Failure Prediction from Limited Hardware Demonstrations

Anjali Parashar, Kunal Garg, Joseph Zhang, and Chuchu Fan

Abstract— **Prediction of failures in real-world robotic systems either requires accurate model information or extensive testing. Partial knowledge of the system model makes simulation-based failure prediction unreliable. Moreover, obtaining such demonstrations is expensive, and could potentially be risky for the robotic system to repeatedly fail during data collection. This work presents a novel three-step methodology for discovering failures that occur in the true system by using a combination of a limited number of demonstrations from the true system and the failure information processed through sampling-based testing of a model dynamical system. Given a limited budget N of demonstrations from true system and a model dynamics (with potentially large modeling errors), the proposed methodology comprises of a) exhaustive simulations for discovering algorithmic failures using the model dynamics; b) design of initial N_1 demonstrations of the true system using Bayesian inference to learn a Gaussian process regression (GPR)-based failure predictor; and c) iterative $N - N_1$ demonstrations of the true system for updating the failure predictor. To illustrate the efficacy of the proposed methodology, we consider: a) the failure discovery for the task of pushing a T block to a fixed target region with UR3E collaborative robot arm using a diffusion policy; and b) the failure discovery for an F1-Tenth racing car tracking a given raceline under an LQR control policy.**

I. Introduction

Testing and model validation are essential tools for ensuring the safety of autonomous systems before deployment [1]–[5]. These techniques primarily pertain to discovering failures using available model information to study the limitations of decision-making pipelines and their subsequent improvement [5]–[10]. Based on the extent of information available about the model, various model-based tools for testing and falsification tools have been proposed in the literature [11]–[16]. The model-based tools have the advantage of being faster [13], [14], and with differentiable system models, gradient-based optimization strategies can be used for falsification, which have shown to be more efficient than black-box methods [17]–[19]. They also allow exploitation of model-specific information to expedite failure discovery.

Recently, some works have considered simulation-based falsification and testing [14], [20]. These methods minimize the dependency on hardware for testing, which can be extremely time-consuming and require extensive utilization of resources [21]. These methods can be further divided into three categories based on the underlying framework used. The first set of methods uses adversarial optimization to return a single failure per search [17], [22], [23]. Some of these methods suffer from challenges such as getting stuck in local minima, due to their dependency on gradient-based optimization [24]. The second class of methods utilizes learning-based approaches, with recent advances in generative modeling, to generate failure scenarios using information learned from the training dataset, and requires a sufficient amount of training data [20], [25], which can be challenging, since failures in autonomous systems are often low probability events or corner cases [13], [26], [27]. The third set of methods leverage probabilistic inference and sampling-based techniques, which return multiple failures per search [28], [29], however, also suffer from issues of slow convergence and uneven search space exploration. Recently, some works have explored combinations of these fundamental techniques to overcome the limitations of each [13], [14], [16].

Most of these approaches assume that the simulation dynamics, which we refer to as the *model dynamics*, and testing environment represent the realistic testing conditions adequately. However, this can be misleading, especially if the *true system* and its environmental interactions are complex and cannot be adequately captured using appropriate simulation models [16]. Additionally, there are uncertainties in state estimation and dynamics, that affect the performance of true systems [30]. Collectively, these issues lead to failure modes that remain undiscovered, despite exhaustive simulation testing, due to the inadequacy of the simulation models in capturing wide-ranging practical phenomenon that affect the true system. This can impact the performance of autonomous systems, which is concerning from the perspective of safety. It is possible that the discovered failure modes in simulation do not reflect the true severity of real failures and that a scenario reported as safe in simulation may correspond to failure in real-time. This study investigates the discovery of failures that are hard to find just from the model dynamics due to sim-to-real gap and a limited budget for testing the true system. We explore the utility of Gaussian Process Regression (GPR) to extract failure information from true systems with limited demonstrations [31], [32], in addition to conventional exhaustive simulation-based testing. We leverage random sampling for testing the model dynamics and adopt a sampling-based perspective for failure discovery.

In this paper, we investigate the sim-to-real gap from the perspective of falsification, using existing testing pipelines for simulation, while working with limited data from the true system to enable better prediction of failures (Section. II). Our proposed methodology consists of three steps (Section. III), namely, data collection of failure information from model dynamics, followed by an initial set of demonstrations on the true system, which is used to train an initial failure prediction model using GPR. This is followed by a sequential

The authors are with the Department of Aeronautics and Astronautics, Massachusetts Institute of Technology, USA. Email: {anjalip,kgarg,jzha,chuchu}@mit.edu

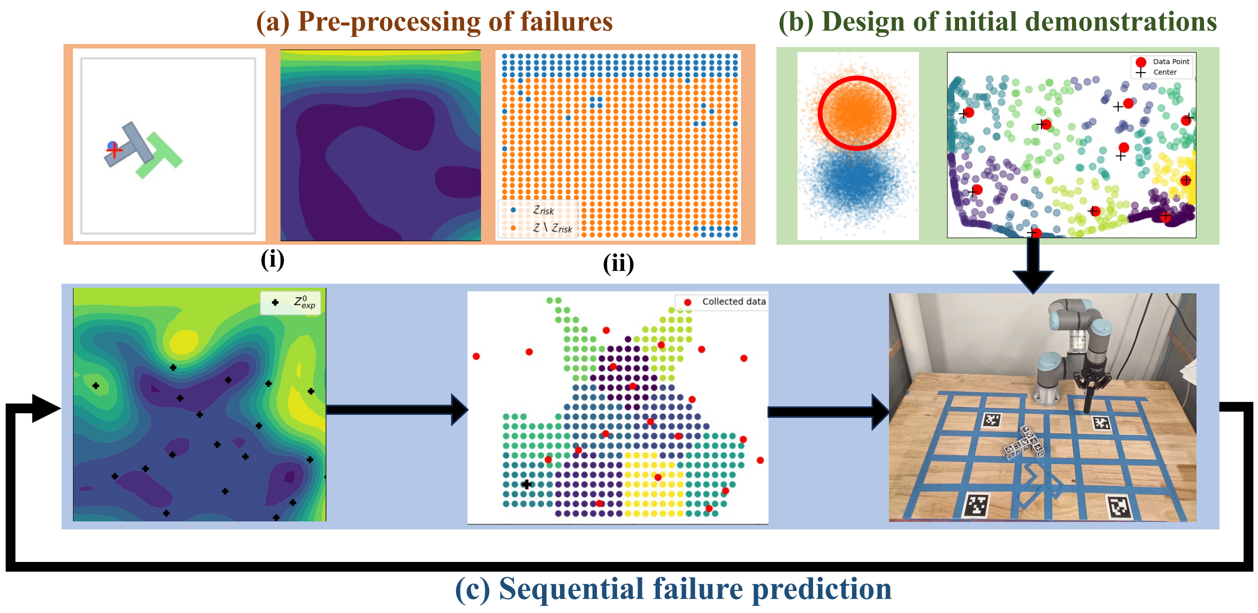
(a) Pre-processing of failures
(b) Design of initial demonstrations
$\mathcal{Z}_{risk}$
$\mathcal{Z} \setminus \mathcal{Z}_{risk}$
Data Point
Center
(i)
(ii)
$\mathcal{Z}^0_{exp}$
Collected data
(c) Sequential failure prediction

Fig. 1: The proposed methodology constitutes a) discovering failures using model information; b) design of initial demonstrations to learn true system failures using Bayesian inference; and c) sequential demonstration from low predicted-risk regions for GPR-based risk prediction update.

demonstration of the true system to uncover failures that cannot be captured by the simulations and subsequent improvement of the failure prediction model. In this step, the demonstrations are chosen to either validate that the regions defined as 'not fail' by the then-learned failure predictor are truly safe or correspond to an undiscovered failure. Fig. 1 summarizes the proposed methodology.

We validate our approach on two robotic examples, namely, pushing a T-block in a designated region [33] with a Diffusion policy using the UR3E collaborative robot end-effector [34], and reference path-tracking with an LQR Speed+Steering controller [35] using the F1-Tenth autonomous racing platform [36]. For both problems, the policies are designed using simulation environments which represent simplified estimations of the true dynamic systems, and are subsequently implemented on the true dynamic systems. This leads to failure modes that are not observed in simulation, which we aim to discover using our methodology. Our methodology is able to discover failures with an accuracy of 89% and 100% as compared to 11% and 36% by a purely simulation-based failure predictor, in the first and the second example, respectively.

II. Problem Formulation

We consider two sets of dynamics in this paper: model (known to the user) and true (unknown to the user). *True dynamics* corresponds to the actual dynamics of the agent, which is unknown whereas *model dynamics*, or simply, model, corresponds to the estimate of the true dynamics and is described as:

$$x_{t+1} = f(x_t, \pi(y_t, z)) + \epsilon_1, \qquad y_t = Cx_t + \epsilon_2, \qquad (1)$$

where $f : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^n$ and $C \in \mathbb{R}^{l \times n}$, with state $x_t \in \mathbb{R}^n$ at time $t \in \mathbb{R}$ and policy $\pi : \mathbb{R}^l \times \mathbb{R}^d \to \mathcal{U} \subseteq \mathbb{R}^m$ which outputs actions based on environmental variables $z \in \mathcal{Z} \subset \mathbb{R}^d$ and system output $y_t \in \mathbb{R}^l$, where ϵ_1 and ϵ_2 are disturbances in dynamics and state estimation, respectively. The environment variable z represents independent variables, such as initial conditions x_0 of the system along with the environmental information exogenous to the system. We assume that the disturbances come from zero-mean Gaussian distributions given by $\epsilon_1 \sim \mathcal{N}(\mathbf{0}, \Sigma_1)$ and $\epsilon_2 \sim \mathcal{N}(\mathbf{0}, \Sigma_2)$, where the covariance matrices $\Sigma_1 \in \mathbb{R}^{n \times n}, \Sigma_2 \in \mathbb{R}^{l \times l}$ of the distributions are defined using scalars $\sigma_1, \sigma_2 \geq 0$ as $\Sigma_i = \sigma_i \boldsymbol{I}$, for $i = 1, 2$, where $\boldsymbol{I}$ is an identity matrix of the appropriate size. For a given z, we denote a trajectory rollout of the model (1) under a given (σ_1, σ_2) as $X_{(z|\sigma_1,\sigma_2)} = (x_i)_{i=0}^T$ and of the true system as X_z^*.[1]

For the purpose of failure prediction, we consider a user-defined risk function $R : \mathbb{R}^d \to \mathbb{R}$ where $R(z) = R(z, X_z)$ denotes the risk corresponding to the trajectory rollout X_z for a given environment variable z. Based on this risk function, we define failure of the system for a corresponding z when the risk $R(z)$ exceeds a user-defined threshold $R_{\text{th}} \in \mathbb{R}$. The falsification of the closed-loop system, or plainly, failure discovery problem can be mathematically formulated as discovering the set $\mathcal{Z}^*_{\text{fail}} := \{z \mid R(z, X_z^*) > R_{\text{th}}\}$. We aim to solve this under the constraint that we can query the true system only a limited times $N > 0$ to obtain N trajectory rollouts $\{X^*_{z_i}\}_{i=1}^N$ for $z_i \in \mathcal{Z}$. We present a three-step methodology to discover failures occurring in the true system by using a combination of a minimal number of demonstrations $\{X_z^*\}_N$ and the failure information from the model dynamics (denoted as $\mathcal{Z}_{\text{fail}}(f)$) obtained through sampling-based falsification (see Figure 1). In brief, the pro-

[1]In what follows, we suppress the explicit dependence on σ_1, σ_2 for the sake of brevity.

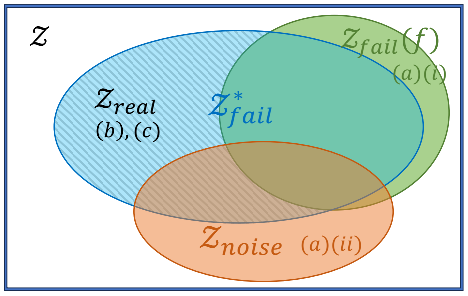

Fig. 2: Venn Diagram showing different sets defined in Section. III. Here, $\mathcal{Z}_{\text{fail}}(f)$ and $\mathcal{Z}_{\text{noise}}$ denote the set of failures that can be discovered using model dynamics (Fig. 1-a. (i) and (ii) respectively), and $\mathcal{Z}_{\text{real}}$ is estimated using demonstrations from the true system (Fig. 1-b. and c.)

posed methodology constitutes a) exhaustive simulations for discovering algorithmic failures using the model dynamics; b) design of initial $N_1 < N$ demonstrations from the true system using Bayesian inference to learn a Gaussian process regression (GPR)-based failure predictor; and c) iterative $N - N_1$ demonstrations of the true system for updating the failure predictor, where $N_2 = N - N_1$.

III. Methodology

We assume that the model (1) can capture certain failures that could occur with the true system, i.e., $\mathcal{Z}_{\text{fail}}(f) \cap \mathcal{Z}^*_{\text{fail}} \neq \emptyset$. Based on this, we obtain that $\mathcal{Z}^*_{\text{fail}} \subseteq \mathcal{Z}_{\text{fail}}(f) \cup \mathcal{Z}_{\text{real}}$, i.e, failures of the true system are a combination of algorithmic failures on the model system $\mathcal{Z}_{\text{fail}}(f)$ and failures due to the mismatch between model dynamics and actual dynamics, disturbances and other potentially unknown reasons, cumulatively represented by $\mathcal{Z}_{\text{real}}$. We say that the set $\mathcal{Z}_{\text{fail}}(f)$ captures algorithmic failures as we assume that the policy π in (1) is designed for the model f but still leads to failures. The first step of our methodology focuses on discovering failures that can be obtained using the model information.

A. *Pre-processing of failures: utilizing model information*

We define the set of the environment variables for the algorithmic failures of the model dynamics as:

$$\mathcal{Z}_{\text{fail}}(f) := \{z \mid R(z, X_{(z|\sigma_1=0,\sigma_2=0)}) > R_{\text{th}}\}, \qquad (2)$$

This set can be obtained through extensive simulations using the model information. Next, we aim to capture the failures due to the mismatch between model dynamics and true dynamics, disturbances, and potentially other unknown reasons. For this, we sample σ_1, σ_2 from a bounded region given by $\mathcal{B} := [\sigma_1^{\text{min}}, \sigma_1^{\text{max}}] \times [\sigma_2^{\text{min}}, \sigma_2^{\text{max}}]$, and using the risk $R(z, X_z)$ corresponding to each disturbance, we collect values of z for which a failure is observed across all disturbances. Here, the maximum values of the variance $\sigma_1^{\text{max}}, \sigma_2^{\text{max}}$ are chosen high enough to appropriately represent the scale and magnitude of disturbances experienced in the true system, based on expert knowledge. We define $\mathcal{Z}_{\text{noise}}$ as the set of z for which the model system fails due to disturbances as:

$$\mathcal{Z}_{\text{noise}} := \{z \mid R(z, X_{(z|\sigma_1,\sigma_2)}) > R_{\text{th}} \; \forall \sigma_1, \sigma_2 \in \mathcal{B} \backslash \{[0,0]\}\}, \qquad (3)$$

so that $\mathcal{Z}^* \cap \mathcal{Z}_{\text{noise}} \neq \emptyset$. The set $\mathcal{Z}_{\text{risk}} := \mathcal{Z}_{\text{noise}} \cup \mathcal{Z}_{\text{fail}}(f)$ captures all possible failures that can be discovered using model dynamics (1). Fig. 2 illustrates these various failure sets and their inter-relationships. The next step is to discover failures that the model system cannot capture through sampling z from the region $\mathcal{Z}_{\text{real}}$ and obtaining demonstrations from true dynamics. Since $\mathcal{Z}_{\text{real}}$ is not known, we obtain these samples from the region $\mathcal{Z} \backslash \mathcal{Z}_{\text{risk}}$ as discussed next.

B. *Sampling from sensitive regions: Design of experiments*

Since we have a limited budget on the number of demonstrations we can obtain from the true dynamics, we choose to collect the demonstrations that maximize the state-space coverage. For a given z, define a coverage function $C : \mathcal{Z} \to \mathbb{R}$ where $C(z) = C(z, X_z)$ is a monotonically increasing function of the explored state-space along the trajectory X_z (see Section IV for examples of the coverage functions used in practice). With this, we aim to sample $z \in \mathcal{Z} \backslash \mathcal{Z}_{\text{risk}}$ from a region corresponding to high coverage, given by:

$$z \sim \{z | C(z, X_z) > C_{\text{th}}, z \in \mathcal{Z} \backslash \mathcal{Z}_{\text{risk}}\}, \qquad (4)$$

where $C_{\text{th}} > 0$ is a user-defined coverage threshold. Sampling directly from this set is intractable, and so, we utilize a Bayesian inference framework here [16], [37]. To sample exclusively from $\mathcal{Z} \backslash \mathcal{Z}_{\text{risk}}$, we learn the decision boundary that distinguishes $\mathcal{Z}_{\text{risk}}$ from the remaining search space by performing supervised binary classification. For this, we use a specific technique known as Flow-GMM [38], which allows us to learn an invertible mapping from search-space $\mathcal{Z}$ to a latent-space $\mathcal{W} \in \mathbb{R}^d$, given by $g_\theta^{-1} : \mathcal{Z} \to \mathcal{W}$ using the Normalizing Flows framework [39]. Using Flow-GMM, we learn isotropic Gaussian latent distributions $\tilde{p}_1, \tilde{p}_2$ corresponding to the regions $\mathcal{Z}_{\text{risk}}$ and $\mathcal{Z} \backslash \mathcal{Z}_{\text{risk}}$ which can be mathematically expressed using mean μ_i^g and covariance Σ_i^g as $\tilde{p}_i = \mathcal{N}(\mu_i^g, \Sigma_i^g)$ for $i = 1, 2$. We can rewrite sampling from the set in (4) to sampling from a distribution in the latent space $\mathcal{W}$ using the learned mapping g_θ as:

$$w \sim \tilde{p}_2(w | C(g_\theta(w), X_{g_\theta(w)}) > C_{\text{th}}) \qquad (5)$$

We utilize exponential modelling for expressing the likelihood $p(C > C_{\text{th}} | g_\theta(w))$, adopted from [37] and $\tilde{p}_2$ as the prior, to construct the posterior distribution for sampling. Using Bayes rule, (5) can be simplified as:

$$\begin{aligned} w &\sim \tilde{p}_2(w | C(g_\theta(w), X_{g_\theta(w)}) > C_{\text{th}}) \\ &\propto \exp\big(-[C_{\text{th}} - C(g_\theta(w), X_{g_\theta(w)})]_+\big)\tilde{p}_2(w). \end{aligned} \qquad (6)$$

Here $[]_+$ represents the ReLU operator, and ensures that each w corresponding to $C(g_\theta(w), X_{g_\theta(w)}) \geq C_{\text{th}}$ is prioritized equally. We use Metropolis-Hashtings algorithm [40] to sample w from the constructed posterior distribution, which can also be replaced by another sampling algorithm of choice.

While we are sampling from the posterior (5), the generated iterates must satisfy $g_\theta(w) \in \mathcal{Z} \backslash \mathcal{Z}_{\text{risk}}$. This is done by using a projection operator $\mathbb{P}[w]$ to ensure that the generated samples lie within a convex set centered at the mean μ_2^g, so that the projected sample has a high probability of being in the set $\mathcal{Z} \backslash \mathcal{Z}_{\text{risk}}$. Since $\tilde{p}_2(w)$ is isotropic, we chose $\boldsymbol{P}[w] = c + r\frac{w-c}{\|w-c\|_2}$ with $c = \mu_2^g$, and r as a user-defined variable that can be decreased to make the sampling more conservative or vice versa. It can be easily verified that for a point w such that $\|w - c\|_2 > r$, the projected point $\tilde{w} = \boldsymbol{P}[w]$ satisfies $|w - c|_2 \leq r$, therefore the mapping of the projected sample $g_\theta(\tilde{w})$ has a high probability of lying in the region $\mathcal{Z}\backslash\mathcal{Z}_{\text{risk}}$ for an appropriately chosen r. Fig. 1 (ii)-(iii) show the latent distribution learned using Flow-GMM corresponding to $\mathcal{Z}_{\text{risk}}$ and $\mathcal{Z}\backslash\mathcal{Z}_{\text{risk}}$, and the boundary of the constructed set $\mathcal{P}$ in red for the Push-T task discussed in Section. IV-A

The pipeline discussed so far generates a collection of samples $Z_{\text{cov}} = \{z \in \mathcal{Z}\backslash\mathcal{Z}_{\text{risk}} \mid C(z, X_z) > C_{\text{th}}\}$. Once we have generated the samples, we choose N_1 candidate values of z distributed uniformly across the search-space. This is achieved by dividing Z_{cov} into N_1 clusters using k-means clustering, and choosing the points in Z_{cov} closest to the geometric centers of the generated clusters for demonstrations. This allow us to collect N_1 risk values given by $R_1 = \{R(z_j, X^*_{z_j})\}$ for the data points $Z_1 = \{z_j\}$. We also obtain M data points $Z_2 = \{z_i\}$ from the region $\mathcal{Z}_{\text{risk}}$ (Section. III-A), with the corresponding risk values given by $R_2 = \{R(z_i, X_{z_i})\}$ using model dynamics f. Define $\mathcal{D}_1 = [Z_1, R_1]$ and $\mathcal{D}_2 = [Z_2, R_2]$ as the dataset of demonstrations obtained from true and model systems, respectively. We use these dateset to train a GP model $\phi_\theta : \mathcal{Z} \to \mathbb{R}$ to predict risk $\hat{R} = \phi_\theta(z)$ for a given z, as illustrated in the next section where θ denotes the model parameters.

C. Sequential failure prediction and training using GPR

Motivated by the success of GPR in learning from limited demonstrations [32], we use GPR as the backbone of the risk prediction pipeline. Using the dataset $\mathcal{D}_f = \mathcal{D}_1 \cup \mathcal{D}_2$, we construct the marginal log likelihood $\log p_{\phi_\theta}(R_f|Z_{\text{f}})$ for learning the model ϕ_θ using a sum of the marginal log likelihoods from both sources of data as:

$$\log p_{\phi_\theta}(R_f|Z_{\text{f}}) = \log p_{\phi_\theta}(R_1|Z_1) + \log p_{\phi_\theta}(R_2|Z_2), \quad (7)$$

where $Z_f = [Z_1, Z_2]$ and $R_f = [R_1, R_2]$. The training objective is the maximization of the marginal log likelihood in (7) with ϕ_θ as the decision variable:

$$\phi_{\theta_0} = \arg\max_{\theta} \log p_{\phi_\theta}(R_f|Z_f). \quad (8)$$

We train the risk predictor model N_2 times by a sequence of N_2 demonstrations on the true system with sequential optimization of θ by solving (8) and generation of a new data-point z for the next demonstration. In particular, at each step k, we first divide the region $\{z \mid \phi_\theta(z) < R_{\text{th}}\}$ which is predicted 'not fail' by the learned model into N_2 clusters using k-means clustering. We choose z_k from the set of geometric means c_i of the clusters as the point which maximizes the distance from the previously chosen points:

$$z_k = \arg\max_{z \in \mathcal{C}_k} \min_{q \in [Z_1, Z_{k-1}]} \|z - q\|_2^2 \quad k = 1, \ldots, N/2, \quad (9)$$

where $\mathcal{C}_k = [c_1, \ldots, c_{N/2}]$ and $Z_{k-1} = [z_1, \ldots, z_{k-1}]$. Each step of demonstration is followed by training of the GPR model ϕ_θ with the dataset updated with z_k:

$$\phi_{\theta_k} = \arg\max_{\theta} \log p(\hat{R}|\phi_\theta, [Z_{\text{f}}, (z)_{i=1}^k]) \quad (10)$$

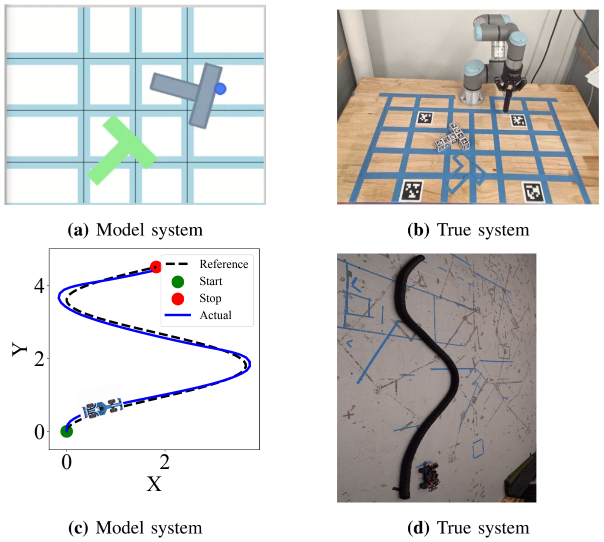

Fig. 3: The simulation (model dynamics) environment and the hardware (actual dynamics) setup for the Push-T (top figures) and F1-Tenth (bottom figures) examples.

Since the number of data points is limited, training the GPR model N_2 times is not challenging in practice. For a larger dataset, updating the pre-trained model instead of re-training can be computationally more efficient.

Remark 1. *The proposed methodology has the advantage of being highly modular. For example, when working with systems with small dimensional environmental variables (e.g., $z \in \mathbb{R}^2$ or $z \in \mathbb{R}^3$) a naïve random testing on the model dynamics is sufficient for generating the samples corresponding to $\mathcal{Z}_{risk}$. This can be substituted with a Bayesian inference pipeline and alternative sampling methods for testing in higher dimensional search space. Additionally, the methodology uses k-means clustering and selection of geometric centers for demonstrations of the true system. This is one of the several possible ways in which N data points for demonstration can be selected.*

IV. VALIDATIONS

A. Diffusion Policy on the Push-T setup

Problem Description: In this example, we consider the task of pushing a T-block to a fixed target region using a manipulator arm equipped with a circular end effector. The control policy used for the task is a diffusion policy that uses expert demonstrations to learn the score function of a diffusion model for predicting actions conditioned on observations [34]. The inputs generated by the policy π consist of (x_e, y_e), which corresponds to the (x, y) coordinates of the goal position of the circular end-effector. The policy is known to be robust to visual perturbations, and the implemented policy is trained in simulation using a PyMunk and Gym environment [33], that simulates the contact dynamics of the T-block and the end-effector in a 2D plane. The model dynamics, in this case, is non-differentiable, represents the interactions of the end-effector and the T-block, and that of

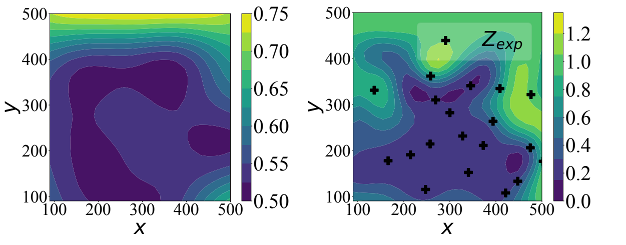

Fig. 4: Risk predictions for Push-T example from GPR trained only simulation data (left) and using the proposed methodology (right).

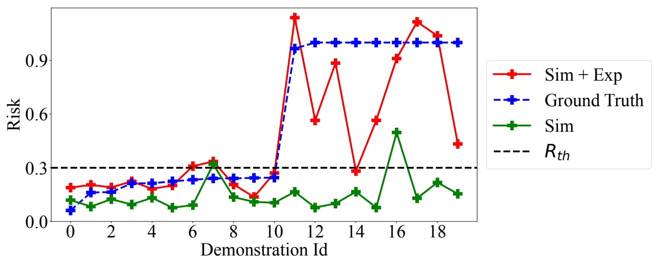

Fig. 5: Model prediction for Push-T on 20 data points where the predicted risk by the learned model ϕ_θ. The demonstration on true system marked as 'Ground Truth' illustrates the prediction to be accurate.

the T-block and the table, and does not take into account the kinematics and dynamics of the manipulator. Fig. 3a-Fig. 3b show the simulation environment and the corresponding hardware setup, respectively. The environment variable z was chosen as the initial position of the T-block ,i.e., $z = [x_b, y_b]$. The initial orientation of the block is kept fixed across all demonstrations to be the same as the target orientation of the block as shown in Fig. 3b.

Experimental setup: We use a UR3E Collaborative Robot Arm equipped with a Robotiq gripper to hold a 3D printed cylinder for the circular end-effector and a 3D printed T-block to construct the true system for the Push-T example. The diffusion policy in [34] is trained in simulation environment using pre-available dataset, where the workspace of the actual robot is not taken into consideration, and the end-effector is assumed to have only 2D motion in the 2D xy-plane. We used the Move-It package [41] to send commands to the manipulator for moving the end-effector to the desired goal location generated by the policy.

For this problem, we set the number of demonstrations to $N = 20$, with $N_1 = 10$ initial demonstrations. For discovering $\mathcal{Z}_{\text{risk}}$ from model dynamics, we chose 900 uniformly spaced samples of $z = [x_b, y_b]$ from the domain $[100, 500] \times [100, 500]$ for each value of $2 \times 10^{-5} \leq \sigma_1 \leq 2 \times 10^{-2}, 7 \times 10^{-6} \leq \sigma_2 \leq 7 \times 10^{-3}$. Fig. 1 (ii) shows the region $\mathcal{Z}_{\text{risk}}$ discovered in simulation using the model dynamics. The policy is trained using a reward $\gamma \in [0, 1]$ that measures the maximum amount of overlap between the fixed target region and the location of the T-block across a given rollout. We used $R = 1 - \gamma$ as the risk function, where $R \geq 0.3$ (or equivalently, $\gamma \leq 0.7$) is obtained when the T-block does not have any overlap with the target region, and hence $R_{\text{th}} = 0.3$ was set as the threshold for failure. The coverage function C was chosen as the variance over $[X_b^1, \ldots, X_b^T]$ given as $C(z, X_z) = \frac{1}{N} \sum_{t=1}^{T} \left(X_b^t - \sum_{t=1}^{T} \frac{X_b^t}{N} \right)^2$, where $X_b^t = [x_b^t, y_b^t]$ is the position of the T-block a time $0 < t \leq T$, normalized to be within $[0, 1] \times [0, 1]$. This enables selecting values of z that allow us to access a large part of the state space for each trajectory rollout, and thereby allow us to discover failures that are unseen in model dynamics. The coverage threshold was chosen as $C_{\text{th}} = 0.15$ based on the coverage values from a few exemplar rollouts.

Result analysis: Fig. 4 shows the contour for risk prediction using GPR with the proposed method labeled Sim+Exp (left) and using data collected with the model system only (right). The primary sources of sim-to-real gap in the Push-T example is the workspace constraints of the UR3E robot, which give rise to additional failures. This is not taken into account in the simulation environment and we discover these failures through the sequential demonstrations conducted using our approach in the paper, which leads to the difference in risk predictions using our method and model system only, as seen in Fig. 4.

For validating the learned failure prediction using the learned model ϕ_θ, we record the risk for 20 randomly sampled test demonstrations on the the true system, and compare against the predictions from our method. Table. I (Push-T) shows the risk prediction error with GPR using two methods, namely, data collected only using model dynamics (reported as Simulation) and data collected using our approach (reported as Simulation+Exp). Fig. 5 shows the individual predictions from our method (red), using data from model only (green) and their comparison against the ground truth (risk from true system, blue).

TABLE I: Comparison of Sim and Sim+Exp risk predictors.

Env	Mode	Sim	Sim + Exp
Push-T	Fail	11%	89%
	Not Fail	91%	82%
F1-Tenth	Fail	36%	100%
	Not Fail	100%	100%

B. LQR Speed+Steering control on the F1-Tenth car

Problem Description: In this example, we consider the problem of the F1-Tenth car tracking a reference race-line using a speed and steering LQR control. The policy is designed using a bicycle dynamic model with steering angle δ and acceleration a as inputs, and input constraints $|\delta| \leq 0.8$ rad, $|a| \leq 2ms^{-2}$. The environment variable in this case is a combination of a parameter for the reference path to be tracked, which is set to be a sinusoidal curve with variable width w, and reference speed at which the vehicle is supposed to move, v_{ref}, i.e., $z = [w, v_{\text{ref}}]$. We implement the policy on the F1-Tenth platform which is considered as the true system. Fig. 3c and Fig. 3d show an example reference trajectory being tracked by the model dynamics, and the corresponding hardware setup with the F1-Tenth vehicle denoting the true system, respectively.

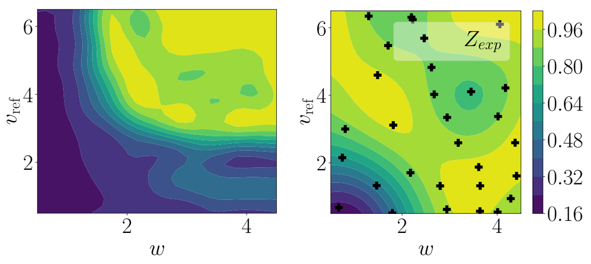

Fig. 6: Risk predictions for F1-Tenth example from GPR trained only simulation data (left) and using the proposed methodology (right).

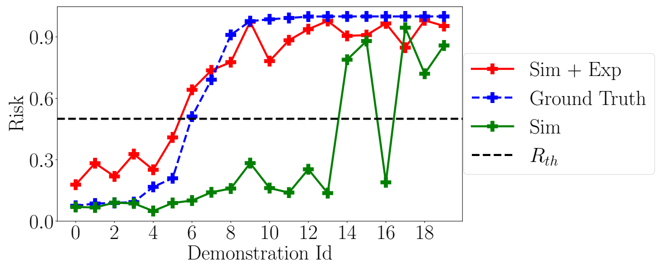

Fig. 7: Model prediction for F1-Tenth on 20 data points where the predicted risk by the learned model ϕ_θ. The demonstration on true system marked as 'Ground Truth' illustrates the prediction to be accurate.

Experimental Setup: The F1-Tenth vehicle used as the true system for hardware validation of the policy is known to have significant slip at high speeds $v \geq v_{\text{th}}$, and the proposed models for the vehicle in the literature consider a hybrid model to describe the lateral dynamics at high speeds, where the lateral dynamics plays an important role [42]. The performance of the vehicle at lower speeds is observed to be comparable to the bicycle model, however, the speed threshold v_{th} at which the effect of slip becomes significant is unknown for the vehicle. To quantify this effect and uncover other possible causes of failures, we test across a diverse set of reference paths, where the width of the path leads to variation in steering angle requirement, in addition to the variation in the reference speed.

We set the number of demonstrations to $N = 30$, with $N_1 = 20$ initial demonstrations. For pre-processing failure information from model dynamics, 900 uniformly spaced samples of (w, v_{ref}) are chosen from the domain $[0.5, 4.5] \times [0.5, 6.5]$ for each value of $10^{-4} \leq \sigma_1 \leq 1, 10^{-5} \leq \sigma_2 \leq 10^{-3}$. We choose a handcrafted risk function for this problem, consisting of the mean error (R_{mean}) and maximum error (R_{max}) between reference path and generated trajectory, and the minimum distance from the goal (R_{final}):

$$R(z) = 20R_{\text{mean}}(z) + R_{\text{max}}(z) + 10R_{\text{final}}(z) \qquad (11)$$

A failure in this context is defined as $R \geq R_{\text{th}}$, with $R_{\text{th}} = 11.5$. The weight of each of the component functions and the threshold is chosen based on the threshold values for the individual terms that led to failures. For subsequent training, the risk is normalized using a `sigmoid` such that the new threshold is $R_{\text{th}} = 0.5$. The coverage function C is chosen as the range of the steering angle across a given trajectory rollout, because presence of steering angle limit is observed to be a dominant cause of failure in model dynamics, and a larger steering angle is required as the curvature of the race-line and reference speed increase. Therefore, for steering angle at time t given by δ_t, the coverage function can be expressed as $C(z, X_z) = \max_{t\in[1,T]} \delta_t - \min_{t\in[1,T]} \delta_t$. The coverage threshold in this case was chosen as $C_{\text{th}} = 1.0$ rad, which is marginally above the steering angle limit.

Result analysis: Fig. 6 (right) shows the predicted risk contours using the trained model with the proposed method (Sim + Exp), while Fig. 6 (left) shows the predicted risk with a GPR model trained with data collected only from the model dynamics (Sim). Both the predictions show failure for high speed and high lane width, as the required steering angle increases as the speed or lane width increases. However, the threshold speed at which failure is observed in the true system is much lower, because the policy generates inputs without taking the lateral dynamics into account, which becomes increasingly prominent as the speed increases. This causes increasing tracking error, coupled with increasing steering angle requirement causes the failure region to expand substantially in the true system.

Fig. 7 shows the predicted risk using the proposed method (red) and using just the model dynamics (green), compared against the ground truth risk (calculated using (11)) obtained from the true system for 20 randomly chosen z. Our method accurately predicts 'fail' and 'not fail' in all the 20 cases. While the model trained only using model dynamics predicts the 'not fail' scenarios accurately, it is unable to accurately forecast the 'fail' scenarios correctly (see Table. I (F1-Tenth)), as it overestimates the 'not fail' region. The video corresponding to all hardware experiments can be found at the project website[2]

V. Conclusions & Future Work

In this work, we considered the problem of discovering failures in a system that emerge due to the sim-to-real gap. As a solution, we propose to use the available data from the simulation, which we refer to as the model dynamics, in addition to limited demonstrations from the true system. This allows us to leverage the benefits of simulation-based testing and hardware testing, while also maintaining the budget for hardware experimentation and overcoming the limitations arising from the lack of information. Our proposed method successfully predicts failures that cannot be captured by model dynamics alone, as illustrated on two different platforms, namely a UR3E manipulator arm and the F1-Tenth autonomous racing platform. Our experiments quantify the sim-to-real gap in the considered examples where the simulation-based failure predictor has a far less accuracy (11% and 36% in the two examples) than the one obtained from the proposed method (89% and 100%, respectively).

The proposed method deals with failure defined on the exogenous system parameters. In our future work, we will explore more general failure representation that are not scenario specific, e.g., set of failures defined on the state

[2]https://mit-realm.github.io/few-demo/

and input space. We are also interested in exploring how to integrate the proposed failure prediction methodology with existing risk-aware decision-making pipelines, that require knowledge of failure scenarios to prevent them.

References

[1] T. Dreossi, T. Dang, A. Donzé, J. Kapinski, X. Jin, and J. V. Deshmukh, “Efficient guiding strategies for testing of temporal properties of hybrid systems,” in *NASA Formal Methods: 7th International Symposium, NFM 2015, Pasadena, CA, USA, April 27-29, 2015, Proceedings 7*. Springer, 2015, pp. 127–142.

[2] J. M. Esposito, J. Kim, and V. Kumar, “Adaptive RRTs for validating hybrid robotic control systems,” in *Algorithmic foundations of robotics vi*. Springer, 2005, pp. 107–121.

[3] A. Corso, R. Lee, and M. J. Kochenderfer, “Scalable autonomous vehicle safety validation through dynamic programming and scene decomposition,” in *2020 IEEE 23rd International Conference on Intelligent Transportation Systems (ITSC)*. IEEE, 2020, pp. 1–6.

[4] A. Corso, P. Du, K. Driggs-Campbell, and M. J. Kochenderfer, “Adaptive stress testing with reward augmentation for autonomous vehicle validatio,” in *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*. IEEE, 2019, pp. 163–168.

[5] 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*, vol. 33, pp. 6402–6416, 2020.

[6] L. Yang and N. Ozay, “Synthesis-guided Adversarial Scenario Generation for Gray-box Feedback Control Systems with Sensing Imperfections,” *ACM Transactions on Embedded Computing Systems*, vol. 20, no. 5s, pp. 102:1–102:25, Sep. 2021.

[7] P. Donti, A. Agarwal, N. V. Bedmutha, L. Pileggi, and J. Z. Kolter, “Adversarially robust learning for security-constrained optimal power flow,” in *Advances in Neural Information Processing Systems*, vol. 34. Curran Associates, Inc., 2021, pp. 28 677–28 689.

[8] M. Cavorsi, O. E. Akgün, M. Yemini, A. J. Goldsmith, and S. Gil, “Exploiting trust for resilient hypothesis testing with malicious robots,” in *2023 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2023, pp. 7663–7669.

[9] U. Ghai, D. Snyder, A. Majumdar, and E. Hazan, “Generating Adversarial Disturbances for Controller Verification,” in *Proceedings of the 3rd Conference on Learning for Dynamics and Control*. PMLR, May 2021, pp. 1192–1204.

[10] C. Xu, W. Ding, W. Lyu, Z. Liu, S. Wang, Y. He, H. Hu, D. Zhao, and B. Li, “SafeBench: A Benchmarking Platform for Safety Evaluation of Autonomous Vehicles,” *Advances in Neural Information Processing Systems*, vol. 35, pp. 25 667–25 682, Dec. 2022.

[11] Y. Annpureddy, C. Liu, G. Fainekos, and S. Sankaranarayanan, “S-TaLiRo: A Tool for Temporal Logic Falsification for Hybrid Systems,” in *Tools and Algorithms for the Construction and Analysis of Systems*, ser. Lecture Notes in Computer Science, P. A. Abdulla and K. R. M. Leino, Eds. Berlin, Heidelberg: Springer, 2011, pp. 254–257.

[12] G. Chou, Y. E. Sahin, L. Yang, K. J. Rutledge, P. Nilsson, and N. Ozay, “Using control synthesis to generate corner cases: A case study on autonomous driving,” *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 37, no. 11, pp. 2906–2917, Nov. 2018.

[13] 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.

[14] 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

[15] C. Dawson, A. Parashar, and C. Fan, “Beyond adversarial examples: sampling and repairing diverse failures with radium.”

[16] A. Parashar, J. Yin, C. Dawson, P. Tsiotras, and C. Fan, “Learning-based bayesian inference for testing of autonomous systems,” *IEEE Robotics and Automation Letters*, pp. 1–8, 2024.

[17] C. Dawson and C. Fan, “Robust counterexample-guided optimization for planning from differentiable temporal logic,” in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2022, pp. 7205–7212.

[18] 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.

[19] 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?” *Bernoulli*, vol. 27, no. 3, pp. 1942 – 1992, 2021.

[20] C. Xu, D. Zhao, A. Sangiovanni-Vincentelli, and B. Li, “Diffscene: Diffusion-based safety-critical scenario generation for autonomous vehicles,” in *The Second Workshop on New Frontiers in Adversarial Machine Learning, ICML*, 2023.

[21] P. Koopman and M. Wagner, “Autonomous vehicle safety: An interdisciplinary challenge,” *IEEE Intelligent Transportation Systems Magazine*, vol. 9, no. 1, pp. 90–96, 2017.

[22] N. Hanselmann, K. Renz, K. Chitta, A. Bhattacharyya, and A. Geiger, “KING: Generating Safety-Critical Driving Scenarios for Robust Imitation via Kinematics Gradients,” in *Computer Vision – ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part XXXVIII*. Berlin, Heidelberg: Springer-Verlag, Oct. 2022, pp. 335–352.

[23] 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.

[24] 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.

[25] 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.

[26] G. Chou, Y. E. Sahin, L. Yang, K. J. Rutledge, P. Nilsson, and N. Ozay, “Using control synthesis to generate corner cases: A case study on autonomous driving,” *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 37, no. 11, pp. 2906–2917, 2018.

[27] H. Sun, S. Feng, X. Yan, and H. X. Liu, “Corner Case Generation and Analysis for Safety Assessment of Autonomous Vehicles,” *Transportation Research Record*, vol. 2675, no. 11, pp. 587–600, Nov. 2021.

[28] 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*, vol. 19, no. 3, pp. 733–744, 2017.

[29] 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*, vol. 31. Curran Associates, Inc., 2018.

[30] A. Z. Ren, H. Dai, B. Burchfiel, and A. Majumdar, “Adaptsim: Task-driven simulation adaptation for sim-to-real transfer,” in *Conference on Robot Learning*. PMLR, 2023, pp. 3434–3452.

[31] S. Iwazaki, S. Takeno, T. Tanabe, and M. Irie, “Failure-aware gaussian process optimization with regret bounds,” in *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. [Online]. Available: https://openreview.net/forum?id=H5pwAeYAun

[32] D. Nguyen-Tuong, J. Peters, and M. Seeger, “Local gaussian process regression for real time online model learning,” *Advances in neural information processing systems*, vol. 21, 2008.

[33] P. Florence, C. Lynch, A. Zeng, O. A. Ramirez, A. Wahid, L. Downs, A. Wong, J. Lee, I. Mordatch, and J. Tompson, “Implicit behavioral cloning,” in *Conference on Robot Learning*. PMLR, 2022, pp. 158–168.

[34] C. Chi, S. Feng, Y. Du, Z. Xu, E. Cousineau, B. Burchfiel, and S. Song, “Diffusion policy: Visuomotor policy learning via action diffusion,” in *Proceedings of Robotics: Science and Systems (RSS)*, 2023.

[35] A. Sakai, D. Ingram, J. Dinius, K. Chawla, A. Raffin, and A. Paques, “PythonRobotics: a Python code collection of robotics algorithms,” *arXiv preprint arXiv:1808.10703*, 2018.

[36] M. O’Kelly, V. Sukhil, H. Abbas, J. Harkins, C. Kao, Y. V. Pant, R. Mangharam, D. Agarwal, M. Behl, P. Burgio *et al.*, “F1/10: An open-source autonomous cyber-physical platform,” *arXiv preprint arXiv:1901.08567*, 2019.

[37] Y. Zhou, S. Booth, N. Figueroa, and J. Shah, “Rocus: Robot controller understanding via sampling,” in *Conference on Robot Learning*. PMLR, 2022, pp. 850–860.

[38] P. Izmailov, P. Kirichenko, M. Finzi, and A. G. Wilson, “Semi-supervised learning with normalizing flows,” in *International conference on machine learning*. PMLR, 2020, pp. 4615–4630.

[39] L. Dinh, J. Sohl-Dickstein, and S. Bengio, “Density estimation using real NVP,” in *International Conference on Learning Representations*, 2017. [Online]. Available: https://openreview.net/forum?id=HkpbnH9lx

[40] C. P. Robert and G. Casella, *The Metropolis—Hastings Algorithm*. New York, NY: Springer New York, 2004, pp. 267–320.

[41] S. Chitta, I. Sucan, and S. Cousins, “Moveit![ros topics],” *IEEE Robotics & Automation Magazine*, vol. 19, no. 1, pp. 18–19, 2012.

[42] M. Althoff, M. Koschi, and S. Manzinger, “Commonroad: Composable benchmarks for motion planning on roads,” in *2017 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 2017, pp. 719–726.