This article has been accepted for publication in IEEE Robotics and Automation Letters. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/LRA.2024.3455782

Learning-based Bayesian Inference for Testing of Autonomous Systems

Anjali Parashar[1], Ji Yin[2], Charles Dawson[1], Panagiotis Tsiotras[2], and Chuchu Fan[1]

Abstract—**For the safe operation of robotic systems, it is important to accurately understand its failure modes using prior testing. Hardware testing of robotic infrastructure is known to be slow and costly. Instead, failure prediction in simulation can help to analyze the system before deployment. Conventionally, large-scale naïve Monte Carlo simulations are used for testing; however, this method is only suitable for testing average system performance. For safety-critical systems, worst-case performance is more crucial as failures are often rare events, and the size of test batches increases substantially as failures become rarer. Rare-event sampling methods can be helpful; however, they exhibit slow convergence and cannot handle constraints. This research introduces a novel sampling-based testing framework for autonomous systems which bridges these gaps by utilizing a discretized gradient-based second-order Langevin algorithm combined with learning-based techniques for constrained sampling of failure modes. Our method can predict more diverse failures by exploring the search space efficiently and ensures feasibility with respect to temporal and implicit constraints. We demonstrate the use of our testing methodology on two categories of testing problems, via simulations and hardware experiments. Our method discovers up to 2X failures compared to naïve Random Walk sampling, with only half of the sample size.**

Index Terms—**Robot Safety, Probabilistic Inference, Formal Methods in Robotics and Automation**

I. INTRODUCTION

TESTING, verification, and model validation are integral to ensuring the safety of robotic systems. Discovering possible failure modes of the system through testing is challenging for several reasons. First, the underlying mathematical problem of failure discovery is highly non-convex with an unpredictable loss landscape. Second, sufficient exploration of the search space is important, as there are multiple failure modes of varying probability of occurrence. To ensure proper coverage of the state space, state-of-the-art methods for testing, such as Random Walk MCMC sampling [1], need a significant amount of computational resources, and can be extremely time-consuming [2].

In the past few years, some new favorable alternatives have emerged. Primarily, there are two ways to approach the problem of failure discovery, given a dynamic system and a known environment. The first set of methods relies on adversarial optimization [3], where the key challenge is that it typically renders one failure case per search. Additionally, initialization plays a key role in the implementation of gradient-based methods for non-convex optimization [4], therefore, inappropriately chosen initial conditions are likely to converge to a few high-probability failure modes, thereby completely missing out on some rare failures that are equally detrimental to the safety of the system.

The second approach utilizes recent advancements in machine learning, such as generative modeling, to learn failure representations by using failure data as input [5]. A key challenge with this approach is the lack of generalization guarantees of neural network-based models for out-of-distribution data, causing the generated failure instances to be biased towards the failures observed in the training dataset.

Some works have recently analyzed failure discovery and model validation under the lens of Bayesian inference [6], [7], which combines the convergence properties of adversarial optimization and the search-space exploration capabilities of sampling-based methods. While these methods do not suffer from initialization limitations of deterministic optimization, they typically carry over some of the drawbacks of sampling methods, such as uneven search space exploration, curse of dimensionality, and slow convergence.

In this paper, we study the question — what is the most efficient way of testing a known dynamic system with known environmental interactions? The objective of this work is to discover the failure modes of a given dynamic system with a known controller, where the focus is on failures caused due to environmental interactions (Fig. 1). We exploit the awareness of testing scenarios to discover failure modes in simulation, with an aim to reduce the dependency on field testing as much as possible. Building upon the fundamental structure of Bayesian inference, we provide a modular approach to testing via learning-based stochastic optimization.

A. Key Contributions

This paper introduces a novel testing framework for control and planning algorithms for robotic systems by formulating

Manuscript received: April 16, 2024; Revised: July 13, 2024; Accepted: August 11, 2024.

This paper was recommended for publication by Editor Aniket Bera upon evaluation of the Associate Editor and Reviewers' comments. This work was partly supported by the National Aeronautics and Space Administration (NASA) ULI grant 80NSSC22M0070 and Air Force Office of Scientific Research (AFOSR) grant FA9550-23-1-0099. Any opinions, findings, and conclusions or recommendations expressed in this publication are those of the authors and don't necessarily reflect the views of the sponsors.

[1]Anjali Parashar, Charles Dawson and Chuchu Fan are with the Laboratory for Information & Decision Systems (LIDS), Massachusetts Institute of Technology, Cambridge, MA 02139 USA (email: anjalip@mit.edu; cbd@mit.edu; chuchu@mit.edu).

[2] Ji Yin and Panagiotis Tsiotras are with the D. Guggenheim School of Aerospace Engineering, Georgia Institute of Technology, Atlanta, GA 30332 USA (email: jyin81@gatech.edu; tsiotras@gatech.edu).

Digital Object Identifier (DOI): see top of this page.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on September 08,2024 at 03:28:25 UTC from IEEE Xplore. Restrictions apply.
© 2024 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.

This article has been accepted for publication in IEEE Robotics and Automation Letters. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/LRA.2024.3455782

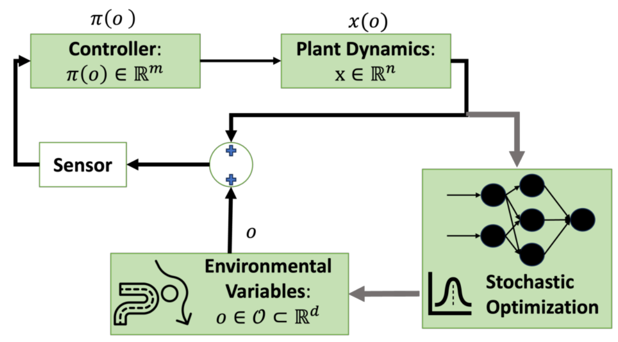

$\pi(o)$
Controller: $\pi(o) \in \mathbb{R}^m$
$x(o)$
Plant Dynamics: $x \in \mathbb{R}^n$
Sensor
o
Environmental Variables: $o \in \mathcal{O} \subset \mathbb{R}^d$
Stochastic Optimization

Fig. 1: Our approach utilizes learning-based priors within a Bayesian inference framework for sampling environment variables using gradient-based Langevin sampling for falsification of robotic systems

testing as a Bayesian inference problem (Section-II). One of the main objectives of this study is to extract meaningful information about the system without necessitating a heavy expenditure on the testing budget. A key contribution of this work is the proposal of learning-based methods for reducing the volume and dimensionality of prior distributions, promoting better performance for Bayesian inference techniques. Additionally, our framework can efficiently handle constrained stochastic optimization (Section-IV).

To address the convergence gap of sampling-based techniques in Bayesian inference, we exploit system awareness and leverage differentiability of controller and dynamics for gradient-based sampling [6]. We make use of the Underdamped Langevin Algorithm [8] (also known as 2nd-order LA), which provides accelerated convergence guarantees and better coverage, leading to the discovery of diverse failure modes. In Section-III-A we provide a theoretical intuition for the same. In Section-V we present three case studies of falsification of dynamic systems with different control algorithms along with experimental validation for analyzing the sim-to-real transfer.

II. Problem Statement

We consider continuous-time closed-loop dynamic systems of the form $\dot{x} = f(x, \pi(x, o))$ for given state $x \in \mathbb{R}^n$, where $a = \pi(x, o) \in \mathbb{R}^m$ is a known controller which outputs actions $a \in \mathcal{A} \subseteq \mathbb{R}^m$ based on system state and environmental observations $o \in \mathcal{O} \subset \mathbb{R}^d$ in a fully observable setting. In the rest of the paper, we refer to o as Environment Variables (EVs). The testing architecture takes a user-defined property as an input (such as "obstacle avoidance") for which we aim to find compelling failure cases that reveal loopholes in the controller performance and underlying limitations of our dynamic system.

For a given user-defined property, we design a cost function $c(o, X)$ where $X = (x_i)_{i=1}^T$ such that a failure of the underlying dynamic system in the presence of a chosen controller can be defined as the corresponding cost function exceeding a threshold c^*. That is, $c(o, X) > c^*$ corresponds to the violation of the desired property. The testing problem can then be mathematically formulated as

$$\text{find } \mathcal{O}_{\text{fail}} = \{o | c(o, X) > c^*\}. \tag{1}$$

Note that by definition, c^* is specific to the desired task and the controller, as the chosen controller affects the severity of failures observed. An appropriate selection of c^* requires user-inputs and fine-tuning to ensure that the user expectation aligns with the corresponding severity of failures observed.

III. Approach: Optimization via Sampling

We employ Bayesian inference to solve (1), as it can discover more than one failure per search. This is done by constructing a Bayesian inference problem that is equivalent to (1) using o as a decision variable. This can be understood as sampling from the conditional distribution of o that triggers failures, that is,

$$o \sim p(o | c(o, X) > c^*). \tag{2}$$

Failures are low-probability events, consequently, regions representing $p(o|c(o, X))$ are extremely low-density regions. To facilitate efficient sampling from these regions, we construct a posterior distribution $p(c(o, X) > c^*|o)$ that is highly biased towards failures, given by:

$$p(c(o, X) > c^*|o) \propto \exp(-\gamma[c^* - c]_+) \tag{3}$$

Here $[\cdot]_+$ represents the ReLU operator and assigns equal cost to every failure mode with $c > c^*$ and γ corresponds to the learning rate of the Langevin dynamics leveraged for solving this problem, explained in detail in Section-III-A. This construction ensures that the target density from which we wish to sample, p^* is indeed a solution to (1) and contains all failure modes regardless of their severity [6]. Using Bayes rule, the inference problem can therefore be written as:

$$p(o|c(o, X) > c^*) \propto p(o) \exp(-\gamma[c^* - c]_+). \tag{4}$$

Here, $p(o)$ represents the prior belief. In this work, we utilize spatio-temporal constraints imposed on o to inform the design of prior using machine-learning-based methods, discussed in Section-IV. This ensures that the constraints are always satisfied. In the next subsection, we provide insights on how to sample from the distribution $p \propto \exp(-\gamma[c^* - c]_+)$ using the 2nd-order Langevin algorithm (LA).

A. Stochastic Optimization via Underdamped Langevin

Recently, reference [6] demonstrated that for end-to-end differentiable systems, gradient-based Langevin Monte Carlo (also known as 1st-order LA) is a helpful tool for solving problems that can be written as in (4). Langevin dynamics (LD) based sampling tools have emerged as a promising alternative to gradient-free MCMC algorithms such as Random walk MCMC, which are popularly used in verification and falsification tools for stochastic optimization [1].

Consider the problem of sampling from a target distribution p^* of the form $p^*(o) \propto \exp(-U(o^*))$, where $o^* = \operatorname{argmin} U(o)$ for a differentiable function $U(o)$. The 1st-order Langevin Algorithm (LA) approaches this problem by constructing gradient-based iterates corresponding to a first-order Stochastic Differential Equation (SDE) given by:

$$\mathrm{d}o_t = -\gamma \nabla U(o_t)\, \mathrm{d}t + \sqrt{2\gamma}\, \mathrm{d}B_t, \tag{5}$$

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on September 08,2024 at 03:28:25 UTC from IEEE Xplore. Restrictions apply.
© 2024 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.

This article has been accepted for publication in IEEE Robotics and Automation Letters. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/LRA.2024.3455782

where B_t denotes the standard Brownian motion and γ corresponds to the learning rate.

Note that the Langevin diffusion process (5) has an invariant measure $p_\infty(o)$ which can be expressed as $p_\infty(o) \propto \exp(-\gamma U)$. Therefore, by modeling the posterior distribution as in (3), we guarantee that the distribution we are sampling from does not change along the trajectory of o_t. This technique has been used in several Bayesian inference based problem formulations [6], [7].

In multimodal distributions, a sufficient amount of global search space exploration is extremely important to discover all possible solutions. Additionally, speed of convergence to the optimum is also crucial. In [9] it was shown that accelerated-gradient-based methods such as Nesterov's method [10], are able to locate the optimal solution with faster speed of convergence for non-convex optimization problems, compared to first-order gradient-based methods. We hypothesize that the Underdamped Langevin Algorithm (also known as 2^{nd}-order LA) [8] performs better than 1^{st}-order LA and Random Walk MCMC in discovering diverse failure modes and converges faster.

The second-order SDE corresponding to 2^{nd}-order LA can be written as

$$\begin{aligned} \mathrm{d}o_t &= -r_t\,\mathrm{d}t, \\ \mathrm{d}r_t &= -\nabla U(o_t)\,\mathrm{d}t - \gamma r_t\,\mathrm{d}t + \sqrt{2\gamma}\,\mathrm{d}B_t. \end{aligned} \tag{6}$$

Note that the momentum variable r_t is also present in accelerated gradient descent and it helps avoid getting stuck in local minima. We hypothesize that the 2^{nd}-order LA can provide better search space exploration for a similar reason, and thus identify diverse failure modes. Recently, it was shown that non-reversible SDEs such as 2^{nd}-order LA search for global optimum at a much faster timescale [11]. Additionally [8] shows that the 2^{nd}-order LA converges to ϵ-accuracy of the global minimum in $\sqrt{(d/\epsilon)}$ steps (as opposed to d/ϵ steps needed for the 1^{st}-order LA [12]), under certain assumptions. This shows that for high-dimensional non-convex optimization, 2^{nd}-order LA is a promising approach, ensuring speed, sufficient search space exploration, and scalability. Furthermore, [6, Theorem 5.1] shows that using an appropriate cost design $c(\cdot)$, the problem of discovering failure modes as in (4) meets these assumptions. Therefore, we can utilize the convergence guarantees enjoyed by the 2^{nd}-order LA in our framework.

For this work, we adopt the discretization of Underdamped Langevin proposed in [8], where each step can be defined as sampling from a normal distribution with mean $\mu(o)$ and covariance Σ introduced in [8, Algorithm-1], namely,

$$o_j \sim \mathcal{N}(\mu(o_{j-1}), \Sigma). \tag{7}$$

IV. Prior design

For testing purposes, the region of failures are low-density distributions and naïvely chosen priors such as uniform distribution can severely over-estimate the likelihood of failure that impacts the efficiency of sampling, as a lot more steps are needed to converge to the true distribution. By having stronger prior estimates, we can significantly improve the efficiency of sampling. We reframe the prior distribution as constraints on the EVs, and create learned representations of EVs using available information about the environmental variable o and its interaction with the system. Using this method, we can simultaneously construct a strong prior and at the same time satisfy hard constraints imposed on the EVs.

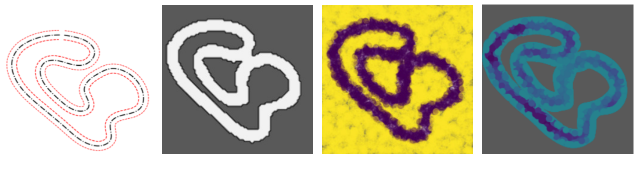


Fig. 2: Neural Projection for the AutoRally racetrack. **(a)** The AutoRally racetrack. **(b)** The decision boundary learned by g_ϕ. **(c)** The classification of uniformly sampled 2D datapoints across the 2D box classified by g_ϕ. **(d)** The points projected on Ω after applying Algorithm-1.

Prior distribution design can be classified by the nature of the environment variables under consideration. Specifically, we consider two cases:

(i) *Static EVs*: ($o \in \mathbb{R}^{d_1 \times m_1}$) These variables remain static throughout the evolution of the dynamic system. Here, m_1 defines the dimension of the search space and d_1 corresponds to the number of identical classes of EVs. In this study, we consider the problem of locating d_1 circular obstacles of fixed radius as a typical example.

(ii) *Sequence EVs*: ($o = (o_t)_{t=1}^{n_1}, o_t \in \mathbb{R}^{d_1 \times m_1}$) These variables have a sequential structure. Typical examples include reference paths and non-ego vehicles. Here n_1 denotes the length of sequence.

This distinction is motivated by the fact that Static EVs are low dimensional but have complex, implicit spatial constraints and do not change their position. On the other hand, Sequence EVs have constraints of temporal consistency, and are high dimensional. Therefore, different approaches suit their prior construction. In this paper, we focus on *independent* EVs, that do not take the ego-agent's behavior into consideration, such as a reference path or obstacles.

The spatio-temporal behavior (i.e., location, evolution with time, etc) of *independent* EVs can be predicted with known confidence. Hence, neural network-based architectures are a highly attractive means for learning accurate prior representations of these variables. The design approach for both cases is discussed in detail in the next sections.

A. Neural Projection for Static Environment Variables

For a fixed number of obstacles, consider the problem of finding obstacle locations on a racetrack for autonomous driving that are highly likely to collide with the vehicle. Here, the environment variable $o \in \mathbb{R}^{d_1 \times 2}$ represents d_1 obstacles scattered in a 2D environment around a pre-designed reference path.

Regions far away from the reference path have an extremely low probability of collision. Hence, a suitable choice of prior would include a region in the vicinity of the reference

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on September 08,2024 at 03:28:25 UTC from IEEE Xplore. Restrictions apply.
© 2024 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.

This article has been accepted for publication in IEEE Robotics and Automation Letters. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/LRA.2024.3455782

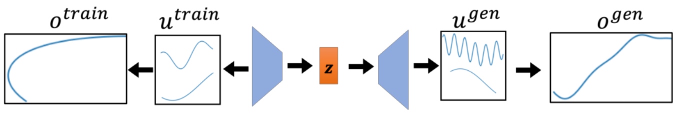


Fig. 3: Encoding latent variables for generating sequential environment variables 'o' using VAE presented in Section-IV-B.

track. Based on this hypothesis, for a given reference track, we construct a region Ω defined by inner and outer track boundaries of a width w. It is hard to express the region Ω as an explicit function since a parametric expression of the racetrack is not always available. We therefore first construct a box $\mathcal{D}$ circumscribing the region Ω such that $\Omega \subseteq \mathcal{D}$ and sample 2D data points uniformly from $\mathcal{D}$. We then train a DNN-based binary classifier $g_\phi(x, y)$ to learn the likelihood of a previously unseen datapoint (x, y) being within Ω or in $\mathcal{D} \backslash \Omega$. Fig. 2b shows an example of the learned decision boundary for the AutoRally racetrack [13] using $g_\phi(x, y)$.

To refine our Bayesian inference using this information, for any point within $\mathcal{D}$, we propose a Neural Projection Operator (NPO) $\boldsymbol{P} : \mathcal{D} \to \Omega$ as follows:

$$\boldsymbol{P}[o] = o - \nabla g_\phi(o) g_\phi(o) / \|\nabla g_\phi(o)\|^2. \tag{8}$$

This is a Newton's method-based projection operator, inspired by the Neural Projection technique presented in [14]. $\boldsymbol{P}[o]$ ensures that any point lying outside Ω is mapped to the nearest point within Ω. Fig. 2d shows an example applying Neural Projection with the AutoRally track region Ω, given uniformly generated samples in a 2D box $\mathcal{D}$.

Finally, for sampling static environment variables, we propose Algorithm-1, which combines the 2nd-order LA with the Neural Projection operator at each iteration to strictly sample from within Ω. After each update of 2nd-order LA sampling, we perform M updates of gradient-based projection using (8) given by $\tilde{o}_j \leftarrow \boldsymbol{P}[\tilde{o}_{j-1}]$, for $j = 1, \ldots, M$ where M is large enough to ensure that $\tilde{o}_M \in \Omega$. The process is outlined in Algorithm-1 shown below:

Algorithm 1 Neural Projection with Langevin

```
1: Initial conditions $o_0 \sim \mathcal{U}[\mathcal{D}]$          ▷ Uniform sampling
2: for $k = 1$ to $N$ do
3:     $\tilde{o}_0 \sim \mathcal{N}(\mu(o_{k-1}), \Sigma)$        ▷ 2nd-order LA sampling (7)
4:     for $j = 1$ to $M$ do
5:         $\tilde{o}_j \leftarrow \boldsymbol{P}[\tilde{o}_{j-1}]$          ▷ Neural Projection (8)
6:     end for
7:     $o_k = \tilde{o}_M$          ▷ Update $o_k$ after $M$ projection steps
8: end for
```

B. Latent Variable Encoding for Sequence Environment Variables

As a motivating example, given a reference-tracking controller, we consider the problem of finding reference paths that are hard for a dynamic system to follow. For this task, the environment variable under consideration is a reference path $o \in \mathbb{R}^{N \times 2}$ of length N, where $o_k = [x_k, y_k]^\mathsf{T}$ for $k = 0, \ldots, N-1$.

1) Challenges with optimization for sequence EVs: Firstly, the decision variable o is high-dimensional, and finding falsifying examples in a high-dimensional space can be quite inefficient and slow. Secondarily, our search space is constrained to be a set of "realisitic" trajectories which should have temporal consistency.

2) Approach: To tackle the second problem of temporal consistency, we use the evolution of a dynamic system to generate o. Consider a control affine dynamic system with open-loop control $\tilde{u} \in \mathbb{R}^m$ and drift and control vector fields $f(\cdot) : \mathbb{R}^p \to \mathbb{R}^p$ and $g(\cdot) : \mathbb{R}^m \to \mathbb{R}^p$, respectively. We generate every element of the sequence o_k using the discretization of the open-loop dynamic system $\dot{\tilde{o}} = f(\tilde{o}) + g(\tilde{u})$ such that $o_k = \tilde{o}_{k[0:2]}$. Therefore, we have, for all $k = 0, \ldots, N-1$,

$$\tilde{o}_k = \tilde{o}_{k-1} + \Delta t (f(\tilde{o}_{k-1}) + g(\tilde{u}_{k-1})).$$

For an appropriate choice of $f(\tilde{o})$, $g(\tilde{u})$ and Δt, we can generate a wide variety of sequences.

To resolve the first challenge of dimension reduction, we use Variational Autoencoders (VAE) to learn a lower-dimensional time-invariant representation of a continuous control input $\tilde{u}(t)$ such that $\tilde{u}_k = \tilde{u}(k\Delta t)$ defined in the range $t \in [0, N\Delta t]$. To learn a generalized low-dimensional representation, we need to train the model on a rich dataset that consists of a wide range of EVs. To generate the appropriate training dataset, we formulate the control inputs at every timestep t as a finite sum of n_1 sinusoidal component functions $h(A_j, \omega_j, \phi_j)$ where A_j, ω_j, ϕ_j represent the amplitude, frequency, and phase lag corresponding to each sinusoidal function. Mathematically, for a given $t \in [0, N\Delta t]$, $\tilde{u}$ can be expressed as:

$$\tilde{u}(t) = \sum_{j=1}^{n_1} A_j \sin(\omega_j t + \phi_j). \tag{9}$$

We generate the training data by uniformly sampling across a range of $\{A_j, \omega_j, \phi_j\}_{j=1}^{n_1}$ for a fixed sum length n_1 and chosen open-loop dynamic system. We wish to span across a large variety of functions for $\tilde{u}(t)$ to generate the training dataset. The expression for $\tilde{u}(t)$ shown in (9) is equivalent to a truncated Fourier series representation for $\tilde{u}(t)$. Therefore, by spanning across a range of $\{A_j, \omega_j, \phi_j\}_{j=1}^{n_1}$ in a low-dimensional space, we generate a large diversity of functions $\tilde{u}(t)$, which in turn leads to diverse evolutions of the chosen dynamic system, generating complicated realistic trajectories. This ensures that the generated trajectories have temporal consistency and diversity, and do not correspond to a restrictive family of parametric curves such as circles, ellipsoids, etc.

With $X = (A_j, \omega_j, \phi_j)_{j=1}^{n_1} \in \mathbb{R}^{n_1 \times 3}$ as the input and output of the Autoencoder, we train a DNN-based VAE, where the encoder network $g_{\text{enc}} : \mathbb{R}^{n_1 \times 3} \to \mathbb{R}^{d_2}$ learns a latent representation of the input space such that $d_2 \le 3n_1$. The network is trained to minimize the KL-divergence between the ground truth sequences and sequences generated by the control inputs rendered by the decoder network $g_{\text{dec}} : \mathbb{R}^{d_2} \to \mathbb{R}^{n_1 \times 3}$. The training pipeline is summarized in Fig. 3.

Therefore, we compress the problem of finding an N-dimensional sequence to an equivalent problem of finding a

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on September 08,2024 at 03:28:25 UTC from IEEE Xplore. Restrictions apply.
© 2024 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.

This article has been accepted for publication in IEEE Robotics and Automation Letters. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/LRA.2024.3455782

d_2-dimensional latent variable $z \in \mathbb{R}^{d_2}$ such that $d_2 \leq N$. After reducing the dimension of our search space, we use Langevin-based optimization to find the set of latent variables $\mathcal{Z}_{\text{fail}} = \{z|\bar{c}(g_{\text{dec}}(z)) > c^*\}$, where $\bar{c}$ measures the cost $c(o, X)$ generated by this method. Fig. 4 shows examples of different kinds of reference paths generated by our method, along with the corresponding trajectories generated by an LQR speed+steering controller for a bicycle dynamic model discussed in Section-V-B.

3) Limitations of our approach: The raceline dataset generation pipeline presented in this section is helpful for cases where baseline dataset is not already available in the required format for training the VAE. The overall pipeline relies on expert knowledge to make subjective assessment of “good quality” data for training purposes. The term “good quality” here refers to data that represents realistic representation of EVs that can lead to user-relevant failures upon inference.

V. Simulation & Experiments

We present results from a case study of testing using the sequence environment variables (Section-V-B) and falsification using the static environment variables (Section-V-C, V-D) along-with experimental validation in Section-V-E.

A. Baselines & Metrics

To demonstrate the modularity of our approach, we present an implementation of our framework on the 1st-order LA [12] and the gradient-free Random sampling (brute force method) in addition to the 2nd-order LA. We test for the effect of acceleration, speed of convergence, and coverage by comparing the failure discovery rate, mean and max cost value and dispersion metric C_{disp} respectively, across all baselines.

For comparing coverage of the search space, we introduce our own metric C_{disp} in (10). To calculate C_{disp}, sampled failure modes are clustered using Gaussian Mixture Clustering, and the maximum Euclidean distance between the mean positions of all clusters across all dimensions is computed. Mathematically, C_{disp} can be written as:

$$C_{\text{disp}} = \max_{1 \leq k \leq d} \{ \max_{1 \leq i,j \leq M} \{ \| \mu_i^k - \mu_j^k \|_2^2 \} \}. \tag{10}$$

Here M denotes the number of cluster components, μ_i^k denotes the mean of the i^{th} cluster in the k^{th} dimension. A higher value of C_{disp} implies better coverage of search space.

B. Case-study 1: Falsification of reference path tracking using speed and steering LQR control

1) Problem Description: We consider the problem of tracking a given reference path using an LQR controller, adopted from [15], for a bicycle dynamic model ($n = 4$) with steering angle and acceleration as control input ($u = [\delta, a]^\mathsf{T}$) and specified control limits ($\pm\delta^{\max}, \pm a^{\max}$). We define the performance metric as the average distance from the reference path o, with the cost function designed as

$$c(o, X) = \sum_{i=1}^{N} \| \bar{x}_i - o_i \|_2^2, \tag{11}$$

where $\bar{x} = x_{[0:2]}$. Failure is defined as $c > c^*$, with $c^* = 200$ chosen by trial and error. For this task, the environment variable is a reference path $o \in \mathbb{R}^{N \times 2}$ of length $N = 100$, where $o_k = [x_k, y_k]^\mathsf{T}$ for $k = 0, \ldots, N-1$. Hence, the testing objective is to find the set of reference paths for which LQR fails to generate a suitable tracking trajectory.

For generating falsifying racelines, we utilize the Sequence Environment variable design approach described in Section IV-B. We chose the bicycle dynamic vehicle model with steering and throttle as inputs to define $f(\tilde{o})$ and $g(\tilde{u})$, where, $\tilde{o} = [o_x, o_y, o_\theta, o_v]^\mathsf{T}$ and $o = \tilde{o}_{[0:2]}$.

2) Dataset Design: To train the VAE, we chose reference trajectories from a distribution that is very close to the actual failure distribution. This is done intentionally to observe the learned distribution and generalization capability of the VAE. We achieved this by introducing several instances of reference trajectories that require violation of control limits to track correctly. This is done by uniformly sampling around regions of amplitude $\{A_i\}_{i=1}^n$ higher than the control limit magnitude i.e., for a given σ, we chose:

$$\{A_i\}_{i=1}^n \sim \mathcal{U}[\delta^{\max} - 3\sigma, a^{\max} - 3\sigma] \times [\delta^{\max} + \sigma, a^{\max} + \sigma]. \tag{12}$$

Using this method, we learn the **Nominal** distribution of our four-dimensional latent space ($\mathcal{Z} \subseteq \mathbb{R}^4$) centered around $\mu_z = [0.8, 0.9, 0.95, 9]^\mathsf{T}$, as shown in Fig. 5 in blue.

3) Observations: We apply the 2nd-order LA to generate a failure distribution by sampling from the latent space $\mathcal{Z}$. Eventually, we notice that our method discovers a failure region far from the learned (nominal) distribution (**Region-1**, $\mu_z = [0.06, 0.03, 0.03, 0]^\mathsf{T}$, highlighted in red, Fig. 5) in addition to a low variance distribution near the nominal distribution (**Region-2**, $\mu_z = [0.99, 0.93, 0.89, 0.99]^\mathsf{T}$, Fig. 5).

We compared three baseline algorithms (10 chains, 100 epochs each), namely, brute force method, 1st and 2nd-order LA using the Sequence EV design framework. The results are summarized in Table-I. We observe that the 2nd-order LA outperforms the 1nd-order LA and the brute force method in discovering higher frequency of failures as well as covering larger search-space.

TABLE I: LQR Speed+Steering Control (Bicycle Model) baseline comparison.

	Failure rate	Mean cost	Max cost	C_{disp}
Random sampling	0.53	0.45	1.27	0.91
1st-order LA	0.43	0.36	1.27	1.43
2nd-order LA	**0.75**	**0.51**	**1.52**	**1.48**

C. Case-study 2: Falsification of Obstacle avoidance via SQP

1) Problem Description: In this example, we use the JAX library called trajax [16] to solve a reference tracking problem with obstacle avoidance on a semicircular track with obstacle locations of fixed radius as the decision variable. Here, the environment variable $o \in \mathbb{R}^{d_1 \times 2}$ represents d_1 obstacles scattered in a 2D environment within the semicircular track for a unicycle dynamic model with speed and angular velocity

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on September 08,2024 at 03:28:25 UTC from IEEE Xplore. Restrictions apply.
© 2024 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.

This article has been accepted for publication in IEEE Robotics and Automation Letters. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/LRA.2024.3455782

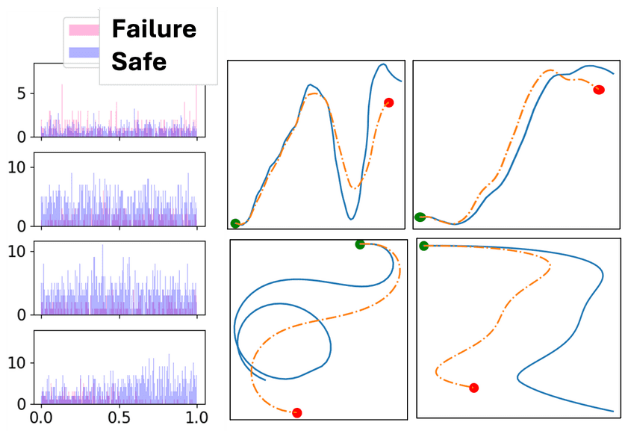


Fig. 4: Encoding distribution generated using Brute force method to reflect the high proportion of non-failure encodings (Left). Only 30% of the samples correspond to failure cases. Our method generates 1.9X failures using 0.5X samples with 75% failure discovery rate (Table-I). Examples of reference trajectory tracking failure cases generated using our method (Right). Reference trajectories shown in orange and actual trajectories of the vehicle shown in blue.

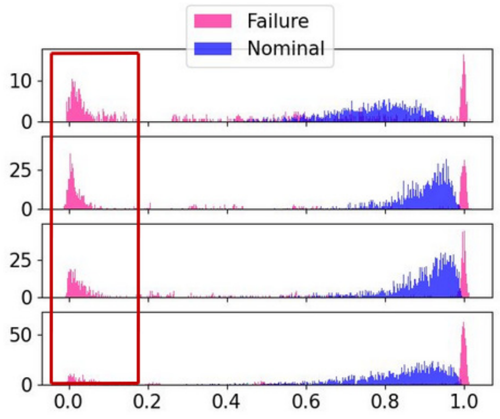


Fig. 5: VAE Latent space visualization for LQR on Bicycle Model example. The **Nominal** distribution is the latent distribution learned by the VAE on the training dataset. The **Failure** distribution corresponds to that discovered using our approach.

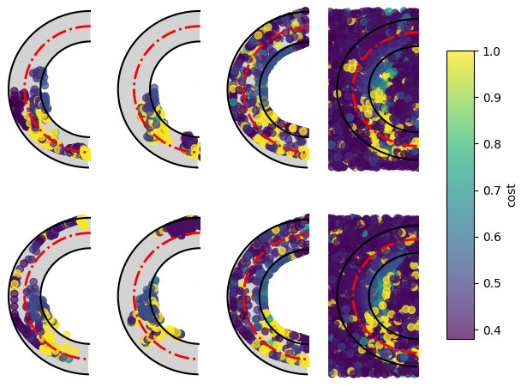


Fig. 6: Comparison of samples generated by 2nd-order LA, 1st-order LA, Brute force-II and Brute force-I (left to right). Obstacle-1 (Obstacle-2) positions are shown at the top (bottom). 2nd-order LA provides more spatial coverage of search space than the baselines. The colorbar shows the cost where failure corresponds to samples with cost ≥ 0.6. Our algorithm discovers more than 2X failures with 0.3X the sample size compared to Brute force-I method (w/o projection).

as control inputs $u = [v, \omega]^{\mathsf{T}}$. The inner and outer track boundaries are used for constructing a region Ω such that $o \in \Omega$, i.e, the prior $p(o)$ is well defined by the constraint of being within the boundaries, but is not explicitly known. We use the NPO presented in Section IV-A to learn g_ϕ that outputs the likelihood of a point being in or out of the track.

We aim to find obstacles that cause a significant deviation from the reference trajectory, and find instances where the solver favors feasibility over optimality. Algorithm-1 is applied for testing against the performance metric of the distance from the reference trajectory, with the cost function as in (11).

2) Observations: We conducted simulations for the case $d_1 = 2$, that is, scattering two obstacles in the environment and comparing between different methods across previously discussed metrics. In addition to the comparison between the 1st and the 2nd-order LA, we also compared with two variants of the brute force method, namely, Brute-force-I (Random Sampling) and Brute-force-II. Brute-force-II implements random sampling with neural projection, by applying the NPO $\boldsymbol{P}[o]$ (8) to each $o \sim \mathcal{U}(\mathcal{D})$. We implemented three chains with 200 epochs each for Brute-force-II, 1st and 2nd-order LA and compared it against 2,200 randomly generated samples by Brute-force-I method. The results are summarized in Table-II

Fig. 6 shows a distribution of obstacles generated by the various algorithms. The 2nd-order LA is able to move across low-probability regions to discover newer failure locations, while the 1st-order LA often gets stuck in local optima. Table-II shows that the 2nd-order LA has a higher dispersion and failure discovery rate compared to Brute-force-II, despite its even coverage of search space. Table-II also highlights the improvement achieved by neural projection, as Brute-force-II discovers more failures with a significantly smaller sample size (0.3X).

TABLE II: SQP baseline comparison.

	Failure rate	Mean cost	Max cost	C_{disp}
Brute force-I	0.17	0.50	1.0	28.68
Brute force-II	0.25	0.58	1.0	27.13
1st-order LA	0.40	0.67	1.0	26.76
2nd-order LA	**0.44**	**0.69**	**1.0**	**30.69**

D. Case-study 3: Falsification of Obstacle Avoidance for Trajectory Tracking via MPPI

1) Problem description: We use MPPI for autonomous racing on the AutoRally racing platform presented in [13] with a single-track bicycle dynamic model ($n = 7, m = 2$). Unlike the previously discussed examples, MPPI is a stochastic control algorithm that requires randomly sampled trajectories to synthesize an optimal control policy. The environment consists of a bounded reference trajectory and sets of d_1-obstacles scattered in the 2D space. The cost function of MPPI is fine-tuned to enable obstacle avoidance in addition to reference trajectory tracking. Here, we constructed a testing scenario of scattering two sets ($d_1 = 2$) of circular 2D obstacles with fixed radius ($r = 1.5$) and implemented Algorithm-1 to

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on September 08,2024 at 03:28:25 UTC from IEEE Xplore. Restrictions apply.
© 2024 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.

This article has been accepted for publication in IEEE Robotics and Automation Letters. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/LRA.2024.3455782

find falsifying locations of obstacles. Failure is defined as a collision and the corresponding cost function $c(\cdot)$ is:

$$\bar{c}(o, X) = \frac{1}{k} \log \sum_{i=1}^{d_1} \sum_{j=1}^{T} \exp\left(k\|\bar{x}_t - o_i\|_2^2\right), \qquad c(o, X) = -\bar{c}(o, X). \tag{13}$$

Here T is the length of simulation, k is a constant, set to $k = 1,000$. The function $\bar{c}(o, X)$ approaches $\max(\|\bar{x}_t - o_i\|_2^2)$ as $k \to +\infty$. Therefore, the sampling algorithm seeks to minimize the maximum distance of either obstacle from the trajectory. We used a smooth maximum function to ensure that both obstacles are in close proximity of the trajectory and can effectively contribute towards a possible collision.

2) Observations: Our method identifies several local minima on the racetrack. Additionally, we observe that collision with an obstacle depends on the local state behavior, i.e., the speed, the angular velocity, and the position at the time of collision and a few steps before that, and is relatively unaffected by the state of the vehicle long before that. We classify the discovered failure modes into two categories:

(i) *Type-1*: Each instance of this failure mode is a combination of two independent local minima, i.e., the vehicle has a high probability of collision with both obstacles individually. For such failure instances, we can write

$$p_{\text{fail}}(o_i|o_j) = p_{\text{fail}}(o_i) \quad \text{for } i = 1, 2, \; j = 2, 1. \tag{14}$$

(ii) *Type-2*: This failure mode consists of two obstacles such that the probability of collision of the vehicle is increased due to the presence of the other obstacle. Hence, the location of the first obstacle informs the location of the other obstacle and this combination creates new local minima. For such a failure we have

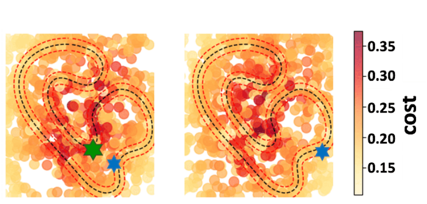
$$\begin{aligned} p_{\text{fail}}(o_2, o_1) &> \max\{p_{\text{fail}}(o_2), p_{\text{fail}}(o_1)\} \\ p_{\text{fail}}(o_1|o_2) &= p_{\text{fail}}(o_1), \end{aligned} \tag{15}$$

where, o_1 is the first obstacle encountered by the vehicle and o_2 is encountered after o_1.

All **Type-1** failure modes can be expressed as the union of independent failure modes of lower cardinality, which corresponds to $d_1 = 1$ in this case.

To demonstrate the utility of our method, we performed brute-force sampling without projection using 1,700 random samples to generate an approximate cost distribution of obstacles (Fig. 7a). The high-cost region marked in green is a local optimum corresponding to the *Type-1* failure mode shown in Fig. 7b, generated by Algorithm-1. Note that random sampling is not able to locate *Type-2* failures effectively, as *Type-2* failures are sensitive to small perturbations in the location of **Obstacle-2**, but our algorithm successfully locates several *Type-1* and *Type-2* failures with almost an order of magnitude fewer samples than the brute-force method. Fig. 8c shows an instance of **Type-2** failure mode. Colliding (resp., safe) trajectories shown in blue (resp,, orange). The likelihood of collision is calculated over 100 random trials.

We also compared the performance with baseline algorithms for a shorter length trajectory tracking example over five



(a)

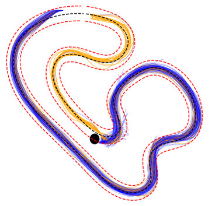
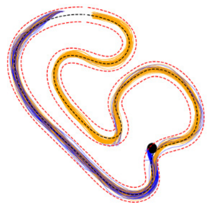
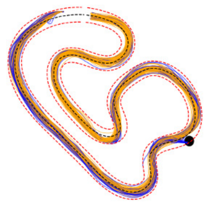

(b) Type-1 failure

Fig. 7: Cost distribution of **Obstacle-1** (left, Plot (a)) and **Obstacle-2** (right, Plot(a)), generated using the brute-force method. Examples of obstacles corresponding to *Type-1* failure (green) and *Type-2* (blue) are shown. Plot (b) shows the collision likelihood with the *Type-1* obstacle (94%). Note that the collision happens regardless of the location of **Obstacle-2**.


(a) Obstacle-1


(b) Obstacle-2

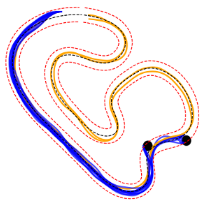

(c) Type-2 failure

Fig. 8: Likelihood of collision of **Obstacle-1** (Plot (a), 61%) and **Obstacle-2** (Plot (b), 39%) respectively. Plot (c) shows collision likelihood with both **Obstacle-1**, **Obstacle-2** (81%).

chains of 50 epochs each. The results are summarized in Table-III. The reported cost values in Table-III is given by $\exp(c/250)$, where c is defined in (13). Here, we notice that Brute-force-II performs similarly to 2nd-order LA, which is primarily due to the restricted search space of the sampling region. This shows that for smaller sampling volumes, in the absence of true gradient information, the neural projection technique can be helpful in reducing sampling volume and thereby increasing the efficiency of the sampling algorithm.

We notice that the 2nd-order LA outperforms the baselines, by doing well both in terms of exploration and in terms of convergence.

TABLE III: MPPI baseline comparison.

	Failure rate	Mean cost	Max cost	C_{disp}
Brute force-I	0.06	0.35	0.53	24.11
Brute force-II	0.116	0.37	0.55	23.57
1st-order LA	0.081	0.44	0.56	11.84
2nd-order LA	**0.1**	**0.48**	**0.59**	**17.06**

E. Experimental Validation

We analyze the sim-to-real transfer of our approach by conducting hardware experiments for testing of static EVs (obstacles, Section-V-D) and sequence EVs (reference path, Section-V-B) using the AutoRally racing platform and the F1Tenth platform, respectively. Please refer to the video[1] for experimental demonstration.

[1]https://mit-realm.github.io/neural-langevin-website/

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on September 08,2024 at 03:28:25 UTC from IEEE Xplore. Restrictions apply.
© 2024 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.

This article has been accepted for publication in IEEE Robotics and Automation Letters. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/LRA.2024.3455782

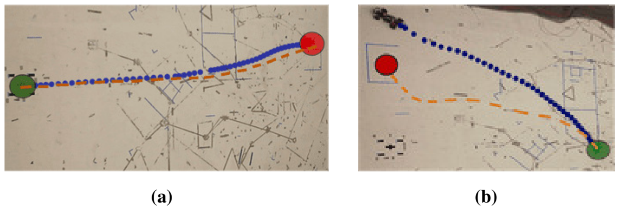


Fig. 9: Fig. (a), (b) show the real-time demonstration of nominal and "failure" reference paths, respectively. The "failure" reference path cannot be tracked by implementing the LQR controller as explained in Section-V-B.

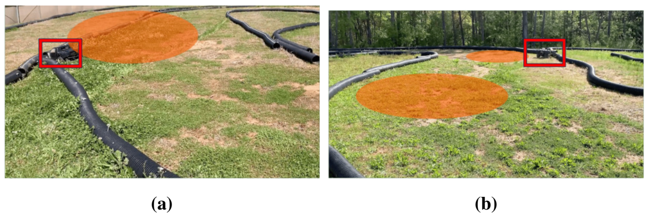


Fig. 10: Fig. (a), (b) show the hardware demonstrations of *Type-1* and *Type-2* failure respectively found using our approach. Corresponding simulation and analysis in Fig. 7 and Fig. 8.

1) LQR reference path tracking falsification using the F1Tenth platform: We performed hardware demonstration for reference path falsification using the F1Tenth autonomous racing platform [17]. The LQR controller for Speed and Steering Control presented in Section-V-B was implemented on the vehicle for tracking a known reference path. Fig. 9b shows instances where the vehicle was not able to reach the goal and deviated significantly from the reference path. The marginal difference in the observed vehicle trajectory and the simulated response reflects the sim-to-real gap of our model-based testing framework, which we aim to address in future iterations of our work.

2) MPPI Trajectory tracking with obstacle avoidance falsification using the AutoRally platform: We validated the falsification of the MPPI controller on the AutoRally platform in the presence of obstacles with a goal to replicate the *Type-1* and *Type-2* failures observed in Fig. 7b and Fig. 8. Fig. 10 shows the hardware demonstration of the failures discovered in simulation.

VI. Conclusion and Future Work

In this paper, we have proposed a novel learning-based framework for testing of robotic systems via Bayesian inference. As seen from the various examples presented in the paper, learning-based representations can solve the curse of dimensionality using latent variable encoding, and enable an explicit constraint satisfaction scheme using the neural projection technique.

Across all examples, we observe that the 2nd-order LA provides an optimal way to balance coverage and failure discovery rate via momentum-based acceleration. We also provide a new way to analyze failure patterns by constructing high-dimensional failure distributions using low-dimensional failures of the same class. This eliminates the need to perform high-dimensional sampling in many scenarios.

This paper analyzes the role of differentiability in testing, and the conclusion is that differentiability can substantially help in feedback control-based architectures as the explicit construction of gradients via backpropagation is easy. For optimal control architectures such as SQP and MPPI, one would benefit from an easy-to-implement sensitivity analysis of the environment variables of the optimization problem. Furthermore, the question of hierarchical failures introduced in Section-V-D is promising to significantly improve the memory and convergence of many testing methods.

References

[1] 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*. Springer, 2011, pp. 254–257.
[2] P. Koopman and M. Wagner, "Autonomous vehicle safety: An interdisciplinary challenge," *IEEE Intelligent Transportation Systems Magazine*, vol. 9, no. 1, pp. 90–96, 2017.
[3] E. Wong and Z. Kolter, "Provable defenses against adversarial examples via the convex outer adversarial polytope," in *International conference on machine learning*. PMLR, 2018, pp. 5286–5295.
[4] S. S. Du, C. Jin, J. D. Lee, M. I. Jordan, A. Singh, and B. Poczos, "Gradient descent can take exponential time to escape saddle points," *Advances in neural information processing systems*, vol. 30, 2017.
[5] A. Kundu, S. Gon, and R. Ray, "Data-driven falsification of cyber-physical systems," in *Proceedings of the 17th Innovations in Software Engineering Conference*, 2024, pp. 1–5.
[6] C. Dawson and C. Fan, "A Bayesian approach to breaking things: efficiently predicting and repairing failure modes via sampling," in *7th Annual Conference on Robot Learning*, 2023. [Online]. Available: https://openreview.net/forum?id=fNLBmtyBiC
[7] H. Delecki, A. Corso, and M. Kochenderfer, "Model-based validation as probabilistic inference," in *Learning for Dynamics and Control Conference*. PMLR, 2023, pp. 825–837.
[8] Y.-A. Ma, N. S. Chatterji, X. Cheng, N. Flammarion, P. L. Bartlett, and M. I. Jordan, "Is there an analog of Nesterov acceleration for gradient-based MCMC?" 2021.
[9] A. Parashar, "Accelerated algorithms for constrained optimization and control," Ph.D. dissertation, Massachusetts Institute of Technology, 2023.
[10] Y. Nesterov, *Lectures on Convex Optimization*. Springer International Publishing, 2018, vol. 137.
[11] X. Gao, M. Gurbuzbalaban, and L. Zhu, "Breaking reversibility accelerates Langevin dynamics for non-convex optimization," *Advances in Neural Information Processing Systems*, vol. 33, pp. 17 850–17 862, 2020.
[12] Y.-A. Ma, Y. Chen, C. Jin, N. Flammarion, and M. I. Jordan, "Sampling can be faster than optimization," *Proceedings of the National Academy of Sciences*, vol. 116, no. 42, pp. 20 881–20 885, 2019.
[13] J. Yin, Z. Zhang, E. Theodorou, and P. Tsiotras, "Trajectory distribution control for model predictive path integral control using covariance steering," in *International Conference on Robotics and Automation (ICRA)*, 2022, pp. 1478–1484.
[14] S. Yang, X. He, and B. Zhu, "Learning physical constraints with neural projections," *Advances in Neural Information Processing Systems*, vol. 33, pp. 5178–5189, 2020.
[15] J. Levinson, J. Askeland, J. Becker, J. Dolson, D. Held, S. Kammel, J. Z. Kolter, D. Langer, O. Pink, V. Pratt, M. Sokolsky, G. Stanek, D. Stavens, A. Teichman, M. Werling, and S. Thrun, "Towards fully autonomous driving: Systems and algorithms," in *IEEE Intelligent Vehicles Symposium (IV)*, 2011, pp. 163–168.
[16] S. Singh, J.-J. Slotine, and V. Sindhwani, "Optimizing trajectories with closed-loop dynamic SQP," in *International Conference on Robotics and Automation (ICRA)*, 2022, pp. 5249–5254.
[17] B. D. Evans, R. Trumpp, M. Caccamo, H. W. Jordaan, and H. A. Engelbrecht, "Unifying F1Tenth autonomous racing: Survey, methods and benchmarks," *arXiv preprint arXiv:2402.18558*, 2024.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on September 08,2024 at 03:28:25 UTC from IEEE Xplore. Restrictions apply.
© 2024 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.