termination threshold for the norm is reached when the optimization converges to a local minimum.

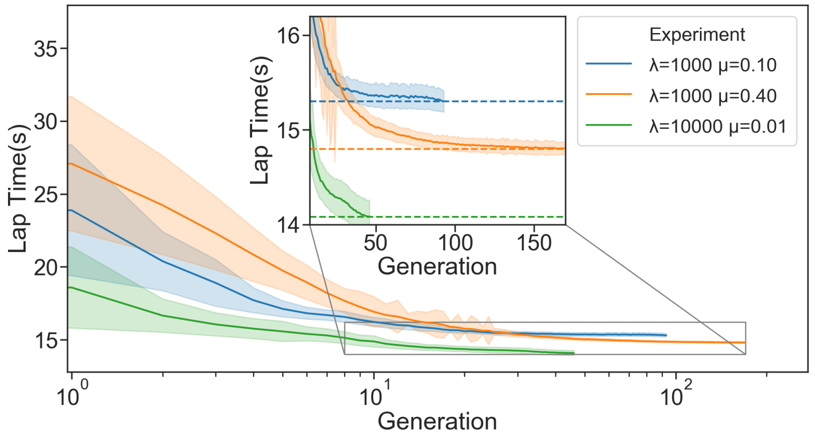


Figure 22: Effect of population size λ and quantile μ on performance. The combination of λ at 10000, and μ at 0.01 has the best performance in terms of the balance between best lap time and convergence rate.

To further evaluate the CMA-ES implementation in TunerCar, we compare the mean and variance of the top 100 lap times relative to a naive random sampling for fixed computational

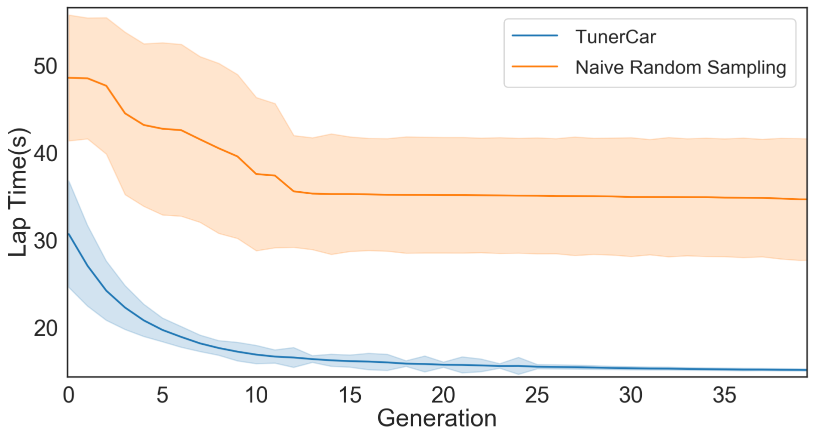


Figure 23: Performance on Philadelphia Track A with fixed computation.

budgets between 10,000 and 200,000 simulations. The results, shown in Figure 23 show that our methodology converges to a set of solutions with lap times significantly lower than that of naive random sampling.

We designed three sets of experiments to evaluate the effectiveness of our toolchain. First, we conducted experiments to validate the results of our approach using the hardware detailed in Section 6.5. These experiments are conducted on *Philadelphia Track A, B* which are challenging due to the 90 degree turns and narrow drivable surface areas (as seen in Fig. 24). The list of physical car parameters to tune was limited to an easily modifiable subset that avoided major structural changes to the car's components: total car mass and longitudinal location of the car's center of mass. After the pipeline has computed an optimal trajectory and set of parameters, we physically modified the car's parameters and deployed the tuned system, using the optimized waypoints and velocities.

Our second set of experiments was conducted in simulation and compared against historical racing times recorded in the environments. We chose to run the optimization process on a set of two tracks – *Porto* (Mangharam et al., 2018b) and *Torino* (Mangharam et al., 2018a) – where we have hosted competitive racing events in order to benchmark the performance of TunerCar relative to hand-tuned solutions. Due to the variety of modifications various entrants utilized (*e.g.* springs, suspension, geometry, and custom chassis), we do not restrict

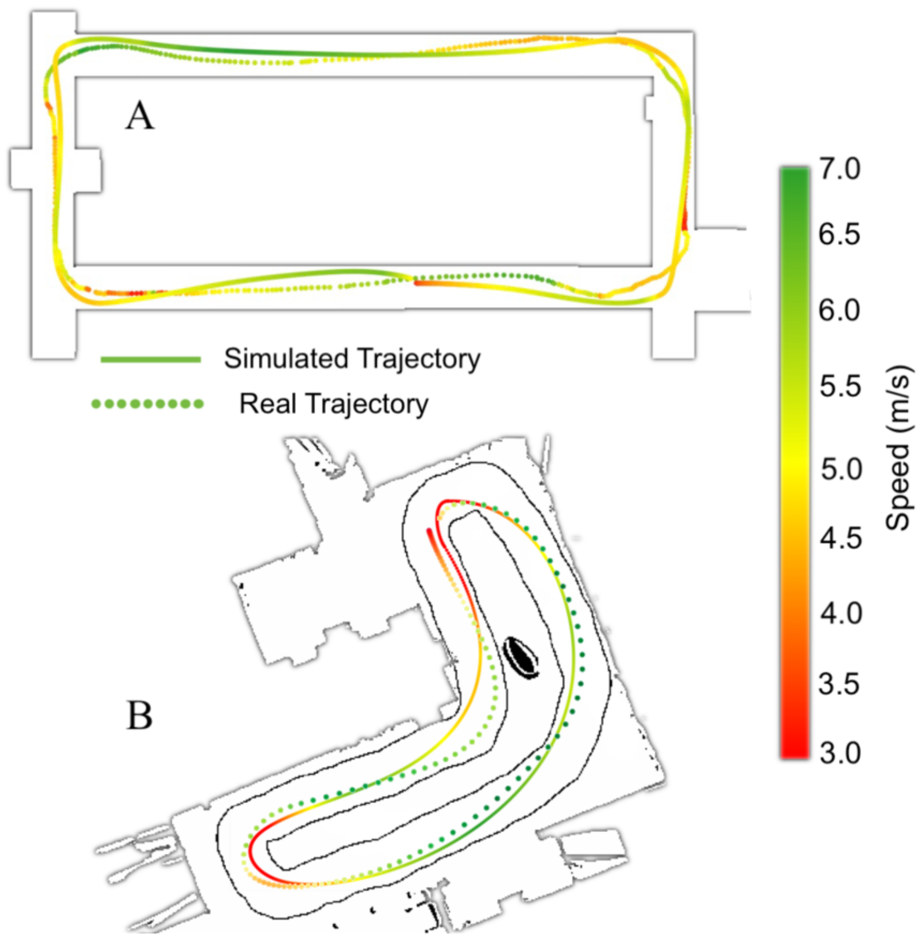


Figure 24: Comparison of simulated versus real performance: (A) Philadelphia Track A, (B) Philadelphia Track B

Table 9: TunerCar performance in simulation and reality

Philadelphia Track A		
Method	**Lap Time (s, sim)**	**Lap Time (s, real)**
TunerCar	16.2376 ± 0.2161	16.19
Pure Pursuit	17.2307 ± 0.0620	17.00
Expert Solutions	N/A	22.2291 ± 3.5958
Philadelphia Track B		
Method	**Lap Time (s, sim)**	**Lap Time (s, real)**
TunerCar	13.2424 ± 0.7782	14.03
Pure Pursuit	13.8256 ± 0.8848	16.80

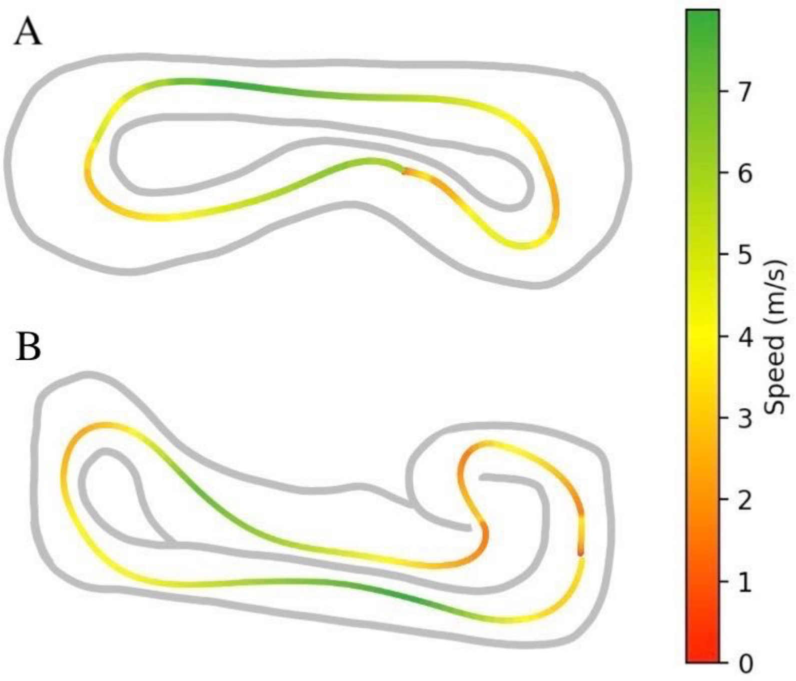


Figure 25: Agent solutions on the tracks: (A) Porto, (B) Torino

Table 10: TunerCar vs. expert-tuned agents in simulation

Method	**Lap Time (Porto, s)**	**Lap Time (Torino, s)**
TunerCar	8.11 ± 0.7372	17.96 ± 0.1214
Pure Pursuit	11.9 ± 0.8342	20.52 ± 0.1367
Best Expert Solution	9.37	19.19

the set of physical parameters.

Lastly, we evaluate the reality gap in light of the proposed online optimization strategy by comparing vehicle performance in simulation to that which was achievable in reality. Figure 25 and Table 10 details the solutions found by TunerCar which exceed expert tuning-performance on (8.A) *Torino* and (8.B) *Porto*. Figure 24 and Table 9 show a comparison of the relative performance between top solutions deployed on the car and in the simulator on (9.A) *Philadelphia Track A* and (9.B) *Philadelphia Track B* as well as a comparison to expert engineered controllers (both crowd-sourced and created by the authors)

6.7. Conclusions

We introduce TunerCar, a superoptimization toolchain for jointly optimizing racing strategy, vehicle parameters, planning methods, and control algorithms for an autonomous racecar. We detail the target hardware, software, simulators, and systems infrastructure necessary for implementation. Our methodology deploys a modern evolution strategy onto a massively parallelized software stack that enables simulations to proceed 6x times faster than real-world rollouts, achieving up to 21% faster lap times compared to expert solutions on real world race tracks. We validate our solution on an AV software stack and 1/10th-scale open-source vehicle developed in Chapter 5.

While the method as described in this chapter has not been rigorously tested in head-to-head racing, the online path-planning component is capable of navigating the vehicle in the presence of other agents. Chapter 7 investigates the feasibility of deploying this methodology utilizing an expanded search space that encodes the behaviors of other racers. A clear limitation of the presentation of this work is a lack of comparisons to other black-box optimizations, we mitigate this deficiency by measuring the toolchains performance against a variety of hand-tuned solutions entered in F1TENTH competitions. Given that the experimental results show that we can find significant performance improvements over the other competition entries, it is important to explore whether other search strategies can create even faster agents.

CHAPTER 7 : FormulaZero

7.1. Chapter Overview

Elements of this chapter have been adapted from "FormulaZero: Distributionally Robust Online Adaptation via Offline Population Synthesis," which appeared in the Proceedings of the International Conference on Machine Learning 2020 as joint work with Aman Sinha, Hongrui Zheng, Rahul Mangharam, John Duchi, and Russ Tedrake.

7.2. Introduction

Current 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. The strategic implications of this tradeoff are magnified in racing. During the 2019 Formula One season, the race-winner achieved the fastest lap in only 33% of events (Federation Internationale de l'Automobile, 2019). Empirically, the weak correlation between achieving the fastest lap-time and winning suggests that consistent and robust performance is critical to success. In this paper, we investigate this intuition 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, especially in robotics applications. 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 through an *ambiguity* set $\mathcal{P}$ for the state-action transitions. Then the AV's objective is

$$\text{maximize} \inf_{P_{sa} \in \mathcal{P}} \sum_t \lambda^t \mathbb{E}[r(o(t))]. \tag{7.1}$$

The obvious price of robustness (Bertsimas and Sim, 2004) 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. The name of our approach—FormulaZero—alludes both to the Formula One racing league and the fact that we use self-play (and no demonstrations) to learn competitive behaviors, similar to the approach of AlphaZero (Silver et al., 2018).

Section 7.2.1 gives context to our learning problem, including connections to classical control techniques. In Section 7.3, 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 7.4, 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 7.6 details the practical implications of the theoretical results, emergent properties of the method, and the experimental performance of our approach.

7.2.1. Related work

Reinforcement learning (RL) has achieved unprecedented success on classic two-player games (*e.g.* Silver et al., 2018), leading to new approaches in partially-observable games with continuous action spaces (Arulkumaran et al., 2019; Berner et al., 2019). In these works, agents train via self-play using Monte Carlo tree search (Browne et al., 2012; Sutton and Barto, 2018) or population-based methods (Jaderberg et al., 2017, 2019). The agents optimize expected performance rather than adapt to individual variations in opponent strategy, which can lead to poor performance against particular opponents (Bansal et al., 2017). 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 (Nilim and El Ghaoui, 2005) or a POMDP (Kaelbling et al., 1998). Early results Bagnell et al. (2001) and Nilim and El Ghaoui (2005) describe solutions for robust planning in (PO)MDPs with tabular state/action spaces. Equivalent results in control are analytical formulations applicable to uncertainty in linear time-invariant systems (Doyle et al., 1988; Vinnicombe, 1993; Zhou et al., 1996). Recent works (Tamar et al., 2014; Pinto et al., 2017; Mandlekar et al., 2017; Gleave et al., 2019) 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.

Our approach is designed to adjust the agent's evaluation of short-term plans relative to uncertainty in the opponent's behavior rather than provide worst-case guarantees. Complementary to and compatible with our approach are techniques which provide the latter guarantees, such as robust model predictive control (Bemporad and Morari, 1999). Extensions of robust control for nonlinear systems and complex uncertainty models are also compatible (*e.g.* Majumdar and Tedrake (2013); Althoff and Dolan (2014); Gao et al. (2014)). In contrast to formal approaches which explicitly guarantee robustness, some authors have proposed multitask or meta-learning approaches (*e.g.* Caruana, 1997; He et al., 2016; Finn et al., 2018) can implicitly learn to play against multiple opponents. However, such techniques do not explicitly model uncertainty or quantify robustness, which we deem necessary in the high-risk, safety-critical regime.

Planning in belief space is closely related to our approach and is well-studied in robotics (see *e.g.* Kochenderfer, 2015). Specifically in the AV domain, Galceran et al. (2015) and Ding and Shen (2019) 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. (2011) 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 (Ben-Tal et al., 2013; Namkoong and Duchi, 2017). These methods reweight training data to reduce the variance of the training loss (Namkoong and Duchi, 2017). While others apply DRO to episodic RL for training *offline* (Sinha et al., 2017; Smirnova et al., 2019), we reweight the belief *online*.

Online methods for control fall under the umbrella of adaptive control (Kumar, 1985; Åström and Wittenmark, 2013). Dean et al. (2018) and Agarwal et al. (2019) establish regret bounds for adaptive control methods applied to LTI systems, tightening the relationship to online learning. Due to the more general nature of our problem, we draw from the adversarial multi-armed bandit framework of online learning (Abernethy and Rakhlin, 2009; Bubeck et al., 2012; Shalev-Shwartz et al., 2012).

Our belief state corresponds to a categorical distribution of polices governing an opponent's next action; the goal is to predict which strategy the opponent is using and compute the best response. This 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 (Liniger and Lygeros, 2019; Wang et al., 2019b); or, without this simplifying assumption, that uses demonstrations to learn possible sets of policies (Sadigh et al., 2016; Williams et al., 2017). In contrast, we learn the set of policies without demonstrations and use DRO to robustly score the AV's plans.

We convert the problem of predicting opponent behavior in a continuous action space into an adversarial bandit problem by learning a set of cost functions that characterize a discrete set of policies. As a result, we would like the opponent models to be both near-optimal and diverse. We use determinantal point processes (DPPs) (Kulesza et al., 2012) to sample diverse configurations of the parameter space. However, first we must learn a DPP kernel, which requires that we efficiently sample *competitive* cost functions from the larger configuration space. Since we assume no structure to the set of competitive cost functions, we employ a Markov-chain Monte Carlo (MCMC) method. Complementary methods include variational-inference (*e.g.* Arenz et al. (2018)) and evolutionary (*e.g.* Mouret and Clune (2015)) approaches, which can be challenging to scale up to unstructured, high-dimensional settings of which we have little prior domain knowledge. In our approach, we update the classic simulated tempering method (Marinari and Parisi, 1992) with a novel annealing scheme (Kirkpatrick et al., 1983; Černỳ, 1985) designed for population diversity. We describe this approach next.

7.3. 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 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 (Matyas, 1965; Hastings, 1970; Kirkpatrick et al., 1983; Černý, 1985; Ingber, 1993; Hu and Hu, 2011). 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 (Bergstra and Bengio, 2012; Jaderberg et al., 2017) 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 (Smith, 1984; Bélisle et al., 1993; Lovász, 1999).

7.3.1. AAdaPT

AAdaPT builds upon replica-exchange MCMC, also called parallel tempering, which is a standard approach to maintaining a population of configurations (Swendsen and Wang, 1986; Geyer, 1991). 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 26). Horizontal proposals consist of swapping two configurations in adjacent temperature levels uniformly at random; the proposal is accepted using standard Metropolis-Hastings (MH) criteria (Hastings, 1970). 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 (Geyer, 1991). 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 4), 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 D.1). 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

Algorithm 4 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 (7.2))
  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 D.1)
```

proposals (Smith, 1984; Bélisle et al., 1993; Lovász, 1999) 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 (Lovász, 1999; Lovász and Vempala, 2003, 2006). Note that in our implementation the MCMC steps are only performed on x, while θ updates occur via SGD (see below).

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

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

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

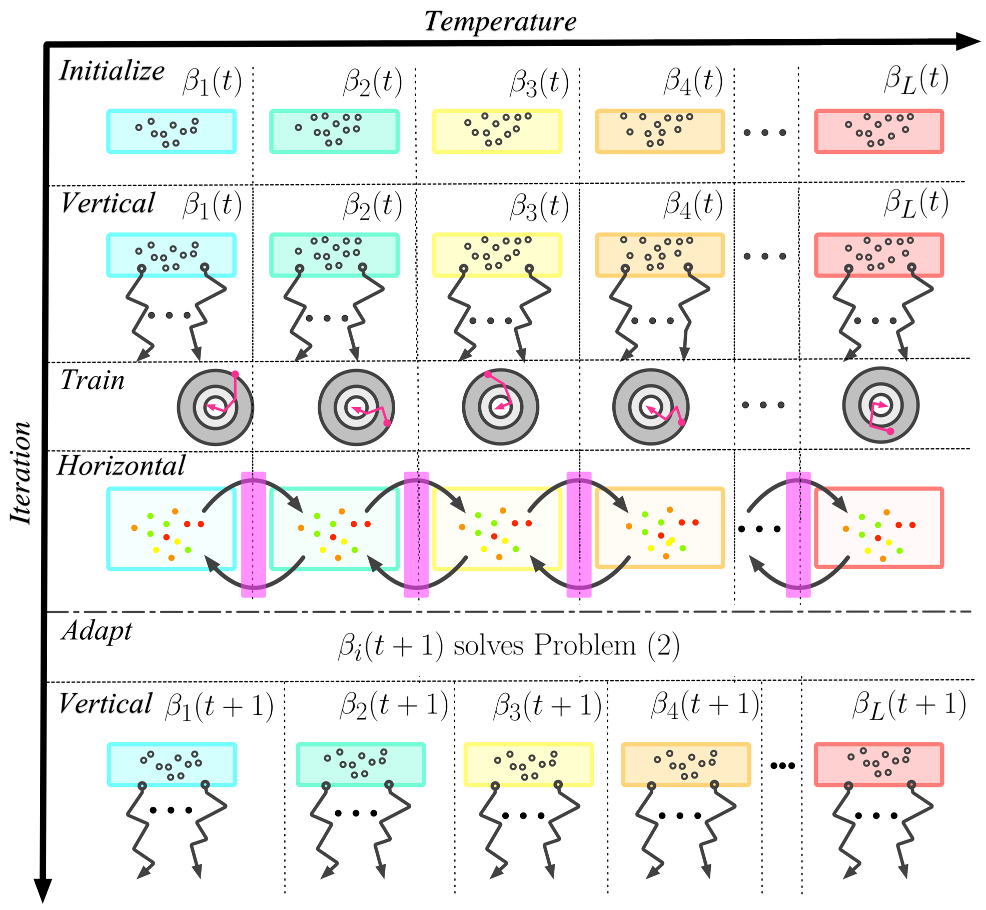


Figure 26: Illustration of AAdaPT. Vertical MCMC steps (jagged black arrows) occur in parallel for $x^{i,j}$, followed by gradient descent for trainable parameters $\theta^{i,j}$ (magenta arrows) and horizontal MCMC configuration swaps between populations (curved black arrows). Temperatures $\beta_i(t)$ are then updated by problem (7.2).

a given β_i, we set β_{i-1} by solving the following convex optimization problem:

$$\underset{\{\beta_{i-1}\geq\beta_i,\quad p_{i-1,i}\leq\alpha^{\frac{t}{(L-1)}}\}}{\text{minimize}} \qquad \beta_{i-1}, \tag{7.2}$$

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 (7.2).

Evaluating proposals via self-play: We apply AAdaPT to a multi-agent game. It is only possible to evaluate $f(x, \theta)$ in the context of other agents, but we consider the setting where demonstrations from potential opponents are either difficult to obtain or held secret. Thus, we iteratively evaluate f via self-play. For each configuration (x, θ), we perform a race in the simulated environment between two vehicles with the same policy (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

$$H_{ab} = \|x^{1,a} - x^{1,b}\|. \tag{7.3}$$

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 $d \le D$ configurations from this DPP.

7.4. 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 7.3). For a given goal state, we compute the parameters of a cubic spline that reaches the goal by solving a nonconvex trajectory optimization problem (McNaughton, 2011); 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 7.3), similar to Sadat et al. (2019) (see Appendix 7.5 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 prototype 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 $P_0(t) :=$ 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 $D_f(P\|Q) = \int \phi(\frac{dP}{dQ})dQ$. We use sets $\mathcal{P}(t) := \{Q : D_f(Q\|P_0)(t) \leq \rho\}$ 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 (7.1) 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

$$\sup_{Q\in\mathcal{P}(t)} \mathbb{E}_Q[C(t;p)] = \sup_{q:\sum_i w_i\phi(\frac{q_i}{w_i})\leq\rho} \sum_i q_i c_i(t;p). \tag{7.4}$$

When $\rho = 0$, this is the expected cost under P_0; larger ρ adds robustness. Solving the convex optimization problem (7.4) first requires computing the costs $c_i(t)$. Using $\lambda \geq 0$ for the constraint $D_f\left(Q \| P_0\right) \leq \rho$, a partial Lagrangian is

$$\mathcal{L}(q, \lambda)=\sum_i q_i c_i(t)-\lambda\left(\sum_i w_i \phi\left(q_i / w_i\right)-\rho\right).$$

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 (7.4). Maximizing $\mathcal{L}(q, \lambda)$ with respect to q for a given λ requires $O(d)$ time using a variant of median-based search (Duchi et al., 2008) (see Appendix D.2). Thus, computing an ϵ-suboptimal solution uses $O(d \log(1/\epsilon))$ time.

The supremum in the robust cost (7.4) 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 (7.4) 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.

7.4.1. Approximating the robust cost

For a large library of prototypical opponents (large d), computing every c_i in the objective (7.4) 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 $\mathcal{P}_{N_w} := \{q :$

$D_f\left(q \| \mathbf{1}/N_w\right) \le \rho\}$ to $\mathcal{P}$ and solve the following empirical version of problem (7.4):

$$\underset{q \in \mathcal{P}_{N_w}}{\text{maximize}} \quad \sum_k q_k c_{j_k}(t; p). \tag{7.5}$$

This optimization problem (7.5) makes manifest the price of robustness in two ways. The first involves 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. (2003) 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.

The second way we illustrate the price of robustness is by quantifying the quality of the approximation (7.5) 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

$$R(Q; p) := \mathbb{E}_Q[C(t; p)], \quad \hat{R}(q; p) := \frac{1}{N_w} \sum_{k=1}^{N_w} q_k c_{j_k}(t; p).$$

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 Appendix D.2 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 (7.4), ϵ-optimal solutions require $O(N_u \log(1/\epsilon))$ time for $N_u \leq N_w$ unique indices j_k. In our experiments (*cf.* Section 7.6), most of the computation time involves the setup to compute the N_u costs c_{j_k}.

7.4.2. 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 $o_{\text{opp}}^H(t) := \{o_{\text{opp}}(t), o_{\text{opp}}(t-1), ..., o_{\text{opp}}(t-h+1)\}$ over a horizon h, we define the likelihood of this sequence coming from the i^{th} prototype opponent as

$$l_i^h(t) = \log d\mathbb{P}\left(o_{\text{opp}}^h(t) | G(\theta^{1,i})\right), \tag{7.6}$$

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 $l_i^h(t)$, we define the losses $L_i(t) := 1 - l_i^h(t)/\bar{l}$.

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 (Shalev-Shwartz et al., 2012). In a resource-constrained setting, we can only select a few of these losses, so we use EXP3 (Auer et al., 2002) 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 7.4.1). Denote our unbiased subgradient estimate as $\gamma(t)$:

$$\gamma_i(t) = \frac{1}{N_w} \sum_{k=1}^{N_w} \frac{L_i(t)}{w_i(t)} \mathbf{1}\{J_k = i\}. \tag{7.7}$$

Algorithm 5 describes our slightly modified EXP3 algorithm, which has the following ex-

Algorithm 5 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 (7.7))
    $w_i(t+1) := \frac{w_i(t)\exp(-\eta_t \gamma_i(t))}{\sum_{j=1}^{d} w_j(t)\exp(-\eta_t \gamma_j(t))}$
```

pected regret.

Proposition 3. *Let $z := \frac{d-1}{N_w} + 1$. Algorithm 5 run for T iterations with stepsize $\eta = \sqrt{\frac{2\log(d)}{zT}}$ has expected regret bounded by $\sum_{t=1}^{T} \mathbb{E}\left[\gamma(t)^T(w(t) - w^\star)\right] \leq \sqrt{2zT\log(d)}$.*

See Appendix D.2 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 (Uchiya et al., 2010; Zhou and Tomlin, 2018). We prefer our approach for its simplicity and ability to be easily combined with the robust-cost computation.

7.5. Agent Design

This section gives describes the design of autonomous racing agents which we use to demonstrate population synthesis and online adaptation. Figure 27 gives a graphical overview. Online, each agent measures the world using onboard sensors such as a planar LIDAR. Using 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, a trajectory is generated for each. The set of trajectories are then evaluated within our DRO framework, the best goal is chosen, and a new control command is applied to the vehicle. In what follows we describe each of these components in detail.

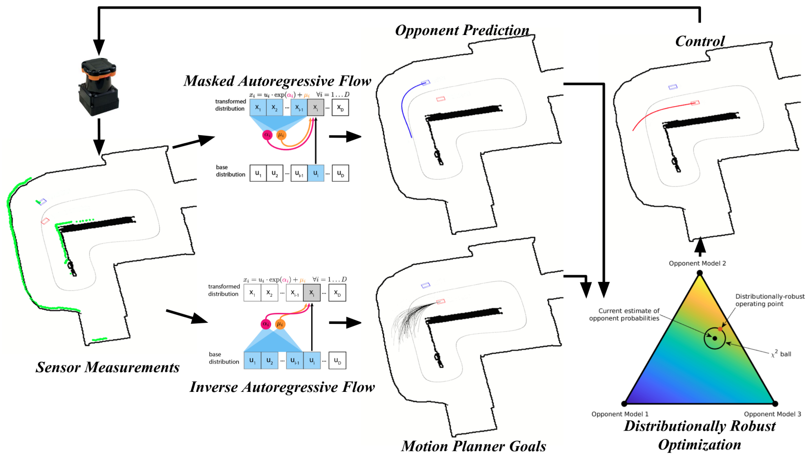

Opponent Prediction
Control
Masked Autoregressive Flow
Sensor Measurements
Inverse Autoregressive Flow
Motion Planner Goals
Opponent Model 2
Current estimate of opponent probabilities
Distributionally-robust operating point
χ^2 ball
Opponent Model 1
Opponent Model 3
Distributionally Robust Optimization

Figure 27: FormulaZero agent design

7.5.1. Sampling behavior proposals

The agent software uses a hierarchical planner similar to Ferguson et al. (2008). One key difference is the use of neural autoregressive flows. Whereas Ferguson et al. (2008) use a deterministic lattice of points to sample future motions, our vehicle draws samples from a neural autoregressive flow. 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) (Kingma et al., 2016). 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 the overview of the agent design, masked (Papamakarios et al., 2017) 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.

Each population member has a dedicated IAF model, which is trained iteratively according to the AAdaPT algorithm described in Section 7.3 using the hyperparameters given in Section 7.6. 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.

7.5.2. Trajectory generation

Given a set of goals sampled from the IAF, the trajectory generator is used to compute kinematically and dynamically feasible trajectories. The trajectory generator combines approaches from (Howard, 2009; Nagy and Kelly, 2001; Kelly and Nagy, 2003; McNaughton, 2011). 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 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 postive y

Table 11: 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

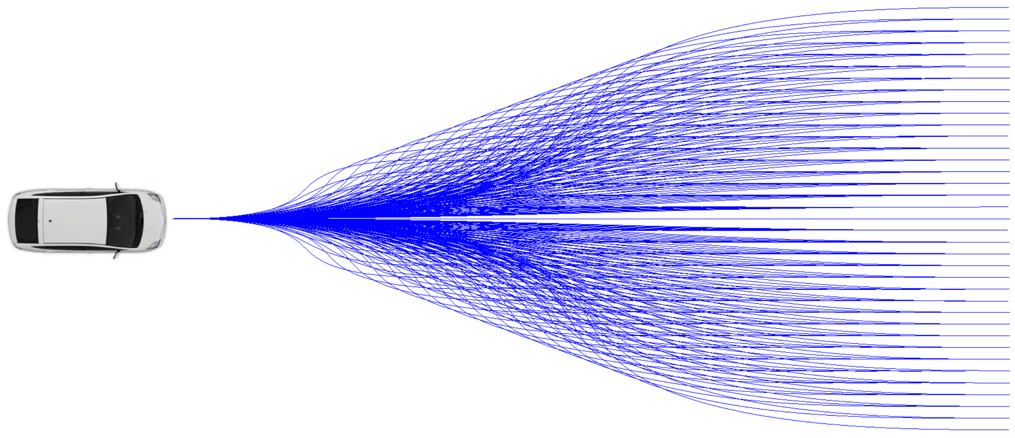


Figure 28: Sample trajectories from the look-up table

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 11. Figure 28 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.

7.5.3. Trajectory Cost Functions

Each of the generated trajectories is evaluated as 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.

7.5.4. 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 geometric tracking method called pure pursuit (Coulter, 1992; Snider, 2009). Due to the decoupling of the trajectory generation and tracking modules it is possible for the tracker to run at a much

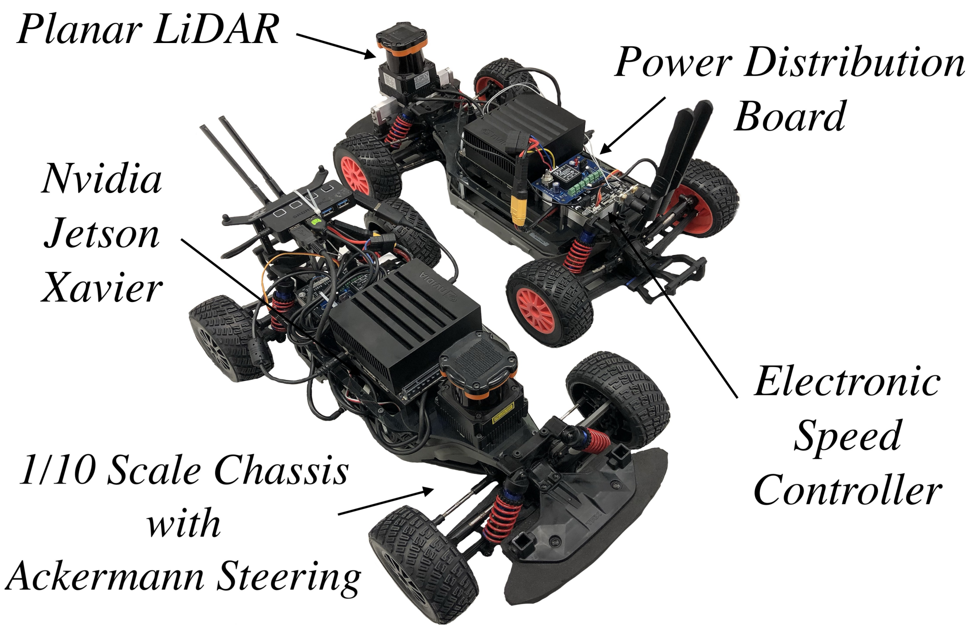

Planar LiDAR
Power Distribution Board
Nvidia Jetson Xavier
Electronic Speed Controller
1/10 Scale Chassis with Ackermann Steering

Figure 29: Components of the 1/10-scale vehicle

higher frequency than the trajectory generator; this is essential for good performance.

7.6. 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 7.3 and 7.4, 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.

The experiments use an existing low-cost $1/10^{th}$-scale, Ackermann-steered AV (Figure 29). Additionally, we create a simulator and an associated OpenAI Gym API (Brockman et al., 2016) 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 Figure 31).

7.6.1. 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 configurations 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 that "tunneling" from the highest-temperature level to the coldest takes $O(L^2)$ accepted steps). Finally, for training θ, we use Adam (Kingma and Ba, 2014) with a learning rate of 10^{-4}.

Figure 30 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 30(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 30(b) illustrates a metric for measuring diversity, the Frobenius norm of the Mahalanobis distance matrix (7.3). 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 31 illustrates qualitative differences between cost functions. Figure 31(a) displays trajectories for agents driven using 5 cost functions sampled from the learned DPP. The cornering behavior is quite different between the trajectories. Figure 31(b) displays 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.

7.6.2. 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 against each of the $d = 10$ diverse opponents sampled from the DPP. For fair

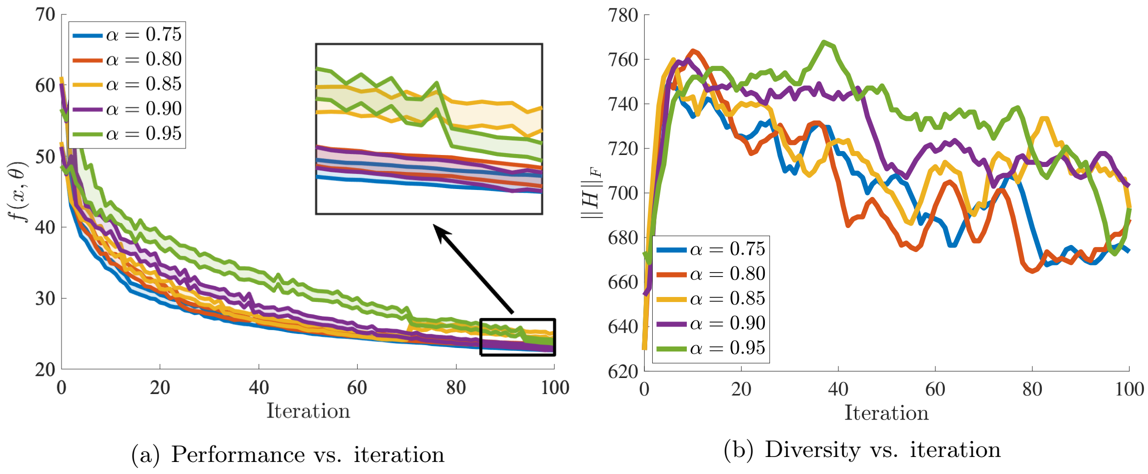
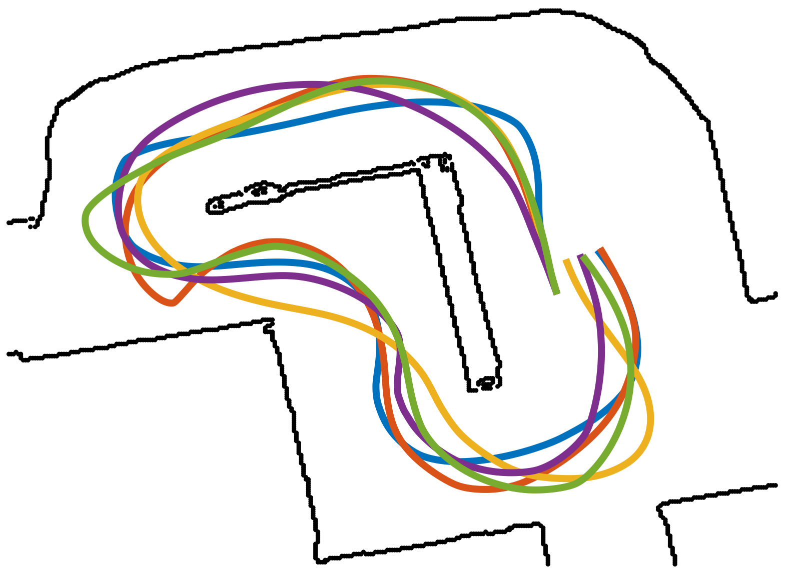


Figure 30: Hyperparameter selection for AAdaPT. (a) 95%-confidence intervals for $f(x, \theta)$ in the coldest temperature level. (b) Frobenius norm of the Mahalanobis distance matrix H (7.3). The value $\alpha = 0.9$ achieves the best performance and diversity.

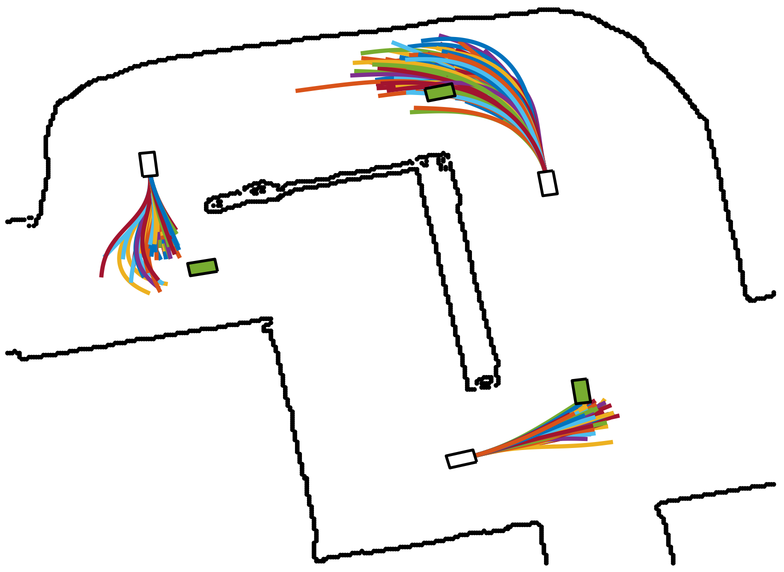
comparisons, half of the races have the opponent starting on the outside and the other half with the opponent on the inside of the track. Importantly, these experiments involve only the most elite policies from the temperature level $\beta_1(T)$. Since the physical characteristics of the vehicles are 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.

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 D.5.5). Smaller iTTC values imply more dangerous scenarios and more aggressive policies. In Table 12, 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 32(a) shows that Algorithm 5 identifies the opponent vehicle within approx-



(a) Rollouts from 5 agents



(b) Snapshot trajectories

Figure 31: Qualitative illustrations of multimodal behavior in the learned population of cost functions

imately 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 13 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

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

Table 12: The effect of distributional robustness 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 13: The effect of adaptivity on win-rate

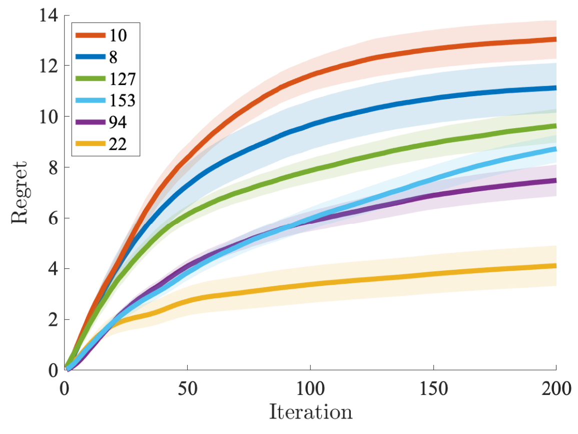
Agent	Win-rate Non-adaptive	Win-rate Adaptive	p-value
$\rho/N_w = 0.001$	0.593± 0.025	0.588± 0.025	0.84
$\rho/N_w = 0.025$	0.593± 0.025	0.600± 0.024	0.77
$\rho/N_w = 0.2$	0.538± 0.025	0.588± 0.025	0.045
$\rho/N_w = 0.4$	0.503± 0.025	0.573± 0.025	0.0098
$\rho/N_w = 0.75$	0.513± 0.025	0.593± 0.025	0.0013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

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 (7.5). 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.

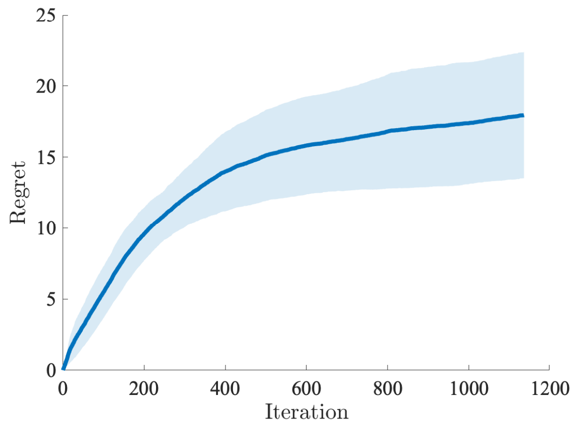
7.6.3. Real-world validation

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 32(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 simu-

[2] `https://youtu.be/7Yat9FZzE4g`



(a) Simulation



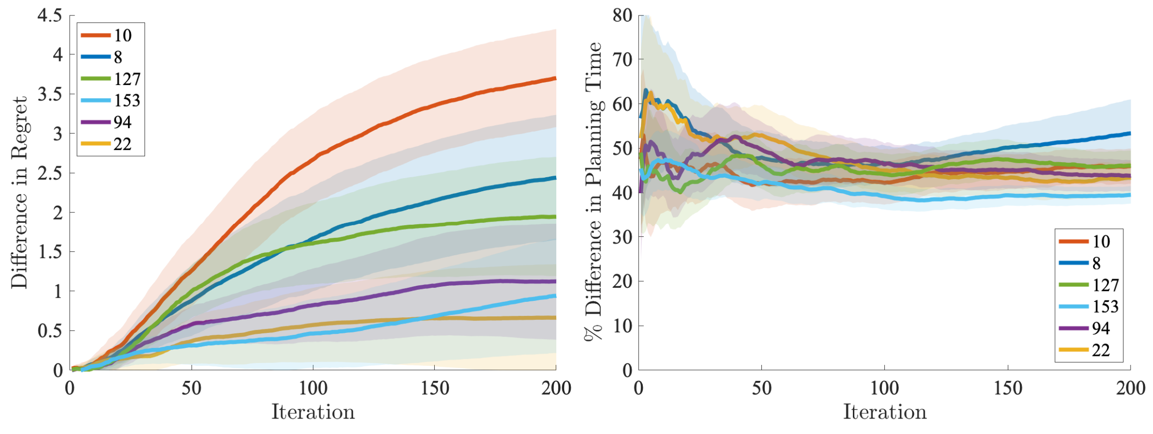
(b) Real

Figure 32: 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.

lated 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 32(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 D.5).

7.6.4. 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 7.4.1) and perform a bandit-style update to the ambiguity set (Section 7.4.2). Now we analyze the time-accuracy tradeoff of performing this sampling approximation rather than using all d prototypical opponents at every time step. Figure 33(a) shows the difference in regret for the same experiments as in Figure 32(a) if we perform full online mirror-descent updates. Denoting the simulations in Figure 32(a) as S and those with the full mirror descent



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

Figure 33: 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.

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 (7.7). Similarly, Figure 33(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 constant 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.

7.6.5. 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 D.5.6 for details on each agent's policy). We perform only simulated experiments, as we are unable to perform further real-world experimentation at the time of writing 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,

Table 14: 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 15: 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

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 14 and 15 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.

7.7. 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.

Part III

Outlook

CHAPTER 8 : Conclusions and Future Work

8.1. Summary

In Part I of this thesis we consider the development of methods for evaluating the performance of AVs. We begin by exploring different paradigms for AV evaluation: formal verification, falsification, and estimation. In Chapter 2 we demonstrate how verification and falsification methods may be applied to AV systems. Through several case studies we highlight why it is necessary to evaluate the risk of failures rather than prove or disprove the existence of failures and propose desiderata of a statistical framework for model checking based on (accelerated) Monte Carlo methods.

In Chapter 3 we develop our risk-based evaluation framework and show, by example, a toolchain which is capable of efficiently evaluating AV systems even when failures are rare. The toolchain includes a method for learning a generative model of the operational domain; a particular challenge is how we represent the actions of other vehicles and dynamic agents. In order to solve this problem we propose developing a population of agents, each trained via generative adversarial imitation learning. The toolchain also includes an implementation of the cross-entropy method for adaptive importance sampling, a collection of simulators, and two ego-vehicle policies. Our experiments show that we accelerate the assessment of rare-event probabilities with respect to naive Monte Carlo methods as well as real-world testing.

In Chapter 4 we improve the risk-based framework by proposing a novel method for accelerated AV evaluation. As in previous chapters we compare the performance of sampling methods by evaluating the variance of the risk estimate under a fixed computation budget. Our proposed method, gradient-guided bridge sampling, employs three concepts—exploration, exploitation, and optimization—in order to evaluate system safety with provable statistical and computational efficiency. We demonstrate the performance of our method on a variety of reinforcement-learning and robotic systems, including an AV system that has driven over

20 million miles on public roads, highlighting its use as a tool for continuous integration and rapid engineering design.

In Part II of this thesis we consider the problem of adapting AV components to new operational domains. Chapter 5 describes an exemplary system which we use to conduct our investigations: a $1/10^{th}$-scale autonomous racing car which includes a hardware design, a simulator, benchmark tasks, and baseline task solutions. The benchmark tasks form the basis for the research questions addressed in the following chapters. The hardware and simulator designs enable in-depth experiments.

In Chapter 6 we explore a simple version of the domain adaptation problem. We are given a physical robot with some adjustable parameters related to the suspension, tires, and weight distribution, a set of constraints defining the map on which the vehicle will be deployed, and a set of planning and control components whose parameters can be adjusted. The goal is to setup the vehicle to achieve maximum performance within the new environment. In the single vehicle racing setting that we consider the performance criteria is the laptime, where lower times are better. As in Part I we compare methods for finding good vehicle configurations by assigning each method equivalent computation budgets and comparing the quality of the best solutions found. We concluded the chapter by demonstrating the use of stochastic search methods on a variety of tracks and performing a real-world experiments to validate the proposed solution.

In Chapter 7 we consider a more challenging instance of the domain adaptation problem: multi-agent racing. 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.

8.2. Future Work

The success of machine learning in a variety of domains has led to new paradigms of software development; the growing divide in methodologies has been categorized as “Software 1.0/2.0”. Software 1.0 involves the explicit design of programs aided by layers of abstraction and modularity. Generally, an engineer is given a formal specification which describes the correct behavior of the program, and implements algorithms and data structures that satisfy the specification. In contrast, Software 2.0 is characterized by the standardization of parametric program structures– *e.g.* ResNet or Transformer neural network architectures or any other program with explicitly identified trainable parameters. This standardization is possible due to the development of efficient methods to fit the parameters to tasks specified by input-output data. In turn, there has been a shift in engineering focus to the curation and acquisition of data, the development of methods which adjust the parameters of the program in order to perform specific tasks, and the evaluation of model performance.

The problem of data acquisition and curation presents several challenges: which data to select and how to process it, how to actively find useful data, and what to do when the available data is known to insufficiently cover the space of possible inputs. In the Software 2.0 paradigm, data is central both to creating software via training and to evaluating the performance of the derived program. Conceptually, the selection and augmentation of data should then be informed, continuously, by its effect on the performance of training and evaluation. In doing so a virtuous cycle can be instantiated. As an AV accumulates miles and expands its operating domain based on the confidence derived from evaluation of past experience, new data can be acquired and used to repair model deficiencies.

The new data will contain both elements which are similar to prior experience as well as new information. The novel information expands the model of the operating domain and reveals *unknown unknowns.* The remaining data serves to calibrate and reify our understanding of *known unknowns.* In turn the AV software can be retrained, and more accurate evaluations in the expanded operating domain are made possible, restarting the cycle. While practical engineering and development work will need to be undertaken in order to build an efficient pipeline, there are also two primary research barriers to implementing the virtuous cycle concept.

First, the cycle's efficiency and safety (two desiderata that are unfortunately negatively correlated) is limited by what is observed during the data collection process. It is likely that the AV could be improved more rapidly if the cycle had greater access to the long tail of rare scenarios, but capturing this information in the wild may require operating the vehicle in conditions for which safety is poorly understood. Safe exploration techniques developed in the reinforcement learning community may offer some guidance in balancing these risks. Alternatively, suppose, that a known unknown is identified– *e.g.* heavy snowfall in San Francisco– a plausible, but unlikely occurrence for which no examples exist from which to confidently estimate the effect on performance. We could, instead, synthesize simulated data by specifying desirable characteristics; notably, actual observations of the event need not be part of this procedure. However, in order to use this synthetic data in an evaluation framework, it is also be necessary to estimate the likelihood of sampling the data in the wild. Thus, in the future, further efforts should be made to understand how we can principally add generative components which have not been observed in the real world.

Second, today, the most powerful tools for Software 2.0 are based on the application of stochastic gradient descent to differentiable parametric programs like neural networks. Unlike Software 1.0, when an error is found in the system, it is challenging to excise the behavior without inducing distribution shifts which may reduce performance or expose the vehicle to new untested scenarios. Thus, we desire new methodologies to ensure that policy updates induce predictable, Pareto-improvements of the system. How such methods will perform when signals from lagging measures such as catastrophic failures is difficult to obtain will also be a primary concern.

8.3. Conclusions

This thesis presents both foundational theory and practical methods for efficiently evaluating and adapting the performance of safety-critical autonomous systems. By definition, such systems can cause injury or death if they malfunction (Bowen and Stavridou, 1993). Thus, improving the tools that practitioners have to perform risk-estimation and adaptation has the potential to provide a strong positive impact. In the case of autonomous vehicles, Sparrow and Howard (2017) argue that it will be morally wrong not to deploy self-driving technology once performance exceeds human capabilities. Our work is an important tool for determining when this performance threshold has been achieved and improving today's technology to meet that threshold.

More broadly, the advent of autonomy could spark significant societal changes. While the widespread availability of autonomy-enabled devices could benefit public health, there are many external risks to privacy, fairness, and safety associated with their development. Thus, Benkler (2019) highlights that there is a growing need for the academic community to take action on defining the performance criteria to which AI applications will be held. Brundage et al. (2020) and Wing (2020) outline broad research agendas around the ethical development and use of AI which are necessarily interdisciplinary. Importantly, the methods presented in this thesis enable the comparison of autonomous systems in a common language—risk—across the spectrum from engineers to regulators and the public. Still, much more work needs to be done to empower researchers to influence policy. These efforts will require systemic initiatives by research institutions and organizations to engage with local, national, and international governing bodies.

APPENDIX: Risk-based Framework

A.1. Scenario specification

A scenario specification consists of a scenario description and outputs both p_γ (3.1), the accident rate, and a dataset consisting of initial conditions and the minimum time to collision, our continuous objective safety measure. Concretely, a scenario description includes

- a set of possible initial conditions, e.g. a range of velocities and poses for each agent
- a safety measure specification for the ego agent,
- a generative model of environment policies, an ego vehicle model,
- a world geometry model, *e.g.*a textured mesh of the static scene in which the scenario is to take place.

Given the scenario description, the search module creates physics and rendering engine worker instances, and Algorithm 1 then adaptively searches through many perturbations of conditions in the scenario, which we call scenario realizations. A set of scenario realizations may be mapped to multiple physics, rendering, and agent instantiations, evaluated in parallel, and reduced by a sink node which reports a measure of each scenarios performance relative to the specification.

In our implementation the safety measure is minimum time-to-collision (TTC). TTC is defined as the time it would take for two vehicles to intercept one another given that they each maintain their current heading and velocity (Vogel, 2003). The TTC between the ego-vehicle and vehicle i is given by

$$TTC_i(t) = \frac{r_i(t)}{[-\dot{r}_i(t)]_+}, \tag{A.1}$$

where r_i is the distance between the ego vehicle and vehicle i, and $\dot{r}_i$ the time derivative of this distance (which is simply computed by projecting the relative velocity of vehicle i onto the vector between the vehicles' poses). The operator $[\cdot]_+$ is defined as $[x]_+ := \max(x, 0)$. We define $TTC_i(t) = \infty$ for $\dot{r}_i(t) \geq 0$.

In this paper, vehicles are described as oriented rectangles in the 2D plane. Since we are interested in the time it would take for the ego-vehicle to intersect the polygonal boundary of another vehicle on the road, we utilize a finite set of range and range measurements in order to approximate the TTC metric. For a given configuration of vehicles, we compute N uniformly spaced angles $\theta_1, \dots, \theta_N$ in the range $[0, 2\pi]$ with respect to the ego vehicle's orientation and cast rays outward from the center of the ego vehicle. For each direction we compute the distance which a ray could travel before intersecting one of the M other vehicles in the environment. These form N range measurements $s_1, \dots, s_N$. Further, for each ray s_i, we determine which vehicle (if any) that ray hit; projecting the relative velocity of this vehicle with respect to ego vehicle gives the range-rate measurement $\dot{s}_i$. Finally, we approximate the minimum TTC for a given simulation rollout X of length T discrete time steps by:

$$f(X) := \min_{t=0,\dots,T} \left(\min_{i=1,\dots,N} \frac{s_i(t)}{[-\dot{s}_i(t)]_+} \right),$$

where we again define the approximate instantaneous TTC as ∞ for $\dot{s}_i(t) \geq 0$. Note that this measure can approximate the true TTC arbitrarily well via choice of N and the discretization of time used by the simulator. Furthermore, note that our definition of TTC is with respect to the *center* of the ego vehicle touching the *boundary* of another vehicle. Crashing, on the other hand, is defined in our simulation as the intersection of boundaries of two vehicles. Thus, TTC values we evaluate in our simulation are nonzero even during crashes, since the center of the ego vehicle has not yet collided with the boundary of another vehicle.

A.2. Network architectures

The MGAIL generator model we use takes the same inputs as that of Kuefler et al. (2017)—the dynamical states of the vehicle as well as virtual LIDAR beam reflections. Specifically,

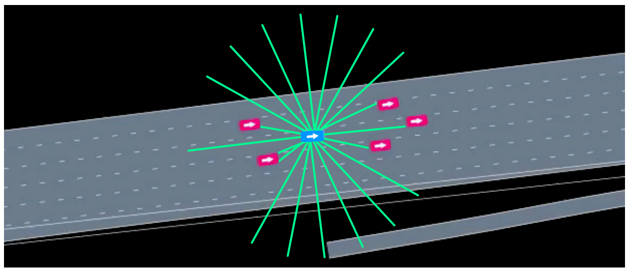


Figure 34: Depiction of LIDAR sensor input used for GAIL models.

we take as inputs: geometric parameters (vehicle length/width), dynamical states (vehicle speed, lateral and angular offsets with respect to the center and heading of the lane, distance to left and right lane boundaries, and local lane curvature), three indicators for collision, road departure, and traveling in reverse, and LIDAR sensor observations (ranges and range-rates of 20 LIDAR beams) as depicted in Figure 34. The generator has two hidden layers of 200 and 100 neurons. The output consists of the mean and variance of normal distributions for throttle and steering commands; we then sample from these distributions to draw a given vehicle's action. The discriminator shares the same size for hidden layers. The forward model used to allow fully-differentiable training first encodes both the state and action through a 150 neuron layer and also adds a GRU layer to the state encoding. A Hadamard product of the results creates a joint embedding which is put through three hidden layers each of 150 neurons. The output is a prediction of the next state.

The end-to-end highway autopilot model is a direct implementation of Bojarski et al. (2016) via the code found at the link `https://github.com/sullychen/autopilot-tensorflow`. In our implementation of the vision-based policy, this highway autopilot model uses rendered images to produce steering commands. LIDAR inputs are used to generate throttle commands using the same network as the non-vision policy.

APPENDIX: Gradient-guided Bridge Sampling

B.1. Split Hamiltonian Monte Carlo (HMC)

In this section, we provide a brief overview of HMC as well as the specific rendition, split HMC (Shahbaba et al., 2014). Given "position" variables x and "momentum" variables v, we define the Hamiltonian for a dynamical system as $H(x, v)$ which can usually be written as $U(x) + K(v)$, where $U(x)$ is the potential energy and $K(v)$ is the kinetic energy. For MCMC applications, $U(x) = -\log(\rho_0(x))$ and we take $v \sim \mathcal{N}(0, I)$ so that $K(v) = \|v\|^2/2$. In HMC, we start at state x_i and sample $v_i \sim \mathcal{N}(0, I)$. We then simulate the Hamiltonian, which is given by the partial differential equations:

$$\dot{x} = \frac{\partial H}{\partial v}, \quad \dot{v} = -\frac{\partial H}{\partial x}.$$

Of course, this must be done in discrete time for most Hamiltonians that are not perfectly integrable. One notable exception is when x is Gaussian, in which case the dynamical system corresponds to the evolution of a simple harmonic oscillator (*i.e.* a spring-mass system). When done in discrete time, a symplectic integrator must be used to ensure high accuracy. After performing some discrete steps of the system (resulting in the state (x_f, v_f)), we negate the resulting momentum (to make the resulting proposal reversible), and then accept the state $(x_f, -v_f)$ using the standard Metropolis-Hastings criterion: $\min(1, \exp(-H(x_f, -v_f) + H(x_i, v_i)))$ (Hastings, 1970).

The standard symplectic integrator—the leap-frog integrator—can be derived using the following symmetric decomposition of the Hamiltonian (performing a symmetric decomposition retains the reversibility of the dynamics): $H(x, v) = U(x)/2 + K(v) + U(x)/2$. Using simple Euler integration for each term individually results in the following leap-frog step of step-size

Algorithm 6 SplitHMC

```
**Input:** Sample $x$, momentum $v \sim \mathcal{N}(0, I)$, scale factor $\beta$, step size $\epsilon$
$v \leftarrow v - 0.5\epsilon\beta I\{f(x) > \gamma\}$
$\hat{x} \leftarrow x\cos(\epsilon) + v\sin(\epsilon)$
$\hat{v} \leftarrow v\cos(\epsilon) - x\sin(\epsilon)$
$\hat{v} \leftarrow \hat{v} - 0.5\epsilon\beta I\{f(\hat{x}) > \gamma\}\nabla f(\hat{x})$
$v \leftarrow -\hat{v}$
$x \leftarrow \hat{x}$ with probability $\min(1, \exp(-H(\hat{x}, \hat{v}) + H(x, v)))$
**Return** $x$
```

ϵ:

$$\begin{aligned} v_{1/2} &= v_i - \frac{\epsilon}{2}\frac{\partial U(x_i)}{\partial x} \\ x_f &= x_i + \epsilon\frac{\partial K(v_{1/2})}{\partial v} \\ v_f &= v_{1/2} - \frac{\epsilon}{2}\frac{\partial U(x_f)}{\partial x}, \end{aligned}$$

where each step simply simulates the individual Hamiltonian $H_1(x, v) = U(x)/2$, $H_2(x, v) = K(v)$, or $H_3(x, v) = U(x)/2$ in sequence. As presented by Shahbaba et al. (2014), this same decomposition can be done in the presence of more complicated Hamiltonians. In particular, consider the Hamiltonian $H(x, v) = U_1(x) + U_0(x) + K(v)$. We can decompose this in the following manner: $H_1(x, v) = U_1(x)/2$, $H_2(x, v) = U_0(x) + K(v)$, and $H_3(x, v) = U_1(x)/2$. We can apply Euler integration to the momentum v for the first and third Hamiltonians and the standard leap-frog step to the second Hamiltonian (or even analytic integration if possible). For this paper, we have $U_0(x) = -\log\rho_0(x)$ and $U_1(x) = -\beta[\gamma - f(x)]_-$.

This is summarized in Algorithm 6

HMC and non-smooth functions In Section 4.3, we assumed that the measure of non-differentiable points is zero for the energy potentials considered by HMC. As discussed by Afshar and Domke (2015), the inclusion of the Metropolis-Hastings acceptance criterion as well as the above assumption ensures that HMC asymptotically samples from the correct distribution even for non-smooth potentials. An equivalent intuitive explanation for this can be seen by viewing the ReLU function $[x]_+$ as the limit of softplus functions $g_k(x) :=$

$\log(1+\exp(kx))/k$ as the sharpness parameter $k \to \infty$. We can freely choose k such that, up to numerical precision, Algorithm 6 is the same whether we consider using a ReLU or sufficiently sharp (*e.g.* large k) softplus potential, because, with probability one, we will not encounter the points where the potentials differ. When further knowledge about the structure of the non-differentiability is known, the acceptance rate of HMC proposals can be improved (Pakman and Paninski, 2013; Lan et al., 2014; Pakman and Paninski, 2014; Afshar and Domke, 2015; Chaari et al., 2016).

B.2. Performance analysis

B.2.1. Proof of Proposition 1

We begin with showing the convergence of the number of iterations. To do this, we first show almost sure convergence of β_k in the limit $N \to \infty$. We note that in the optimization problem (4.6), β_k is a feasible point, yielding $b_k(\beta) = 1$. Thus, $\beta_{k+1} \geq \beta_k \geq \beta_0 := 0$. Due to this growth of β_k with k, we have

$$\frac{Z_{k+1}}{Z_k} = \mathbb{E}_{P_k}\left[\frac{\rho_{k+1}(X)}{\rho_k(X)}\right] \leq 1,$$

$$\begin{aligned}
\mathbb{P}_k(f(X) \leq \gamma) &= \mathbb{E}_{P_{k+1}}\left[\frac{Z_{k+1}}{Z_k}\frac{\rho_k(X)}{\rho_{k+1}(X)} I\{f(X) \leq \gamma)\}\right] \\
&= \frac{Z_{k+1}}{Z_k}\mathbb{E}_{P_{k+1}}\left[I\{f(X) \leq \gamma)\}\right] \\
&\leq \mathbb{P}_{k+1}(f(X) \leq \gamma).
\end{aligned}$$

By the uniform convergence of empirical measures offered by the Glivenko-Cantelli Theorem, the value $a_k \to \mathbb{P}_k(f(X) \leq \gamma)$ almost surely. Then, the stop condition can be rewritten as $b_k(\beta) \geq a_k/s \to \mathbb{P}_k(f(X) \leq \gamma)/s \geq p_\gamma/s$. Since $b_k(\beta)$ is monotonically decreasing in the quantity $\beta - \beta_k$, this constraint gives an upper bound for β_{k+1}, and, as a result, all β_k are almost surely bounded from above and below. We denote this interval as $\mathcal{B}$.

Now, we consider the convergence of the solutions to the finite N versions of problem (4.6), denoted β_k^N, to the "true" optimizers β_k in the limit as $N \to \infty$. Leaving the dependence on β_k implicit for the moment, we consider the random variable $Y := g(X;\beta) := \exp\left((\beta - \beta_k)[\gamma - f(X)]_-\right)$. Then, since $\beta \in \mathcal{B}$ is bounded and g is continuous in β, we can state the Glivenko-Cantelli convergence of the empirical measure uniformly over $\mathcal{B}$: $\sup_{\beta \in \mathcal{B}} \|F^N(Y) - F(Y)\|_\infty \to 0$ almost surely, where F is the cumulative distribution function for Y. Note that the constraints in the problem (4.6) can be rewritten as expectations of this random variable Y. Furthermore, the function g is strictly monotonic in β (and therefore invertible) for non-degenerate $f(X)$ (*i.e.* $f(x) > \gamma$ for some non-negligible measure under P_0). Thus, we have almost sure convergence of the argmin β_{k+1}^N to β_{k+1}.

Until now, we have taken dependence on β_k implicitly. Now we make the dependence explicit to show the final step of convergence. In particular, we can write β_{k+1} as a function of β_k (along with their empirical counterparts), For concreteness, we consider the following decomposition for two iterations:

$$|\beta_2^N(\beta_1^N) - \beta_2(\beta_1)| \le |\beta_2^N(\beta_1^N) - \beta_2(\beta_1^N)| + |\beta_2(\beta_1^N) - \beta_2(\beta_1)|.$$

We have already shown above that the first term on the right hand side vanishes almost surely. By the same reasoning, we know that $\beta_1^N \to \beta_1$ almost surely. The second term also vanishes almost surely since $\beta_{k+1}(\beta)$ is a continuous mapping. This is due to the fact that the constraint functions in problem (4.6) are continuous functions of both β and β_k along with the invertibility properties discussed previously. Then, we simply extend the telescoping series above for any k and similarly show that all terms vanish almost surely. This shows the almost sure convergence for all β_k up to some K.

Now we must show that K is bounded and almost surely converges to a constant. To do this we explore the effects of the optimization procedure. Assuming the stop condition (the second constraint) does not activate, the first constraint in problem (4.6) has the effect of making $\mathbb{Z}_{k+1}/Z_k = \alpha$ (almost surely), which implies $\mathbb{P}_{k+1}(f(X) \le \gamma) = \mathbb{P}_k(f(X) \le \gamma)/\alpha$.

In other words, we magnify the event of interest by a factor of $1/\alpha$. The second constraint can be rewritten as $\mathbb{P}_{k+1}(f(X) \leq \gamma) \leq s$. Thus, we magnify the probability of the region of interest by factors of α unless doing so would increase the probability to greater than s. In that case, we conclude with setting the probability to s (since $\mathbb{P}_{\beta}(f(X) \leq \gamma)$ is monotonically increasing in β). In this way, we have 0 iterations for $p_\gamma \in [s, 1]$, 1 iteration for $p_\gamma \in [\alpha s, s)$, 2 iterations for $p_\gamma \in [\alpha^2 s, \alpha s)$, and so on. Then, the total number of iterations is (almost surely) $\lfloor \log(p_\gamma)/\log(\alpha) \rfloor + I\{p_\gamma/\alpha^{\lfloor \log(p_\gamma)/\log(\alpha) \rfloor} < s\}$.

Now we move to the relative mean-square error of $\hat{p}_\gamma$. We employ the delta method, whereby, for large N, this is equivalent to $\mathrm{Var}(\log(\hat{p}_\gamma))$ (up to terms $o(1/N)$). For notational convenience, we decompose $\widehat{E}_k$ into its numerator and denominator:

$$A_k(X) := \rho_k^B(X)/\rho_{k-1}(X), \qquad \widehat{A}_k := \frac{1}{N} \sum_{i=1}^{N} A_k(x_i^{k-1})$$

$$B_k(X) := \rho_k^B(X)/\rho_k(X), \qquad \widehat{B}_k := \frac{1}{N} \sum_{i=1}^{N} B_k(x_i^{k}).$$

By construction (and assumption of large T), Algorithm 2 has a Markov property that each iteration's samples x_i^k are independent of the previous iterations' samples x_i^{k-1} given β_k. For shorthand, let $\beta_{0:k}$ denote all $\beta_0, \ldots, \beta_k$. Conditioning on $\beta_{0:k}$, we have

$$\mathrm{Var}(A_k) = \mathrm{Var}\left(\mathbb{E}[A_k|\beta_{0:k}]\right) + \mathbb{E}\left[\mathrm{Var}\left(A_k|\beta_{0:k}\right)\right].$$

Since $\beta_{0:k}$ approaches constants almost surely as $N \to \infty$, the first term vanishes and the second term is the expectation of a constant. In particular, the second term is as follows:

$$\begin{aligned}\mathrm{Var}\left(A_k|\beta_{0:k}\right) &= \mathbb{E}\left[A_k^2|\beta_{0:k}\right] - \left(\mathbb{E}\left[A_k|\beta_{0:k}\right]\right)^2 \\ &= \mathbb{E}_{P_{k-1}}\left[\frac{\rho_k(X)}{\rho_{k-1}(X)}\right] - \left(\mathbb{E}_{P_{k-1}}\left[\sqrt{\frac{\rho_k(X)}{\rho_{k-1}(X)}}\right]\right)^2 \\ &= \frac{Z_k}{Z_{k-1}} - \left(\frac{Z_k^B}{Z_{k-1}}\right)^2.\end{aligned}$$

Similarly, $\mathrm{Var}(B_k|\beta_{0:k}) = Z_{k-1}/Z_k - (Z_k^B/Z_k)^2$. Next we look at the covariance terms:

$$\mathrm{Cov}(A_{k-1}, A_k) = \mathrm{Cov}\left(\mathbb{E}[A_{k-1}|\beta_{0:k}], \mathbb{E}[A_k|\beta_{0:k}]\right) + \mathbb{E}\left[\mathrm{Cov}\left(A_{k-1}, A_k|\beta_{0:k}\right)\right].$$

Again, the first term vanishes since $\beta_{0:k}$ approach constants as $N \to \infty$. By construction, the second term is also 0 since the quantities are conditionally independent. Similarly, $\mathrm{Cov}(B_{k-1}, B_k) = 0$ and $\mathrm{Cov}(A_i, B_j) = 0$ for $j \neq i-1$. However, there is a nonzero covariance for the quantities that depend on the same distribution:

$$\begin{aligned}\mathrm{Cov}\left(B_k, A_{k+1}|\beta_{0:k+1}\right) &= \mathbb{E}\left[B_k A_{k+1}|\beta_{0:k+1}\right] - \mathbb{E}\left[B_k|\beta_{0:k+1}\right]\mathbb{E}\left[A_{k+1}|\beta_{0:k+1}\right] \\ &= \mathbb{E}_{P_k}\left[\frac{\sqrt{\rho_{k-1}(X)\rho_{k+1}(X)}}{\rho_k(X)}\right] - \frac{Z_{k+1}^B}{Z_k}\frac{Z_k^B}{Z_k} \\ &= \frac{Z_k^C}{Z_k} - \frac{Z_{k+1}^B}{Z_k}\frac{Z_k^B}{Z_k}.\end{aligned}$$

By the large T assumption, the samples x_i^k and x_j^k are independent for all $i \neq j$ given β_k. Then we have

$$\mathrm{Var}(\widehat{A}_k|\beta_{0:k}) = \mathrm{Var}(A_k|\beta_{0:k})/N, \ \ \mathrm{Var}(\widehat{B}_k|\beta_{0:k}) = \mathrm{Var}(B_k|\beta_{0:k})/N,$$

$$\mathrm{Cov}(\widehat{B}_k, \widehat{A}_{k+1}|\beta_{0:k+1}) = \mathrm{Cov}(B_k, A_{k+1}|\beta_{0:k+1})/N.$$

The last term in $\hat{p}_\gamma$, $\frac{1}{N}\sum_{i=1}^N \frac{\rho_\infty(x_i^K)}{\rho_K(x_i^K)}$, reduces to a simple Monte Carlo estimate since $\frac{\rho_\infty(X)}{\rho_K(X)} = I\{f(X) \leq \gamma\}$. Furthermore, this quantity is independent of all other quantities given $\beta_{0:K}$ and, as noted above, approaches s almost surely as $N \to \infty$.

Putting this all together, the delta method gives (as $N \to \infty$ so that $\beta_{0:K}$ approach constants almost surely),

$$\mathrm{Var}(\log(\hat{p}_\gamma)) \to \sum_{k=1}^K \left(\frac{\mathrm{Var}(\widehat{A}_k)}{(Z_k^B/Z_{k-1})^2} + \frac{\mathrm{Var}(\widehat{B}_k)}{(Z_k^B/Z_k)^2}\right) - 2\sum_{k=1}^{K-1}\frac{\mathrm{Cov}(\widehat{B}_k, \widehat{A}_{k+1})}{Z_{k+1}^B Z_k^B/Z_k^2} + \frac{1-s}{sN} + o\left(\frac{1}{N}\right).$$

The Bhattacharrya coefficient can be written as

$$G(P_{k-1}, P_k) = \int_{\mathcal{X}} \sqrt{\frac{\rho_{k-1}(x)}{Z_{k-1}} \frac{\rho_k(x)}{Z_k}} dx = \frac{Z_k^B}{\sqrt{Z_{k-1} Z_k}}.$$

Furthermore, we have

$$\frac{G(P_{k-1}, P_{k+1})}{G(P_{k-1}, P_k) G(P_k, P_{k+1})} = \frac{Z_k^C}{\sqrt{Z_{k-1} Z_{k+1}}} \frac{\sqrt{Z_{k-1} Z_k}}{Z_k^B} \frac{\sqrt{Z_k Z_{k+1}}}{Z_{k+1}^B} = \frac{Z_k^C Z_k}{Z_k^B Z_{k+1}^B},$$

yielding this final result

$$\mathrm{Var}(\log(\hat{p}_\gamma)) \to \frac{2}{N} \sum_{k=1}^{K} \left(\frac{1}{G(P_{k-1}, P_k)^2} - 1 \right) - \frac{2}{N} \sum_{k=1}^{K-1} \left(\frac{G(P_{k-1}, P_{k+1})}{G(P_{k-1}, P_k) G(P_k, P_{k+1})} - 1 \right) + \frac{1-s}{sN} + o\left(\frac{1}{N}\right). \quad \text{(B.1)}$$

We remark that a special case of this formula is for $K = 1$ and $s = 1$ (so only the first term survives), which is the relative mean-square error for a single bridge-sampling estimate $\widehat{E}_k$.

Now, since $G(P, Q) \geq 0$, the terms in the second sum are ≥ -1 so that the second sum is $\leq 2(K-1)/N$. Furthermore, since $s \geq 1/3$, the last term is also $\leq 2/N$. Thus, if we have $\frac{1}{G(P_{k-1}, P_k)^2} \leq D$ (with $D \geq 1$), then the asymptotic relative mean-square error (B.1) is $\leq 2KD/N$ (up to terms $o\left(\frac{1}{N}\right)$).

B.3. Experimental setups

B.3.1. Hyperparameters

The number of samples N affects the absolute performance of all of the methods tested, but not their relative performance with respect to each other. For all experiments, we use $N = 1000$ for GGB to have adequate absolute performance given our computational budget (see below for the computing architecture used). Other hyperparameters were tuned on the synthetic problem and fixed for the rest of the experiments. The hyperparameters were chosen as follows.

When performing Hamiltonian dynamics for a Gaussian variable, a time step of 2π results in no motion and time step of π results in a mode reversal, where both the velocity and position are negated. The π time step is in this sense the farthest exploration that can occur in phase space (which can be intuitively understood by recognizing that the phase diagram of a simple spring-mass system is a unit circle). Thus, we considered $T = 4, 8, 10, 12$, and 16 with time steps π/T. We found that $T = 10$ provided reasonable exploration (as measured by autocorrelations and by the bias of the final estimator $\hat{p}_\gamma$) and higher values of T did not provide much more benefit. We also performed tuning online for the time step to keep the acceptance ratio between 0.4 and 0.8. This was done by setting the time step to $\sin^{-1}(\min(1, \sin(t) \exp((p - C)/2))$, where t is the current time step, p is the running acceptance probability for a single chain and $C = 0.4$ if $p < 0.4$ or 0.8 if $p > 0.8$. This was done after every T HMC steps.

For the step size of the bridge, we considered $\alpha \in \{0.01, 0.1, 0.3, 0.5\}$. Smaller α results in fewer iterations and better computational efficiency. We settled on $\alpha = 0.3$, which provided reasonable computational efficiency (no more than 11 iterations for the synthetic problem). For AMS, we followed the hyperparameter settings of Webb et al. (2018). Namely, we chose a culling fraction of $\alpha_{\text{AMS}} = 10\%$, where α_{AMS} sets the fraction of particles that are removed and rejuvenated at each iteration (Webb et al., 2018).

Given the above parameters, the number of simulations for each experiment varies based on the final probability in question p_γ (smaller values result in more simulations due to having a higher number of iterations K). We had runs of 111000, 91000, 101000, and 160000 simulations respectively for the synthetic, AttentionAgentRacer, WorldModelRacer, and OpenPilot environments. We used these values as well as the ground truth p_γ values to determine the number of particles allowed for AMS, $N_{\text{AMS}} = 920, 820, 910, 920$ respectively, as AMS has a total cost of $N_{\text{AMS}}(1 + \alpha_{\text{AMS}} T K_{\text{AMS}})$, where $K_{\text{AMS}} \approx \log(p_\gamma)/\log(1 - \alpha_{\text{AMS}})$.

For the surrogate Gaussian process regression model for CarRacing and OpenPilot, we retrained the model on the most recent N simulations after every NT simulations (*e.g.* after every T HMC iterations). This made the amortized cost of training the surrogate model negligible compared to performing the simulations themselves. We used a Matern kernel with parameter $\nu = 2.5$. We optimized the kernel hyperparameters using an L-BFGS quasi-Newton solver.

Computing infrastructure and parallel computation Experiments were carried out on commodity CPU cloud instances, each with 96 Intel Xeon cores @ 2.00 GHz and 85 GB of RAM. AMS, GGB are designed to work in a Map-Reduce paradigm, where a central server orchestrates many worker jobs followed by synchronization step. AMS requires more iterations and fewer parallel worker threads per iteration than GGB. In particular, whereas GGB performs N parallel jobs per iteration, AMS only performs $\alpha_{\text{AMS}} N_{\text{AMS}}$ parallel jobs per iteration. Thus, GGB takes advantage of massive scale and parallelism much more than AMS.

B.3.2. Environment details

Car Racing

We compare the failure rate of agents solving the car-racing task utilizing the two distinct approaches ((Ha and Schmidhuber, 2018a) and (Tang et al., 2020)). The car racing task differs from the other experiments due to the inclusion of a (simple) renderer in the system dynamics. At each the step the agent receives a reward of $-0.1 + \mathcal{I}_{newtile}(1000/N) - \mathcal{I}_{offtrack}(100)$ where N is the total number of tiles visited in the track. The environment is considered solved if the agent returns an average reward of 900 over 100 trials. The search space P_0 is the inherent randomness involved with generating a track. The track is generated by selecting 12 checkpoints in polar coordinates, each with radian value uniformly in the interval $[2\pi i/12, 2\pi(i+1)/12)$ for $i = 0, \ldots 11$, and with radius uniformly in the interval $[R/3, R]$, for a given constant value R. This results in 24 parameters in the search space. The policies used for testing are described below (with training scripts in the code supplement).

AttentionAgent Tang et al. (2020) utilize a simple self-attention module to select patches from a 96x96 pixel observation. First the input image is normalized then a sliding window approach is used to extract N patches of size $M \times M \times 3$ which are flattened and arranged into a matrix of size $3M^2 \times N$. The self-attention module is used to compute the attention matrix A and importance vector (summation of each column of A). A feature extraction operation is applied to the top K elements of the sorted importance vector and the selected features are input to a neural network controller. Both the attention module and the controller are trained together via CMA-ES. Together, the two modules contain approximately 4000 learnable parameters. We use the pre-trained model available here: `https://github.com/google/brain-tokyo-workshop/tree/master/AttentionAgent`.

WorldModel The agent of Ha and Schmidhuber (2018a) first maps a top-down image of the car on track via a variational autoencoder to a latent vector z. Given z, the world model M utilizes a recurrent-mixture density network (Bishop, 1994) to model the distribution of future possible states $P(z_{t+1} \mid a_t, z_t, h_t)$. Note that h_t, the hidden state of the RNN. Finally, a simple linear controller C maps the concatenation of z_t and h_t to the action, a_t. We use the pre-trained model available here: `https://github.com/hardmaru/WorldModelsExperiments/tree/master/carracing`.

APPENDIX: TunerCar

C.0.1. System identification

In this section, we describe the procedure used to identify the vehicle parameters, namely the mass, the location of the center of gravity, the moment of inertia, the friction coefficient, the cornering stiffness, and the maximum acceleration/deceleration rates.

Mass and center of gravity

The vehicle weighs 3518.6±0.1 g, including the mass of the lithium polymer battery. The center of gravity is estimated by balancing the vehicle on the edge of a ruler. For a wheelbase of 0.317 m, the distance of the center of gravity from the front wheels l_f and the rear wheels l_r is measured to be 0.147±0.005 m and 0.147±0.005 m, respectively.

Moment of Inertia

We use the bifilar (two-wire) pendulum method. This method is used to measure the moment of inertia of symmetric objects, such as airplanes (Soule and Miller, 1934), unmanned air vehicles (Jardin and Mueller, 2009), and tennis rackets (Spurr et al., 2014). The bifilar pendulum is a torsional pendulum that consists of the test object suspended by two thin parallel wires that are equidistant from the center of gravity. A small moment is applied to the vehicle, and the angular frequency ω is found by recording the period of oscillation about the vertical axis that goes through the center of gravity. The moment of inertia I_z is given by

$$I_z = \frac{mgd^2}{4L\omega^2}, \tag{C.1}$$

where m is the mass of the vehicle, g the acceleration due to gravity, d the distance between the wires, and L the length of the wires. Equation (C.1) is obtained from the nonlinear mathematical model of a bifilar pendulum; the derivation can be found in (Kotikalpudi et al., 2013). Figure 35 shows the setup of the vehicle used in this paper. The moment of inertia of the vehicle is estimated to be 0.047 kg·m^2.

Friction coefficient and cornering stiffness

The simulator is based on the single-track vehicle model (Althoff et al., 2017). This model relates the friction coefficient μ, the cornering stiffness coefficient C_s, and the vertical force F_z by

$$C_i = \mu C_{S,i} F_{z,i}, \tag{C.2}$$

where $i = \{f, r\}$ for the front and rear axle, respectively. A force scale is used to measure the kinetic friction coefficient μ between the rubber tires and linoleum floor as the vehicle was dragged laterally at a constant velocity. μ is estimated to be 0.5230±0.0014. The cornering stiffness coefficients are $C_{S,f}$ which is estimated to be 4.191±0.002 and $C_{S,r}$ to be 4.8469±0.002.

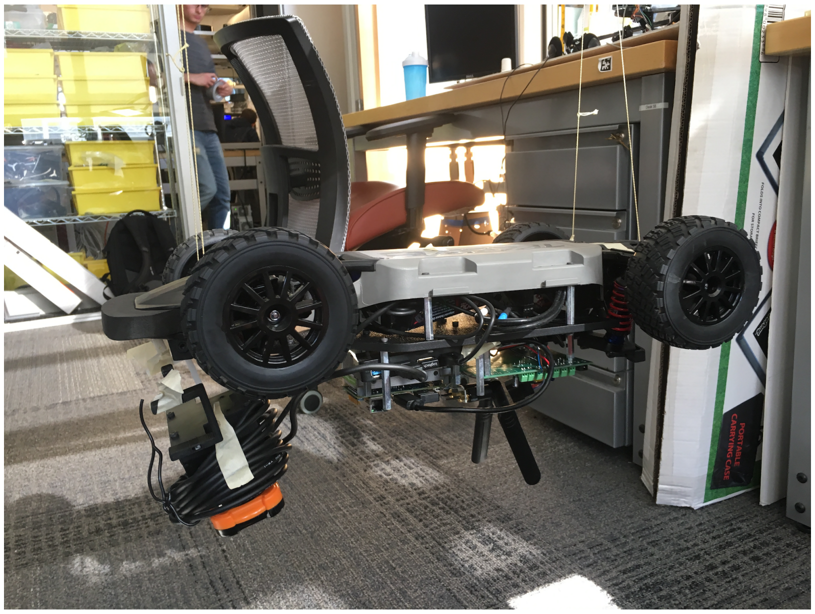


Figure 35: Identification of the moment of inertia

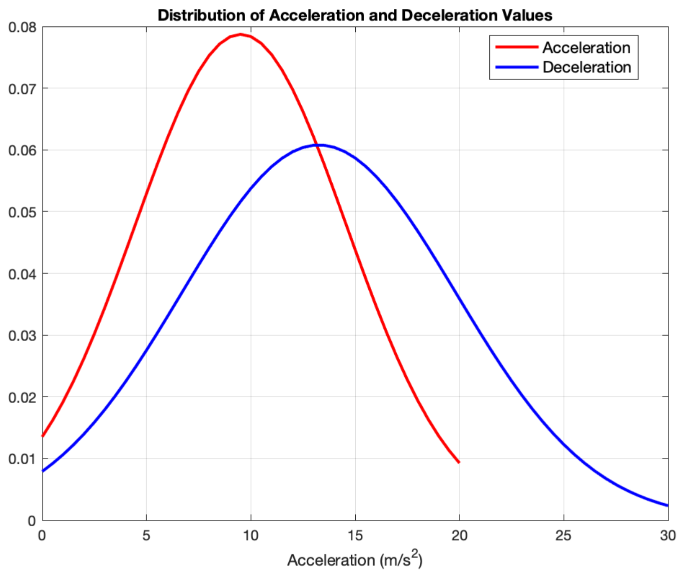


Figure 36: The distributions of maximum acceleration and deceleration obtained from 19 speed tests.

Maximum acceleration and deceleration rates

We perform 19 speed tests ranging from 2 m/s to 11 m/s in increments of 0.5 m/s. The vehicle is commanded to accelerate to a certain speed, maintain that speed for 0.5 s, and then brake. The position data from the particle filter and velocity data from the vehicle odometry are recorded for all the tests. The acceleration and deceleration sections are fitted with separate exponential curves, and the constant speed region is fitted with a linear curve. We estimate the maximum acceleration and deceleration for each test; the distributions are shown in Figure 36. The mean for the acceleration curve $a_{\max}$ is 9.51 m/s^2 while the mean for the deceleration curve $d_{\max}$ is 13.25 m/s^2.

C.0.2. Minimum time path

The objective in autonomous racing is to determine a trajectory that requires minimum time to traverse a track for known vehicle dynamics. We represent this dynamics by $\dot{\mathbf{x}} = f\left(\mathbf{x}(t), \mathbf{u}(t)\right)$, where $\mathbf{x}$ denotes the state of the vehicle and $\mathbf{u}$ the set of control inputs.

Formally, the problem can be stated as

$$\begin{aligned} \underset{T,\mathbf{u}(t)}{\text{minimize}} \quad & \int_0^T 1 dt \\ \text{subject to} \quad & \dot{\mathbf{x}} = f\left(\mathbf{x}(t), \mathbf{u}(t)\right), \\ & \mathbf{x}(0) = \mathbf{x}_S, \; \mathbf{x}(T) = \mathcal{X}_F, \\ & \mathbf{x}(t) \in \mathcal{X}, \; \mathbf{u}(t) \in \mathcal{U}. \end{aligned} \tag{C.3}$$

Here, the second set of constraints includes an initial condition for the start line and a terminal condition for crossing the finish line. $\mathcal{X}$ and $\mathcal{U}$ capture track and actuation constraints, respectively. Problem (C.3) is an example of a minimum time optimal control problem and is computationally hard to solve, especially in the presence of nonlinear constraints (Athans and Falb, 2013).

Now, for a fixed trajectory, calculation of minimum time to traverse and the corresponding speed profile will require solving (C.3) with an additional constraint that (x, y) must lie on the trajectory. It turns out when a friction circle model represents the vehicle dynamics; this new problem is much easier to solve.

The friction circle model with a rear-wheel drive given by

$$m \begin{bmatrix} \ddot{x} \\ \ddot{y} \end{bmatrix} = \begin{bmatrix} \cos(\phi) & -\sin(\phi) \\ \sin(\phi) & \cos(\phi) \end{bmatrix} \begin{bmatrix} F_{\text{long}} \\ F_{\text{lat}} \end{bmatrix}, \tag{C.4}$$

where m is the mass of the vehicle and ϕ the orientation of the vehicle defined as a function of position (x, y) in the global frame. The inputs to the model are a force in the longitudinal direction F_{long} and a force in the lateral direction F_{lat} defined in the frame attached to the vehicle. We enforce a constraint for the friction circle

$$\sqrt{F_{\text{long}}^2 + F_{\text{lat}}^2} \leq \mu_s m g, \tag{C.5}$$

where μ_s is the static coefficient of friction, g is the acceleration due to gravity, and a constraint for the maximum possible force with the rear-wheel drive

$$F_{\text{long}} \leq \frac{l_f}{l_f + lr}\, \mu_s m g, \tag{C.6}$$

where l_f and l_r are the distance of the center of gravity from the front and the rear wheels in the longitudinal direction, respectively. The model ignores the effect of tire slips. The advantage of using the friction circle model is that it requires minimum effort in system identification with only three parameters to identify, namely m, l_f, and l_r.

By transforming the optimization problem from a generalized position space to a path coordinate space and subsequently applying the nonlinear change of variables, the problem of calculating minimum time over a fixed path can be formulated as a convex optimization problem (Verscheure et al., 2009). For the friction circle model (C.4) with additional constraints (C.5) and (C.6), the optimization is still convex (Lipp and Boyd, 2014).

APPENDIX: FormulaZero

D.1. Offline population synthesis

Here we provide extra details for Section 7.3.

Horizontal steps Horizontal steps occur as follows. Two random particles are sampled uniformly at random from adjacent temperature 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 particularly simple form (*cf.* Algorithm 7).

Algorithm 7 Horizontal swap

```
Sample $i \sim \text{Uniform}(1, 2, \ldots, L-1)$.
Sample $j, k \overset{\text{i.i.d.}}{\sim} \text{Uniform}(1, 2, \ldots, D)$.
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.

D.2. Online robust planning

Here we provide extra details for Section 7.4.

D.2.1. Solving problem (7.5)

We can rewrite the constraint $D_f\left(q \| \mathbf{1}/N_w\right) \leq \rho$ as $\|q - \mathbf{1}/N_w\|^2 \leq \rho/N_w$. 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. (2008).

D.2.2. 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

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

To ease notation, we hide dependence on J and for a sample $J_1, \dots, J_{N_w}$ 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 (2016). 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 $C = (C_1, \dots, C_{N_w})$ 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 (Samson, 2000) for Lipschitz convex functions of bounded random variables, we have with probability at least $1-\delta$ that

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

By the containment (D.1), we only need to consider convergence of

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

which we do with the following lemma.

Lemma 1 (Sinha and Duchi (2016))**.** *Let* $Z = (Z_1, \dots, Z_{N_w})$ *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 D.2.3 for the proof.

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

D.2.3. Proof of Lemma 1

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

Lemma 2 (Ben-Tal et al. (2013))**.** *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*

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

See Appendix D.2.4 for the proof.

We prove the result for general ϕ-divergences $\phi(t) = t^k - 1$, $k \ge 2$. 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

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

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

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

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

$$\eta \in \left[-\frac{1+(1+\rho)^{\frac{1}{k^*}}}{(1+\rho)^{\frac{1}{k^*}}-1}, 1\right]\cdot M. \tag{D.2}$$

For convenience, we first define the shorthand

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

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

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

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

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

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

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

$$\begin{aligned} \mathbb{E}\left[\sup_{Q\in\mathcal{P}_{N_w}} \mathbb{E}_Q[Z]\right] &= \mathbb{E}\left[\inf_{\eta\in U} S_{N_w}(\eta\right] \\ &= \mathbb{E}\left[\inf_{\eta\in U} \{S_{N_w}(\eta) - \mathbb{E}[S_{N_w}(\eta)] + \mathbb{E}[S_{N_w}(\eta)]\}\right] \\ &\geq \inf_{\eta\in U} \mathbb{E}[S_{N_w}(\eta)] \\ &\quad - \mathbb{E}\left[\sup_{\eta\in U} |S_{N_w}(\eta) - \mathbb{E}[S_{N_w}(\eta)]|\right]. \end{aligned} \tag{D.3}$$

To bound the first term in expression (D.3), we use the following lemma.

Lemma 3 (Sinha and Duchi (2016)). *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:*

$$\mathbb{E}\left[\left(\frac{1}{n}\sum_{i=1}^{n} Z_i^p\right)^{\frac{1}{p}}\right] \geq \|Z\|_p - \frac{p-1}{p}\sqrt{\frac{2}{n}}\sqrt{\mathrm{Var}(Z^p/\mathbb{E}[Z^p])}\|Z\|_2, \tag{D.4a}$$

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

$$\mathbb{E}\left[\left(\frac{1}{n}\sum_{i=1}^{n} Z_i^p\right)^{\frac{1}{p}}\right] \geq \|Z\|_p - C\frac{p-1}{p}\sqrt{\frac{2}{n}}. \tag{D.4b}$$

See Appendix D.2.5 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

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

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 (D.3), 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

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

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

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

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

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

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

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

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

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

This gives the desired result of the lemma.

D.2.4. Proof of Lemma 2

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

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

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 (Luenberger, 1969, Theorem 8.6.1 and Problem 8.7). Noting that L is simply a positive (but otherwise arbitrary) function, we obtain

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

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

D.2.5. Proof of Lemma 3

For $a > 0$, we have

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

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

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

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 $\lambda \geq 0$ that

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

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 $c = \frac{2}{a} - 1$. $\square$

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 $\text{Var} Z^p \leq \mathbb{E}[Z^{2p}] \leq C^p\|Z\|_p^p$), 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.

D.2.6. Proof of Proposition 3

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*

$$\sum_{t=1}^T \mathbb{E}\left[\gamma(t)^T(w(t) - w^\star)\right] \leq \frac{\log(d)}{\eta} + \frac{\eta}{2}\mathbb{E}\left[\sum_{t=1}^T\sum_{j=1}^d w_j(t)\gamma_j(t)^2\right] \tag{D.5}$$

See Appendix D.2.7 for the proof. Now we bound the right-hand term of the regret bound

(D.5) 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,

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

Plugging in the prescribed $\eta=\sqrt{\frac{2\log(d)}{zT}}$ into the bound (D.5) yields the result.

D.2.7. 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

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

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

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

Summing this preceding display over iterations t yields

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

Now let $\psi(w) = \sum_i w_i \log w_i$. Then, with $w_1 = \mathbf{1}/d$, $B\psi(w^*, w_1) \leq \log(d)$. 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*

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

See Appendix D.2.8 for the proof. Setting $y = w_{t+1}$, $x = w_t$, and $g = \gamma_t$ in Lemma 6 yields

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

Taking expectations on both sides yields the result.

D.2.8. 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

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

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

$$-\frac{1}{\eta} B_\psi(y,x) + g^T(x-y) \le \frac{\eta}{2} \sum_{i=1}^d g_i^2 p_i(\widetilde{g}). \tag{D.6}$$

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 $ab \leq \frac{1}{3}|a|^3 + \frac{2}{3}|b|^{3/2}$ for any a, b so $g_i g_j^2 \leq \frac{1}{3} g_i^3 + \frac{2}{3} g_j^3$. This implies that the numerator in our expression for $s^{'}(\lambda)$ is non-positive. Thus, $s(\lambda) \leq s(0) = \sum_{i=1}^d g_i^2 x_i$ which gives the result when combined with inequality (D.6).

D.3. Hardware

The major components of the vehicle used in experiments are shown in Figure 37. 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 (Vedder, 2015) 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.3.1. Mapping

We create occupancy grid maps of tracks using Google Cartographer (Hess et al., 2016). 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.

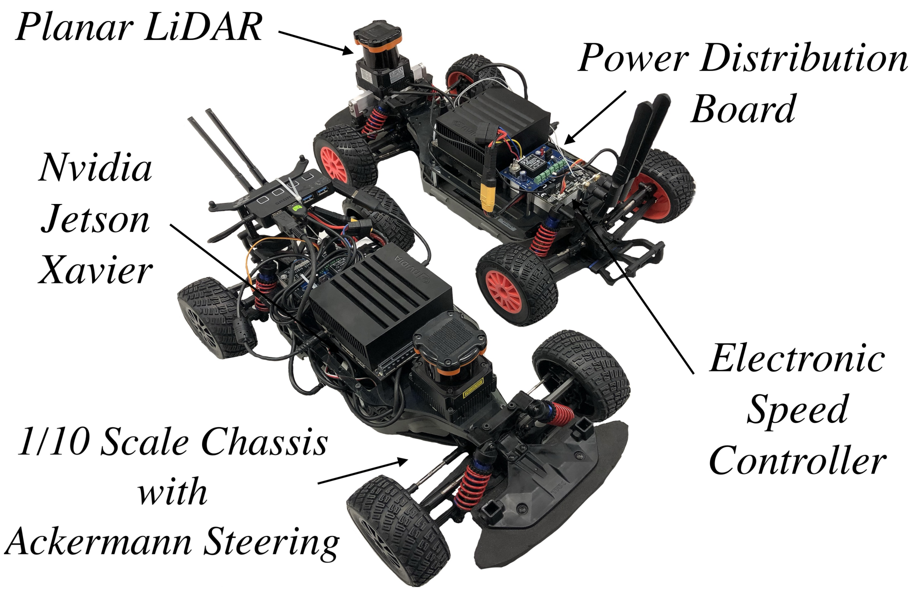

Planar LiDAR
Power Distribution Board
Nvidia Jetson Xavier
Electronic Speed Controller
1/10 Scale Chassis with Ackermann Steering

Figure 37: Components of the 1/10 Scale Vehicle

D.3.2. 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 (Walsh and Karaman, 2017) 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.4. Communication and system architecture

The ZeroMQ (Hintjens, 2013) 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-deterministically 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.

D.5. 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 (Brockman et al., 2016) API is used to perform rollouts of the experiments.

D.5.1. Vehicle Dynamics

The single-track model in Althoff et al. (2017) 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.

D.5.2. 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. (2019).

D.5.3. 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 (Dean and Ghemawat, 2008) pattern between the task distributor, result collector, and the workers. Each worker receives the description of the configuration to be simulated, *e.g.* (x, θ). Then the workers asynchronously perform simulations and send results to the collector.

D.5.4. Addressing the simulation/reality gap

As noted in Section 7.6 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. 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.

D.5.5. 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. (2018)). Norden et al. (2019) 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.

D.5.6. Out-of-distribution agent strategies

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

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* (Karaman and Frazzoli, 2011) 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. (2017), 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 (Coulter, 1992).

OOD2: RL-based Lane Switching

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 38, 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;

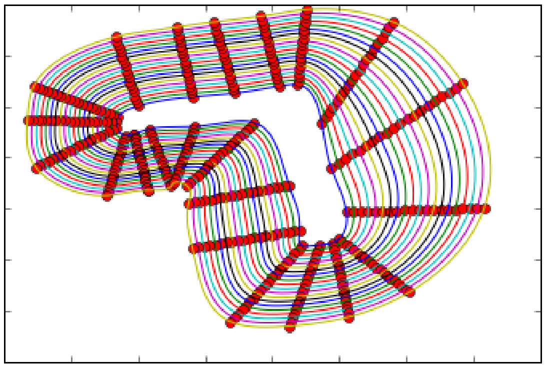


Figure 38: Lanes that cover the track

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 (Fujimoto et al., 2018) is used to estimate the Q function and make the lane switching decisions. iTTC defined in Appendix D.5.5 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 (Coulter, 1992) tracking the selected lane. We used an existing implementation, `https://github.com/pnorouzi/rl-path-racing`, of this algorithm.

BIBLIOGRAPHY

H. Abbas, A. Winn, G. Fainekos, and A. A. Julius. Functional gradient descent method for metric temporal logic specifications. In *2014 American Control Conference*, pages 2312–2317. IEEE, 2014.

H. Abbas, M. O'Kelly, and R. Mangharam. Relaxed decidability and the robust semantics of metric temporal logic. In *Proceedings of the 20th International Conference on Hybrid Systems: Computation and Control*, pages 217–225. ACM, 2017.

H. Abbas, M. O'Kelly, A. Rodionova, and R. Mangharam. Safe at any speed: A simulation-based test harness for autonomous vehicles. In R. Chamberlain, W. Taha, and M. Törngren, editors, *Cyber Physical Systems. Design, Modeling, and Evaluation*, pages 94–106, Cham, 2019. Springer International Publishing.

J. Abernethy and A. Rakhlin. Beating the adaptive bandit with high probability. In *2009 Information Theory and Applications Workshop*, pages 280–289. IEEE, 2009.

H. M. Afshar and J. Domke. Reflection, refraction, and hamiltonian monte carlo. In *Advances in neural information processing systems*, pages 3007–3015, 2015.

N. Agarwal, B. Bullins, E. Hazan, S. M. Kakade, and K. Singh. Online control with adversarial disturbances. *arXiv preprint arXiv:1902.08721*, 2019.

A. Agnihotri, M. O'Kelly, R. Mangharam, and H. Abbas. Teaching autonomous systems at 1/10th-scale: Design of the f1/10 racecar, simulators and curriculum. In *Proceedings of the 51st ACM Technical Symposium on Computer Science Education*, SIGCSE '20, pages 657–663, New York, NY, USA, 2020. Association for Computing Machinery.

A. V. Aho, R. Sethi, and J. D. Ullman. Compilers, principles, techniques. *Addison Wesley*, 7(8):9, 1986.

M. Althoff. An introduction to cora 2015. In *Proc. of the Workshop on Applied Verification for Continuous and Hybrid Systems*, 2015.

M. Althoff and J. M. Dolan. Online verification of automated road vehicles using reachability analysis. *IEEE Transactions on Robotics*, 30(4):903–918, 2014.

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.

R. Alur, C. Courcoubetis, N. Halbwachs, T. A. Henzinger, P.-H. Ho, X. Nicollin, A. Olivero, J. Sifakis, and S. Yovine. The algorithmic analysis of hybrid systems. *Theoretical Computer Science*, 138(1):3–34, 1995.

R. Alur, T. A. Henzinger, G. Lafferriere, and G. J. Pappas. Discrete abstractions of hybrid systems. *Proceedings of the IEEE*, 88(7):971–984, 2000.

Y. S. R. Annapureddy and G. E. Fainekos. Ant colonies for temporal logic falsification of hybrid systems. In *Proc. of the 36th Annual Conference of IEEE Industrial Electronics*, pages 91–96, 2010.

Y. Annpureddy, C. Liu, G. Fainekos, and S. Sankaranarayanan. S-taliro: A tool for temporal logic falsification for hybrid systems. In *International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, pages 254–257. Springer, 2011.

O. Arenz, M. Zhong, G. Neumann, et al. Efficient gradient-free variational inference using policy search. 2018.

T. Arnold and M. Scheutz. Against the moral turing test: accountable design and the moral reasoning of autonomous systems. *Ethics and Information Technology*, 18(2):103–115, 2016. ISSN 1572-8439. doi: 10.1007/s10676-016-9389-x. URL `http://dx.doi.org/10.1007/s10676-016-9389-x`.

S. Arora, A. Bhaskara, R. Ge, and T. Ma. Provable bounds for learning some deep representations. In *International Conference on Machine Learning*, pages 584–592. , 2014.

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

S. Asmussen and P. W. Glynn. *Stochastic Simulation: Algorithms and Analysis*. Springer, 2007.

K. J. Åström and P. Eykhoff. System identification—a survey. *Automatica*, 7(2):123–162, 1971.

K. J. Åström and B. Wittenmark. *Adaptive control*. Courier Corporation, 2013.

M. Athans and P. L. Falb. *Optimal control: an introduction to the theory and its applications*. Courier Corporation, 2013.

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.

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

Baidu Apollo Team. Apollo: Open source autonomous driving, 2017. URL `https://github.com/ApolloAuto/apollo`.

C. Baier and J.-P. Katoen. *Principles of model checking*. MIT press, 2008.

B. Balaji, S. Mallya, S. Genc, S. Gupta, L. Dirac, V. Khare, G. Roy, T. Sun, Y. Tao, B. Townsend, et al. Deepracer: Educational autonomous racing platform for experimentation with sim2real reinforcement learning. *arXiv preprint arXiv:1911.01562*, 2019.

M. Bansal, A. Krizhevsky, and A. Ogale. Chauffeurnet: Learning to drive by imitating the best and synthesizing the worst. *arXiv preprint arXiv:1812.03079*, 2018.

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

N. Baram, O. Anschel, I. Caspi, and S. Mannor. End-to-end differentiable adversarial imitation learning. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 390–399. JMLR. org, 2017.

P. L. Bartlett, D. J. Foster, and M. J. Telgarsky. Spectrally-normalized margin bounds for neural networks. In *Advances in Neural Information Processing Systems*, pages 6241–6250. , 2017.

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.

A. Bemporad and M. Morari. Robust model predictive control: A survey. In *Robustness in identification and control*, pages 207–226. Springer, 1999.

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.

Y. Benkler. Don't let industry write the rules for ai. *Nature*, 569(7754):161–162, 2019.

C. H. Bennett. Efficient estimation of free energy differences from monte carlo data. *Journal of Computational Physics*, 22(2):245–268, 1976.

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

C. Berner, G. Brockman, B. Chan, V. Cheung, P. Dębiak, 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.

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

M. Betancourt. A conceptual introduction to hamiltonian monte carlo. *arXiv preprint arXiv:1701.02434*, 2017.

C. M. Bishop. Mixture density networks. 1994.

C. Blundell, J. Cornebise, K. Kavukcuoglu, and D. Wierstra. Weight uncertainty in neural networks. *arXiv preprint arXiv:1505.05424*, 2015.

M. Bojarski, D. Del Testa, D. Dworakowski, B. Firner, B. Flepp, P. Goyal, L. D. Jackel, M. Monfort, U. Muller, J. Zhang, et al. End to end learning for self-driving cars. *arXiv preprint arXiv:1604.07316*, 2016.

J.-F. Bonnefon, A. Shariff, and I. Rahwan. The social dilemma of autonomous vehicles. *Science*, 352(6293):1573–1576, 2016.

F. Borrelli, A. Bemporad, and M. Morari. *Predictive control for linear and hybrid systems.* Cambridge University Press, 2017.

J. Bowen and V. Stavridou. Safety-critical systems, formal methods and standards. *Software Engineering Journal*, 8(4):189–209, 1993.

C.-E. Bréhier, T. Lelièvre, and M. Rousset. Analysis of adaptive multilevel splitting algorithms in an idealized case. *ESAIM: Probability and Statistics*, 19:361–394, 2015.

G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba. Openai gym. *arXiv preprint arXiv:1606.01540*, 2016.

C. B. Browne, E. Powley, D. Whitehouse, S. M. Lucas, P. I. Cowling, P. Rohlfshagen, S. Tavener, D. Perez, S. Samothrakis, and S. Colton. A survey of monte carlo tree search methods. *IEEE Transactions on Computational Intelligence and AI in games*, 4(1):1–43, 2012.

M. Brundage, S. Avin, J. Wang, H. Belfield, G. Krueger, G. Hadfield, H. Khlaaf, J. Yang, H. Toner, R. Fong, et al. Toward trustworthy ai development: Mechanisms for supporting verifiable claims. *arXiv preprint arXiv:2004.07213*, 2020.

S. Bubeck, N. Cesa-Bianchi, et al. Regret analysis of stochastic and nonstochastic multi-armed bandit problems. *Foundations and Trends® in Machine Learning*, 5(1):1–122, 2012.

J. Bucklew. *Introduction to rare event simulation.* Springer Science & Business Media, 2013.

T. Campos, J. P. Inala, A. Solar-Lezama, and H. Kress-Gazit. Task-based design of ad-hoc modular manipulators. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 6058–6064. IEEE, 2019.

R. Caruana. Multitask learning. *Machine learning*, 28(1):41–75, 1997.

F. Castellani and G. Franceschini. Use of genetic algorithms as an innovative tool for race car design. Technical report, SAE Technical Paper, 2003.

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

F. Cérou and A. Guyader. Adaptive multilevel splitting for rare event analysis. *Stochastic Analysis and Applications*, 25(2):417–443, 2007.

L. Chaari, J.-Y. Tourneret, C. Chaux, and H. Batatia. A hamiltonian monte carlo method for non-smooth energy sampling. *IEEE Transactions on Signal Processing*, 64(21):5585–5594, 2016.

M. Cheah, S. A. Shaikh, J. Bryans, and H. N. Nguyen. Combining third party components securely in automotive systems. In *IFIP International Conference on Information Security Theory and Practice*, pages 262–269. Springer, 2016.

M.-H. Chen, Q.-M. Shao, and J. G. Ibrahim. *Monte Carlo methods in Bayesian computation*. Springer Science & Business Media, 2012.

X. Chen, E. Ábrahám, and S. Sankaranarayanan. Flow*: An analyzer for non-linear hybrid systems. In *Computer Aided Verification*, pages 258–263. Springer, 2013.

Y. Chen, R. Dwivedi, M. J. Wainwright, and B. Yu. Fast mixing of metropolized hamiltonian monte carlo: Benefits of multi-step gradients. *arXiv preprint arXiv:1905.12247*, 2019.

N. Cheney, R. MacCurdy, J. Clune, and H. Lipson. Unshackling evolution: evolving soft robots with multiple materials and a powerful generative encoding. *ACM SIGEVOlution*, 7(1):11–23, 2014.

E. M. Clarke. The birth of model checking. In *25 Years of Model Checking*, pages 1–26. Springer, 2008.

N. Cohen, O. Sharir, and A. Shashua. On the expressive power of deep learning: A tensor analysis. In *Conference on Learning Theory*, pages 698–728. , 2016.

A. Corso, R. J. Moss, M. Koren, R. Lee, and M. J. Kochenderfer. A survey of algorithms for black-box safety validation. *arXiv preprint arXiv:2005.02979*, 2020.

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

E. Coumans and Y. Bai. Pybullet, a python module for physics simulation for games, robotics and machine learning. *GitHub repository*, 2016.

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

S. Dean, H. Mania, N. Matni, B. Recht, and S. Tu. Regret bounds for robust adaptive control of the linear quadratic regulator. In *Advances in Neural Information Processing Systems*, pages 4188–4197, 2018.

M. Deisenroth and C. E. Rasmussen. Pilco: A model-based and data-efficient approach to policy search. In *Proceedings of the 28th International Conference on machine learning (ICML-11)*, pages 465–472, 2011.

P. Del Moral. Feynman-kac formulae. In *Feynman-Kac Formulae*, pages 47–93. Springer, 2004.

P. Del Moral, A. Doucet, and A. Jasra. Sequential monte carlo samplers. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 68(3):411–436, 2006.

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

A. Donzé. Breach, a toolbox for verification and parameter synthesis of hybrid systems. In *International Conference on Computer Aided Verification*, pages 167–170. Springer, 2010.

A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun. CARLA: An open urban driving simulator. In *Proceedings of the 1st Annual Conference on Robot Learning*, pages 1–16, 2017a.

A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun. Carla: An open urban driving simulator. *arXiv preprint arXiv:1711.03938*, 2017b.

A. Doucet, N. De Freitas, and N. Gordon. An introduction to sequential monte carlo methods. In *Sequential Monte Carlo methods in practice*, pages 3–14. Springer, 2001.

J. Doyle, K. Glover, P. Khargonekar, and B. Francis. State-space solutions to standard h_2 and h_∞ control problems. In *1988 American Control Conference*, pages 1691–1696. IEEE, 1988.

J. C. Doyle, B. A. Francis, and A. R. Tannenbaum. *Feedback control theory*. Courier Corporation, 2013.

T. Dreossi, D. J. Fremont, S. Ghosh, E. Kim, H. Ravanbakhsh, M. Vazquez-Chanlatte, and S. A. Seshia. Verifai: A toolkit for the design and analysis of artificial intelligence-based systems. *arXiv preprint arXiv:1902.04245*, 2019.

S. Duane, A. D. Kennedy, B. J. Pendleton, and D. Roweth. Hybrid monte carlo. *Physics letters B*, 195(2):216–222, 1987.

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`.

J. C. Duchi, M. I. Jordan, M. J. Wainwright, and A. Wibisono. Optimal rates for zero-order convex optimization: The power of two function evaluations. *IEEE Transactions on Information Theory*, 61(5):2788–2806, 2015.

A. Durmus, E. Moulines, et al. Nonasymptotic convergence analysis for the unadjusted langevin algorithm. *The Annals of Applied Probability*, 27(3):1551–1587, 2017.

J. M. Esposito, J. Kim, and V. Kumar. Adaptive rrts for validating hybrid robotic control systems. In *Algorithmic Foundations of Robotics VI*, pages 107–121. Springer, 2004.

G. E. Fainekos and G. J. Pappas. Robustness of temporal logic specifications. In *Formal Approaches to Software Testing and Runtime Verification*, pages 178–192. Springer, 2006.

G. E. Fainekos and G. J. Pappas. Robustness of temporal logic specifications for continuous-time signals. *Theoretical Computer Science*, 410(42):4262–4291, 2009.

Federation Internationale de l'Automobile. Formula one 2019 results. `https://www.formula1.com/en/results.html/2019/`, 2019.

P. F. Felzenszwalb and D. P. Huttenlocher. Distance transforms of sampled functions. *Theory of computing*, 8(1):415–428, 2012.

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

C. Finn, K. Xu, and S. Levine. Probabilistic model-agnostic meta-learning. In *Advances in Neural Information Processing Systems*, pages 9516–9527, 2018.

A. Fishberg, T. Henderson, K. Gregson, Z. Dodds, D. Krawciw, C. Nicholls, and S. Karaman. Autonomous racecar: 1/10-scale autonomous car platform for research and education in robotics. In *2019 IEEE International Conference on Robotics and Automation (ICRA) Tutorial*. IEEE, 2019.

J. Friedman, T. Hastie, and R. Tibshirani. Sparse inverse covariance estimation with the graphical lasso. *Biostatistics*, 9(3):432–441, 2008.

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

Y. Gal and Z. Ghahramani. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *International Conference on Machine Learningearning*, pages 1050–1059, 2016.

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.

E. Games. Unreal engine 4 documentation. *URL https://docs. unrealengine. com/latest/INT/index. html*, 2015.

Y. Gao, A. Gray, H. E. Tseng, and F. Borrelli. A tube-based robust nonlinear predictive control approach to semiautonomous ground vehicles. *Vehicle System Dynamics*, 52(6): 802–823, 2014.

A. Gelman and X.-L. Meng. Simulating normalizing constants: From importance sampling to bridge sampling to path sampling. *Statistical science*, pages 163–185, 1998.

C. J. Geyer. Markov chain monte carlo maximum likelihood. 1991.

I. Gilboa and M. Marinacci. Ambiguity and the bayesian paradigm. In *Readings in formal epistemology*, pages 385–439. Springer, 2016.

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.

B. Goldfain, P. Drews, C. You, M. Barulic, O. Velev, P. Tsiotras, and J. M. Rehg. Autorally: An open platform for aggressive autonomous driving. *IEEE Control Systems Magazine*, 39(1):26–55, 2019.

J. Gonzales, U. Rosolia, and G. Marcil. Barc: Berkeley autonomous race car. URL `http://www.barc-project.com/`.

A. Graves. Practical variational inference for neural networks. In *Advances in Neural Information Processing Systems*, pages 2348–2356, 2011.

S. Gulwani, O. Polozov, R. Singh, et al. Program synthesis. *Foundations and Trends® in Programming Languages*, 4(1-2):1–119, 2017.

D. Ha and J. Schmidhuber. World models. *arXiv preprint arXiv:1803.10122*, 2018a.

D. Ha and J. Schmidhuber. Recurrent world models facilitate policy evolution. In *Advances in Neural Information Processing Systems 31*, pages 2451–2463. Curran Associates, Inc., 2018b. URL `https://papers.nips.cc/paper/7512-recurrent-world-models-facilitate-policy-evolution`. `https://worldmodels.github.io`.

K. Hacker, K. Lewis, and E. M. Kasprzak. Racecar optimization and tradeoff analysis in a parallel computing environment. Technical report, SAE Technical Paper, 2000.

J. M. Hammersley and D. C. Handscomb. Monte carlo methods. 1964.

N. Hansen. The CMA evolution strategy: A tutorial. *arXiv preprint arXiv:1604.00772*, 2016.

N. Hansen and A. Ostermeier. Completely derandomized self-adaptation in evolution strategies. *Evolutionary computation*, 9(2):159–195, 2001.

N. Hansen, S. D. Müller, and P. Koumoutsakos. Reducing the time complexity of the derandomized evolution strategy with covariance matrix adaptation (cma-es). *Evolutionary computation*, 11(1):1–18, 2003.

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

K. Hayward. *Application of Evolutionary Algorithms to Engineering Design.* University of Western Australia, 2007.

H. He, J. Boyd-Graber, K. Kwok, and H. Daumé III. Opponent modeling in deep reinforcement learning. In *International conference on machine learning*, pages 1804–1813, 2016.

H. Heinecke, K.-P. Schnelle, H. Fennel, J. Bortolazzi, L. Lundh, J. Leflour, J.-L. Maté, K. Nishikawa, and T. Scharnhorst. Automotive open system architecture-an industry-wide initiative to manage the complexity of emerging automotive e/e-architectures. Technical report, SAE Technical Paper, 2004.

P. Henderson, R. Islam, P. Bachman, J. Pineau, D. Precup, and D. Meger. Deep reinforcement learning that matters. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

T. A. Henzinger, P. W. Kopke, A. Puri, and P. Varaiya. What's decidable about hybrid automata? In *Journal of Computer and System Sciences*, pages 373–382. ACM Press, 1995.

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.

T. C. Hesterberg and B. L. Nelson. Control variates for probability and quantile estimation. *Management Science*, 44(9):1295–1312, 1998.

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

J. Ho and S. Ermon. Generative adversarial imitation learning. In *Advances in Neural Information Processing Systems*, pages 4565–4573, 2016.

M. Hoffman, P. Sountsov, J. V. Dillon, I. Langmore, D. Tran, and S. Vasudevan. Neutralizing bad geometry in hamiltonian monte carlo using neural transport. *arXiv preprint arXiv:1903.03704*, 2019.

T. Homem-de Mello. A study on the cross-entropy method for rare-event probability estimation. *INFORMS Journal on Computing*, 19(3):381–394, 2007.

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

J. Hu and P. Hu. On the performance of the cross-entropy method. In *Simulation Conference (WSC), Proceedings of the 2009 Winter*, pages 459–468. IEEE, 2009.

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.

J. Hu, P. Hu, and H. S. Chang. A stochastic approximation framework for a class of randomized optimization algorithms. *IEEE Transactions on Automatic Control*, 57(1): 165–178, 2012.

J. Hu, E. Zhou, and Q. Fan. Model-based annealing random search with stochastic averaging. *ACM Transactions on Modeling and Computer Simulation (TOMACS)*, 24(4):21, 2014.

X. Huang, M. Kwiatkowska, S. Wang, and M. Wu. Safety verification of deep neural networks. In *International Conference on Computer Aided Verification*, pages 3–29. Springer, 2017.

L. R. Humphrey, E. M. Wolff, and U. Topcu. Formal specification and synthesis of mission plans for unmanned aerial vehicles. In *AAAI Spring Symposium Series*, 2014.

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

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.

M. Jaderberg, W. M. Czarnecki, I. Dunning, L. Marris, G. Lever, A. G. Castaneda, C. Beattie, N. C. Rabinowitz, A. S. Morcos, A. Ruderman, et al. Human-level performance in 3d multiplayer games with population-based reinforcement learning. *Science*, 364(6443): 859–865, 2019.

A. Jain and M. Morari. Computing the racing line using bayesian optimization. *arXiv preprint arXiv:2002.04794*, 2020.

M. R. Jardin and E. R. Mueller. Optimized measurements of unmanned-air-vehicle mass moment of inertia with a bifilar pendulum. *Journal of Aircraft*, 46(3):763–775, 2009.

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.

H. Kahn and T. Harris. Estimation of particle transmission by random sampling. 1951.

N. Kalra and S. M. Paddock. Driving to safety: How many miles of driving would it take to demonstrate autonomous vehicle reliability? *Transportation Research Part A: Policy and Practice*, 94:182–193, 2016.

N. R. Kapania, J. Subosits, and J. C. Gerdes. A sequential two-step algorithm for fast generation of vehicle racing trajectories. *Journal of Dynamic Systems, Measurement, and Control*, 138(9):091005, 2016.

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

E. M. Kasprzak, K. E. Lewis, and D. L. Milliken. Steady-state vehicle optimization using pareto-minimum analysis. *SAE transactions*, pages 2624–2631, 1998.

S. Kato, S. Tokunaga, Y. Maruyama, S. Maeda, M. Hirabayashi, Y. Kitsukawa, A. Monrroy, T. Ando, Y. Fujii, and T. Azumi. Autoware on board: Enabling autonomous vehicles with embedded systems. In *2018 ACM/IEEE 9th International Conference on Cyber-Physical Systems (ICCPS)*, pages 287–296. IEEE, 2018.

G. Katz, C. Barrett, D. Dill, K. Julian, and M. Kochenderfer. Reluplex: An efficient smt solver for verifying deep neural networks. *arXiv:1702.01135 [cs.AI]*, 1:1, 2017.

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.

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

D. P. Kingma, T. Salimans, and M. Welling. Variational dropout and the local reparameterization trick. In *Advances in Neural Information Processing Systems*, pages 2575–2583, 2015.

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.

B. R. Kiran, I. Sobh, V. Talpaert, P. Mannion, A. A. A. Sallab, S. Yogamani, and P. Pérez. Deep reinforcement learning for autonomous driving: A survey. *arXiv preprint arXiv:2002.00444*, 2020.

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

O. Klimov. Carracing-v0. 2016. *URL https://gym. openai. com/envs/CarRacing-v0*, 2016.

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

S. Kong, S. Gao, W. Chen, and E. Clarke. dreach: δ-reachability analysis for hybrid systems. In *International Conference on TOOLS and Algorithms for the Construction and Analysis of Systems*, pages 200–205. Springer, 2015a.

S. Kong, S. Gao, W. Chen, and E. M. Clarke. dreach: Delta-reachability analysis for hybrid systems. In *Tools and Algorithms for the Construction and Analysis of Systems - 21st International Conference, TACAS 2015*, pages 200–205, 2015b.

A. Kotikalpudi, B. Taylor, C. Moreno, H. Pfifer, and G. Balas. Swing tests for estimation of moments of inertia. *Unpublished notes, University of Minnesota Dept. of Aerospace Engineering and Mechanics*, 2013.

R. Koymans. Specifying real-time properties with metric temporal logic. *Real-Time Systems*, 2(4):255–299, 1990.

D. P. Kroese, R. Y. Rubinstein, and P. W. Glynn. The cross-entropy method for estimation. *Handbook of Statistics: Machine Learning: Theory and Applications*, 31:19–34, 2013.

A. Kuefler, J. Morton, T. Wheeler, and M. Kochenderfer. Imitating driver behavior with generative adversarial networks. In *2017 IEEE Intelligent Vehicles Symposium (IV)*, pages 204–211. IEEE, 2017.

A. Kulesza, B. Taskar, et al. Determinantal point processes for machine learning. *Foundations and Trends® in Machine Learning*, 5(2–3):123–286, 2012.

P. R. Kumar. A survey of some results in stochastic adaptive control. *SIAM Journal on Control and Optimization*, 23(3):329–380, 1985.

D. Kundu. *Subjective and objective quality evaluation of synthetic and high dynamic range images*. PhD thesis, 2016.

M. Kwiatkowska, G. Norman, and D. Parker. Prism 4.0: Verification of probabilistic real-time systems. In *International conference on computer aided verification*, pages 585–591. Springer, 2011.

S. Lan, B. Zhou, and B. Shahbaba. Spherical hamiltonian monte carlo for constrained target distributions. In *JMLR workshop and conference proceedings*, volume 32, page 629. NIH Public Access, 2014.

C. A. Lattner. *LLVM: An infrastructure for multi-stage optimization*. PhD thesis, University of Illinois at Urbana-Champaign, 2002.

S. M. LaValle. Rapidly-exploring random trees: A new tool for path planning. 1998.

B. Leimkuhler and S. Reich. *Simulating Hamiltonian Dynamics*, volume 14. Cambridge University Press, 2004.

K. Leung, N. Arechiga, and M. Pavone. Back-propagation through stl specifications: Infusing logical structure into gradient-based methods.

I. Lewis, W. Cannon, M. Moll, and L. E. Kavraki. How much do unstated problem constraints limit deep robotic reinforcement learning? *arXiv preprint arXiv:1909.09282*, 2019.

LG Electronics, Inc. Lgsvl simulator, 2019. URL `https://www.lgsvlsimulator.com/`.

P. Liang, M. I. Jordan, and D. Klein. Learning programs: A hierarchical bayesian approach. In *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, pages 639–646, 2010.

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

A. Liniger, A. Domahidi, and M. Morari. Optimization-based autonomous racing of 1: 43 scale rc cars. *Optimal Control Applications and Methods*, 36(5):628–647, 2015.

T. Lipp and S. Boyd. Minimum-time speed optimisation over a fixed path. *International Journal of Control*, 87(6):1297–1311, 2014.

J. Liu, N. Ozay, U. Topcu, and R. M. Murray. Synthesis of reactive switching protocols from temporal logic specifications. *IEEE Transactions on Automatic Control*, 58(7):1771–1785, 2013.

S. M. Loos, A. Platzer, and L. Nistor. Adaptive cruise control: Hybrid, distributed, and now formally verified. In *International Symposium on Formal Methods*, pages 42–56. Springer, 2011.

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

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

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

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

W. Luo, B. Yang, and R. Urtasun. Fast and furious: Real time end-to-end 3d detection, tracking and motion forecasting with a single convolutional net. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 3569–3577, 2018.

J. Lygeros. Lecture notes on hybrid systems. In *Notes for an ENSIETA workshop*, 2004.

A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu. Towards deep learning models resistant to adversarial attacks. *arXiv preprint arXiv:1706.06083*, 2017.

A. Majumdar and R. Tedrake. Robust online motion planning with regions of finite time invariance. In *Algorithmic foundations of robotics X*, pages 543–558. Springer, 2013.

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.

R. Mangharam, M. Behl, H. Abbas, M. Bertonga, and M. O'Kelly. 2018 Italian Grand Prix, 2018a.

R. Mangharam, M. Behl, H. Abbas, and M. O'Kelly. 2018 Portuguese Grand Prix, 2018b.

R. Mangharam, M. Behl, H. Abbas, M. Bertonga, and M. O'Kelly. 2019 New York City Grand Prix, 2019a.

R. Mangharam, M. Behl, H. Abbas, and M. O'Kelly. 2019 Canadian Grand Prix, 2019b.

E. Marinari and G. Parisi. Simulated tempering: a new monte carlo scheme. *EPL (Europhysics Letters)*, 19(6):451, 1992.

A. W. Marshall. The use of multi-stage sampling schemes in monte carlo computations. Technical report, RAND CORP SANTA MONICA CALIF, 1954.

H. Massalin. Superoptimizer: a look at the smallest program. In *ACM SIGARCH Computer Architecture News*, volume 15, pages 122–126. IEEE Computer Society Press, 1987.

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

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

X.-L. Meng and S. Schilling. Warp bridge sampling. *Journal of Computational and Graphical Statistics*, 11(3):552–586, 2002.

X.-L. Meng and W. H. Wong. Simulating ratios of normalizing constants via a simple identity: a theoretical exploration. *Statistica Sinica*, pages 831–860, 1996.

J.-B. Mouret and J. Clune. Illuminating search spaces by mapping elites. *arXiv preprint arXiv:1504.04909*, 2015.

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

S. Nakamoto and A. Bitcoin. A peer-to-peer electronic cash system. *Bitcoin.–URL: https://bitcoin. org/bitcoin. pdf*, 2008.

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

R. M. Neal. Annealed importance sampling. *Statistics and computing*, 11(2):125–139, 2001.

R. M. Neal. Estimating ratios of normalizing constants using linked importance sampling. *arXiv preprint math/0511216*, 2005.

R. M. Neal. Mcmc using hamiltonian dynamics. *arXiv preprint arXiv:1206.1901*, 2012.

NHTSA. Federal automated vehicles policy, September 2016.

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

P. Nilsson, O. Hussien, A. Balkan, Y. Chen, A. D. Ames, J. W. Grizzle, N. Ozay, H. Peng, and P. Tabuada. Correct-by-construction adaptive cruise control: Two approaches.

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

M.-S. Oh and J. O. Berger. Adaptive importance sampling in monte carlo integration. *Journal of Statistical Computation and Simulation*, 41(3-4):143–168, 1992.

M. O'Kelly, H. Abbas, S. Gao, S. Kato, S. Shiraishi, and R. Mangharam. Apex: Autonomous vehicle plan verification and execution. In *SAE Technical Paper*. SAE International, 04 2016.

M. O'Kelly, A. Sinha, H. Namkoong, R. Tedrake, and J. C. Duchi. Scalable end-to-end autonomous vehicle testing via rare-event simulation. In *Advances in Neural Information Processing Systems*, pages 9827–9838, 2018.

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/`.

I. Osband, C. Blundell, A. Pritzel, and B. Van Roy. Deep exploration via bootstrapped dqn. In *Advances in neural information processing systems*, pages 4026–4034, 2016.

A. Pakman and L. Paninski. Auxiliary-variable exact hamiltonian monte carlo samplers for binary distributions. In *Advances in neural information processing systems*, pages 2490–2498, 2013.

A. Pakman and L. Paninski. Exact hamiltonian monte carlo for truncated multivariate gaussians. *Journal of Computational and Graphical Statistics*, 23(2):518–542, 2014.

Y. V. Pant, H. Abbas, and R. Mangharam. Smooth operator: Control using the smooth robustness of temporal logic. In *2017 IEEE Conference on Control Technology and Applications (CCTA)*, pages 1235–1240. IEEE, 2017.

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

G. Papamakarios, E. Nalisnick, D. J. Rezende, S. Mohamed, and B. Lakshminarayanan. Normalizing flows for probabilistic modeling and inference. *arXiv preprint arXiv:1912.02762*, 2019.

T. P. Pavlic, P. A. Sivilotti, A. D. Weide, and B. W. Weide. Comments on adaptive cruise control: hybrid, distributed, and now formally verified. *OSU CSE Dept TR22*, 2011.

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.

N. Piterman, A. Pnueli, and Y. Sa'ar. Synthesis of reactive (1) designs. In *International Workshop on Verification, Model Checking, and Abstract Interpretation*, pages 364–380. Springer, 2006.

X. Qin, N. Aréchiga, A. Best, and J. Deshmukh. Automatic testing and falsification with dynamically constrained reinforcement learning. *arXiv preprint arXiv:1910.13645*, 2019.

W. Qiu, F. Zhong, Y. Zhang, S. Qiao, Z. Xiao, T. S. Kim, Y. Wang, and A. Yuille. Unrealcv: Virtual worlds for computer vision. *ACM Multimedia Open Source Software Competition*, 2017.

C. Quiter and M. Ernst. Deepdrive, 2018. URL `https://deepdrive.voyage.auto/`.

J. Ragan-Kelley, C. Barnes, A. Adams, S. Paris, F. Durand, and S. Amarasinghe. Halide: a language and compiler for optimizing parallelism, locality, and recomputation in image processing pipelines. In *Acm Sigplan Notices*, volume 48, pages 519–530. ACM, 2013.

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.

G. O. Roberts and O. Stramer. Langevin diffusions and metropolis-hastings algorithms. *Methodology and computing in applied probability*, 4(4):337–357, 2002.

N. Roohi, R. Kaur, J. Weimer, O. Sokolsky, and I. Lee. Self-driving vehicle verification towards a benchmark. *arXiv preprint arXiv:1806.08810*, 2018.

W. Roscoe. Donkey car: An opensource diy self driving platform for small scale cars, 2019.

U. Rosolia and F. Borrelli. Learning how to autonomously race a car: a predictive control approach. *IEEE Transactions on Control Systems Technology*, 2019.

S. Ross and D. Bagnell. Efficient reductions for imitation learning. In *Proceedings of the* *thirteenth international conference on artificial intelligence and statistics*, pages 661–668, 2010.

S. Ross, G. Gordon, and D. Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pages 627–635, 2011.

P. J. Rossky, J. Doll, and H. Friedman. Brownian dynamics as smart monte carlo simulation. *The Journal of Chemical Physics*, 69(10):4628–4633, 1978.

R. Y. Rubinstein. Combinatorial optimization, cross-entropy, ants and rare events. In *Stochastic optimization: algorithms and applications*, pages 303–363. Springer, 2001.

R. Y. Rubinstein and D. P. Kroese. *The cross-entropy method: A unified approach to Monte Carlo simulation, randomized optimization and machine learning.* Information Science & Statistics, Springer Verlag, NY, 2004.

R. Y. Rubinstein and R. Marcus. Efficiency of multivariate control variates in monte carlo simulation. *Operations Research*, 33(3):661–677, 1985.

S. Russell. Learning agents for uncertain environments. In *Proceedings of the eleventh annual conference on Computational learning theory*, pages 101–103. ACM, 1998.

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.

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.

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

E. Schkufza, R. Sharma, and A. Aiken. Stochastic superoptimization. In *ACM SIGPLAN Notices*, volume 48, pages 305–316. ACM, 2013.

R. Schram, A. Williams, and M. van Ratingen. Implementation of autonomous emergency braking (aeb), the next step in euro ncap's safety assessment. *ESV, Seoul*, 2013.

S. A. Seshia, D. Sadigh, and S. S. Sastry. Formal methods for semi-autonomous driving. In *Proceedings of the 52nd Annual Design Automation Conference*, page 148. ACM, 2015.

V. Sezer and M. Gokasan. A novel obstacle avoidance algorithm:"follow the gap method". *Robotics and Autonomous Systems*, 60(9):1123–1134, 2012.

S. Shah, D. Dey, C. Lovett, and A. Kapoor. Airsim: High-fidelity visual and physical simulation for autonomous vehicles. In *Field and Service Robotics*, 2017. URL `https://arxiv.org/abs/1705.05065`.

B. Shahbaba, S. Lan, W. O. Johnson, and R. M. Neal. Split hamiltonian monte carlo. *Statistics and Computing*, 24(3):339–349, 2014.

S. Shalev-Shwartz, S. Shammah, and A. Shashua. On a formal model of safe and scalable self-driving cars. *arXiv preprint arXiv:1708.06374*, 2017.

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

D. Siegmund. Importance sampling in the monte carlo study of sequential tests. *The Annals of Statistics*, pages 673–684, 1976.

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.

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

A. Sinha, H. Namkoong, and J. Duchi. Certifying some distributional robustness with principled adversarial training. *arXiv preprint arXiv:1710.10571*, 2017.

A. Sinha, M. O'Kelly, R. Tedrake, and J. C. Duchi. Neural bridge sampling for evaluating safety-critical autonomous systems. *Advances in Neural Information Processing Systems*, 33, 2020.

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

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

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

A. Solar-Lezama. Program sketching. *International Journal on Software Tools for Technology Transfer*, 15(5-6):475–495, 2013.

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.

H. A. Soule and M. P. Miller. The experimental determination of the moments of inertia of airplanes. 1934.

R. Sparrow and M. Howard. When human beings are like drunk robots: Driverless vehicles, ethics, and the future of transport. *Transportation Research Part C: Emerging Technologies*, 80:206–215, 2017.

J. Spurr, S. Goodwill, J. Kelley, and S. Haake. Measuring the inertial properties of a tennis racket. *Procedia Engineering*, 72:569–574, 2014.

S. S. Srinivasa, P. Lancaster, J. Michalove, M. Schmittle, C. Summers, M. Rockett, J. R. Smith, S. Chouhury, C. Mavrogiannis, and F. Sadeghi. MuSHR: A low-cost, open-source robotic racecar for education and research. *CoRR*, abs/1908.08031, 2019.

R. S. Sutton and A. G. Barto. *Reinforcement learning: An introduction.* MIT press, 2018.

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

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

Y. Tang, D. Nguyen, and D. Ha. Neuroevolution of self-interpretable agents. *arXiv preprint arXiv:2003.08165*, 2020.

V. Tjeng and R. Tedrake. Verifying neural networks with mixed integer programming. *arXiv:1711.07356 [cs.LG]*, 2017.

J. Tobin, R. Fong, A. Ray, J. Schneider, W. Zaremba, and P. Abbeel. Domain randomization for transferring deep neural networks from simulation to the real world. In *2017 IEEE/RSJ international conference on intelligent robots and systems (IROS)*, pages 23–30. IEEE, 2017.

E. Todorov, T. Erez, and Y. Tassa. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5026–5033. IEEE, 2012.

C. E. Tuncali, T. P. Pavlic, and G. Fainekos. Utilizing s-taliro as an automatic test generation framework for autonomous vehicles. In *Intelligent Transportation Systems (ITSC), 2016 IEEE 19th International Conference on*, pages 1470–1475. IEEE, 2016.

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.

C. Urmson, J. Anhalt, D. Bagnell, C. Baker, R. Bittner, M. Clark, J. Dolan, D. Duggins, T. Galatali, C. Geyer, et al. Autonomous driving in urban environments: Boss and the urban challenge. *Journal of Field Robotics*, 25(8):425–466, 2008.

US Department of Transportation – FHWA. Ngsim – next generation simulation, 2008.

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.

N. Vasilache, O. Zinenko, T. Theodoridis, P. Goyal, Z. DeVito, W. S. Moses, S. Verdoolaege, A. Adams, and A. Cohen. Tensor comprehensions: Framework-agnostic high-performance machine learning abstractions. *arXiv preprint arXiv:1802.04730*, 2018.

B. Vedder. Vedder electronic speed controller, 2015. URL `https://vesc-project.com/documentation`.

D. Verscheure, M. Diehl, J. De Schutter, and J. Swevers. Recursive log-barrier method for on-line time-optimal robot path tracking. In *Prceedings of the 2009 American Control Conference*, pages 4134–4140. IEEE, 2009.

G. Vinnicombe. Frequency domain uncertainty and the graph topology. *IEEE Transactions on Automatic Control*, 38(9):1371–1383, 1993.

K. Vogel. A comparison of headway and time to collision as safety indicators. *Accident analysis & prevention*, 35(3):427–433, 2003.

A. F. Voter. A monte carlo method for determining free-energy differences and transition state theory rate constants. *The Journal of chemical physics*, 82(4):1890–1899, 1985.

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`.

D. Wang, C. Devin, Q.-Z. Cai, P. Krähenbühl, and T. Darrell. Monocular plan view networks for autonomous driving. *arXiv preprint arXiv:1905.06937*, 2019a.

Q. Wang, P. Zuliani, S. Kong, S. Gao, and E. M. Clarke. Sreach: A probabilistic bounded delta-reachability analyzer for stochastic hybrid systems. In *International Conference on Computational Methods in Systems Biology*, pages 15–27. Springer, 2015.

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

S. Webb, T. Rainforth, Y. W. Teh, and M. P. Kumar. A statistical approach to assessing neural network robustness. 2018.

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.

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.

R. J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3-4):229–256, 1992.

J. M. Wing. Trustworthy ai. *arXiv preprint arXiv:2002.06276*, 2020.

S. Yaghoubi and G. Fainekos. Falsification of temporal logic requirements using gradient based local search in space and time. *IFAC-PapersOnLine*, 51(16):103–108, 2018.

B. Yang, P. Lancaster, S. Srinivasa, and J. R. Smith. Benchmarking robot manipulation with the rubik's cube. *IEEE Robotics and Automation Letters*, 2020.

Z. B. Zabinsky. *Stochastic adaptive search for global optimization*, volume 72. Springer Science & Business Media, 2013.

D. Zhao. *Accelerated Evaluation of Automated Vehicles.* Ph.D. thesis, Department of Mechanical Engineering, University of Michigan, 2016.

D. Zhao, X. Huang, H. Peng, H. Lam, and D. J. LeBlanc. Accelerated evaluation of automated vehicles in car-following maneuvers. *IEEE Transactions on Intelligent Transportation Systems*, 19(3):733–744, 2018.

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

E. Zhou and J. Hu. Gradient-based adaptive stochastic search for non-differentiable optimization. *IEEE Transactions on Automatic Control*, 59(7):1818–1832, 2014.

K. Zhou, J. C. Doyle, and K. Glover. Robust and optimal control. 1996.

A. Zutshi, J. V. Deshmukh, S. Sankaranarayanan, and J. Kapinski. Multiple shooting, cegar-based falsification for hybrid systems. In *Proceedings of the 14th International Conference on Embedded Software*, pages 1–10, 2014.