2023 IEEE International Conference on Robotics and Automation (ICRA 2023)
May 29 - June 2, 2023. London, UK

Safety-Critical Controller Verification via Sim2Real Gap Quantification

Prithvi Akella, Wyatt Ubellacker, and Aaron D. Ames[1]

***Abstract*— The well-known quote from George Box states that: "All models are wrong, but some are useful." To develop more useful models, we quantify the inaccuracy with which a given model represents a system of interest, so that we may leverage this quantity to facilitate controller synthesis and verification. Specifically, we develop a procedure that identifies a sim2real gap that holds with a minimum probability. Augmenting the nominal model with our identified sim2real gap produces an uncertain model which we prove is an accurate representor of system behavior. We leverage this uncertain model to synthesize and verify a controller in simulation using a probabilistic verification approach. This pipeline produces controllers with an arbitrarily high probability of realizing desired safe behavior on system hardware without requiring hardware testing except for those required for sim2real gap identification. We also showcase our procedure working on two hardware platforms - the Robotarium and a quadruped.**

I. Introduction

The nominal controller synthesis process for safety-critical systems follows a well-worn path: develop a model for the system of interest (from first principles, system identification, or otherwise), develop a controller in simulation based on this model, implement the controller on the safety-critical system of interest, and, most likely, tune controller parameters until the system exhibits the desired behavior. In what will follow, we offer a method that aims to mitigate the need for extensive - and perhaps expensive - hardware testing and verification, by simultaneously verifying the simulator used for controller synthesis and the controller itself. The hope is that such a method will augment existing model generation techniques to streamline the controller synthesis and verification procedure.

Our desire to augment existing model generation techniques stems from a desire to leverage extensive prior work in system identification and reduced-order-model control. System identification specifically deals with techniques aimed at generating models that more closely align with the system-to-be-modeled [1]–[4]. This has led to the development of state-of-the-art methods for this process, *e.g.* the Volterra and NARMAX methods [5]–[9]. There have also been more recent efforts at representing systems via Neural Networks [10]–[12] or regressing dynamical models via Gaussian Process Regression [13]–[15]. Additionally, regardless of the method used, there are techniques to progressively make better models that more closely align with their corresponding systems.

Despite the capability of existing methods to generate good models, they oftentimes fail to capture rarer system behavior, *i.e.* corner cases, stochastic disturbances, *etc*. Accounting for such behavior underlies the study into stochastic nonlinear models for system identification [16], [17]. While very expressive, these models tend to be quite computationally complex. Moreover, prior work indicates that reduced order models are oftentimes sufficient to express underlying system physics [18]–[20]. These simpler models are also often leveraged in autonomy stacks for complex systems, *e.g.* as in [21], where the authors leverage a system model and *apriori* known disturbance bounds to generate safe, efficient planned trajectories via an offline HJB computation. Generally speaking, augmenting these simpler models with uncertainty bounds and accounting for these bounds through robust control [19], [22], input-to-state stabilizing barrier functions [23]–[25], or other techniques, *i.e.* learning [26]–[28], tends to yield safe and reliable controllers. This prompts the question then: how should the "correct" uncertainty bound be determined with respect to the model we aim to utilize? If we use too large a bound, and our system could be conservative, but a small bound could yield unsafe behavior.

Our Contribution: Our work aims to address this question of determining the magnitude of uncertainty one should

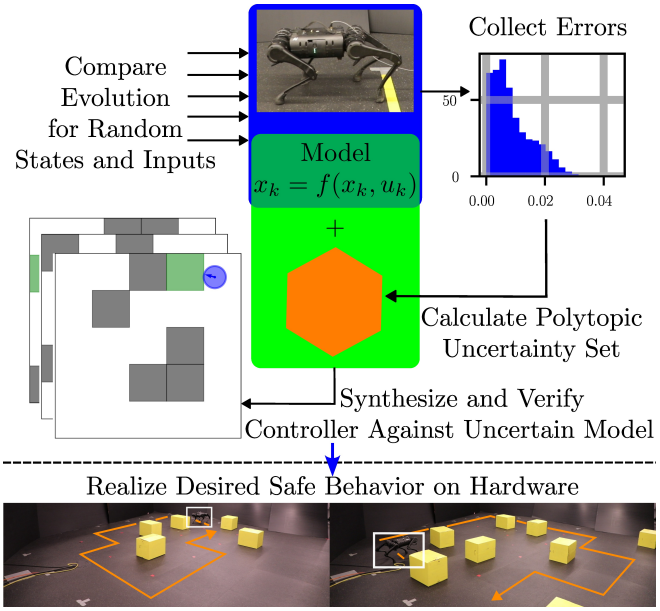

Fig. 1. An overview of our proposed approach. Models are often inaccurate despite best practices to generate accurate models. So, we propose a method to probabilistically determine an uncertainty set to augment our nominal model. Doing so produces an uncertain model that we prove accurately represents true system behavior to an arbitrarily high probability. This allows for simultaneous controller synthesis and verification in simulation while retaining safe performance with limited testing on hardware. (Quadruped highlighted in white in bottom figures)

This work was supported by the AFOSR Test and Evaluation Program, grant FA9550-19-1-0302 and Dow (#227027AT)

[1]Authors are with the California Institute of Technology (email {pakella, wubellac, ames}@caltech.edu)

consider for a provided model that will be used in controller synthesis. In this vein, our results are two-fold. First, we provide a norm bound on the uncertainty between a provided model and its corresponding system. This bound effectively reads as: with minimum probability $1-\epsilon$, the evolution of the system at one time-step will lie within the bounding region prescribed by the evolution of the provided model plus an uncertainty lying within a polytopic set we calculate. Our second result makes use of this uncertainty bound and the authors' prior work in stochastic verification [29], [30] to verify the generated controller against the uncertain model. This results in a pipeline for safety-critical controller synthesis and verification that translates to hardware performance.

Implementation: We showcase our contributions on two hardware platforms, the Georgia Tech Robotarium [31] and a quadruped. For both platforms, we leverage our first result to determine their discrepancies with respect to a nominal unicycle model. Then, we use this discrepancy to synthesize and verify a controller, purely in simulation, that is guaranteed to successfully steer the uncertain unicycle model in a navigation and static obstacle avoidance scenario. For the Robotarium, we also add multiple, uncontrolled, stochastically evolving agents. For both systems, we then show that these verified controllers exhibit similar levels of performance on hardware without any parameter tuning or testing required - effectively working "out-of-the-box" after our procedure. Furthermore, these controllers successfully steered their systems to satisfy their control objectives despite a wide variety of randomized test scenarios.

Paper Structure: First, we provide the necessary mathematical background for our procedure in Section II and formally define the sim2real gap in Section III-A. We describe our approach to sim2real gap quantification in Section III-B and our controller verification procedure in Section IV. Finally, we detail all experimental demonstrations of our procedure in Section V.

II. A Brief Review of Scenario Optimization

In short, both aspects of our procedure arise from separate uses of the scenario optimization procedure [32], [33]. Scenario optimization identifies robust solutions to uncertain convex optimization problems of the following form:

$$\begin{aligned} \mathrm{v}^* = \underset{\mathrm{v}\in\mathbb{V}\subset\mathbb{R}^d}{\operatorname{argmin}} \quad & c^T\,\mathrm{v}, \\ \text{subject to} \quad & \mathrm{v}\in\mathbb{V}_\delta,\ \delta\in\Delta. \end{aligned} \tag{UP}$$

Here, (UP) is the uncertain program as $\delta \in \Delta$ is a sample of some random variable in the probability space $\Sigma = (\Omega, \mathcal{F}, \mathbb{P})$, with sample space $\Omega = \Delta$, (perhaps) unknown event space $\mathcal{F}$, and (perhaps) unknown probability measure $\mathbb{P}$. Convexity is assured via assumed convexity in the spaces $\mathbb{V}$ and $\mathbb{V}_\delta$. Furthermore, since Δ is typically a set of infinite cardinality, *i.e.* $|\Delta| = \infty$, identification of a solution v^* such that $\mathrm{v}^* \in \mathbb{V}_\delta \ \forall\ \delta \in \Delta$ is hard.

To resolve this issue, the study of scenario optimization solves a related optimization problem formed from an N-sized sample set of the random constraint samples δ and provides a probabilistic guarantee on the robustness of the corresponding solution v^*_N. Specifically, if we were to take an N-sized set of samples $\{\delta_\mathrm{i}\}_{\mathrm{i}=1}^{\mathrm{N}}$, we could construct the following scenario program:

$$\begin{aligned} \mathrm{v}^*_N = \underset{\mathrm{v}\in\mathbb{V}\subset\mathbb{R}^d}{\operatorname{argmin}} \quad & c^T\,\mathrm{v}, \\ \text{subject to} \quad & \mathrm{v}\in\bigcap_{\delta_i\in\{\delta_j\}_{j=1}^N}\mathbb{V}_{\delta_i}. \end{aligned} \tag{RP-N}$$

Then, we require the following assumption.

Assumption 1. The scenario program (RP-N) is solvable for any N-sample set $\{\delta_\mathrm{i}\}_{\mathrm{i}=1}^{\mathrm{N}}$ and has a unique solution v^*_N.

Assumption 1 then guarantees existence of a scenario solution v^*_N for (RP-N) for any provided sample set $\{\delta_\mathrm{i}\}_{\mathrm{i}=1}^{\mathrm{N}}$. As such, we can define a set containing those samples $\delta \in \Delta$ to which the scenario solution v^*_N is not robust, *i.e.* $F(\mathrm{v}) = \{\delta \in \Delta \mid \mathrm{v} \notin \mathbb{V}_\delta\}$. With this set definition we can formally define the *violation probability* of our solution.

Definition 1. The *violation probability* $V(\mathrm{v})$ of a given $\mathrm{v} \in \mathbb{V}$ is defined as the probability of sampling a constraint δ to which v is not robust, *i.e.* $V(\mathrm{v}) = \mathbb{P}[\delta \in F(\mathrm{v})]$.

Then, the main theorem is as follows:

Theorem 1 (Adapted from Theorem 1 in [32])**.** *Let Assumption 1 hold. The following inequality is true:*

$$\mathbb{P}^N[V(\mathrm{v}^*_N) > \epsilon] \leq \sum_{i=0}^{d-1}\binom{N}{i}\epsilon^i(1-\epsilon)^{N-i}.$$

In Theorem 1 above, N is the number of sampled constraints δ for the scenario program (RP-N); v^*_N is the scenario solution to the corresponding scenario program; $V(\mathrm{v}^*_N)$ is the violation probability of that solution as per Definition 1; d is the dimension in which v lies, *i.e.* $\mathrm{v} \in \mathbb{R}^d$; and $\mathbb{P}^N$ is the induced probability measure over sets of N-samples of δ given the probability measure $\mathbb{P}$ for δ.

III. Probabilistic Sim2Real Gap Quantification

Our method for sim2real gap quantification will express the identification of such a gap as a convex optimization problem with (perhaps) infinite constraints, and we will take a scenario approach to solve this problem. This approach will yield a robust result - a large enough sim2real gap - that holds with some minimum probability. The next section will formally define what we mean by a sim2real gap.

A. Defining the Gap

First, we denote our true system via x and our nominal model via $\hat{x}$, *i.e.* $\forall\ k, j = 0, 1, 2, \ldots$, with true system time $t_{jK} = \hat{t}_j$ where $\hat{t}$ corresponds to the simulator time, and $K > 0$ a time-dilation factor (an example will be provided):

$$\begin{aligned} &\textbf{True:} \quad x_{k+1} = f(x_k, u_k),\ x_{k,k+1} \in \mathcal{X},\ u_k \in \mathcal{U}, \\ &\textbf{Sim:} \quad \hat{x}_{j+1} = \hat{f}(\hat{x}_j, \hat{u}_j),\ \hat{x}_{j,j+1} \in \hat{\mathcal{X}},\ \hat{u}_j \in \hat{\mathcal{U}}. \end{aligned} \tag{SYS}$$

As an example consistent with the demonstrations to follow, the true system could be a quadruped with the nominal model a unicycle system we aim to represent it with.

To provide a method of comparing the evolution of the two systems in (SYS), we will define two maps - M_x which projects the true system state x to the model state $\hat{x}$, and M_u which extends the model input $\hat{u}$ to the true system input u:

$$M_x : \mathcal{X} \to \hat{\mathcal{X}}, \quad M_u : \hat{\mathcal{U}} \times \mathcal{X} \to \mathcal{U}. \tag{MAPS}$$

These maps in (MAPS) let us formalize the gap we aim to identify between the two systems. First, we assume that we can command an input u to the true system by prescribing an input $\hat{u}$ that we would provide to our associated model. Then, we assume we can measure the projected true system state $M_x(x_k)$ at some time-step K, *i.e.*, $\forall\ k = 0, 1, \ldots, K$,

$$x_{k+1} = f(x_k, M_u(\hat{u}, x_k)),\ O(x_0, \hat{u}) = M_x(x_K). \tag{OBS}$$

To put this in the context of the quadruped/unicycle model example, the underlying control loop operates at 1 kHz and provides a natural discrete abstraction at that time-step. Since, we desire our unicycle model to update at 10 Hz then, $K = 100$. M_x is just the projection of the quadruped state to its unicycle components. Likewise, M_u is the underlying control loop that runs at 1 kHz to realize the commanded forward walking speed and rotation. While these maps seem abstract, we will provide examples in Section V.

This observation map O in (OBS) permits us to quantify a discrepancy between model evolution and observed true system evolution. However, we only want to make this comparison when the projection of the initial state $M_x(x_0) \in \hat{\mathcal{X}}$ - as otherwise, the projected initial state is not addressed by our representative model. This results in the following space definition and problem statement:

$$\Pi(M_x) = \left\{ x \in \mathcal{X} \mid M_x(x) \in \hat{\mathcal{X}} \right\} \tag{1}$$

Definition 2. Let O be as defined in (OBS), $\hat{f}, \mathcal{X}, \hat{\mathcal{U}}$ be as defined in (SYS), and $\Pi(M_x)$ be as defined in (1). The *sim2real gap* $\Lambda \in \mathbb{R}$ is such that $\forall\ (x_0, \hat{u}) \in \Pi(M_x) \times \hat{\mathcal{U}}$,

$$\Lambda \geq \left\| O(x_0, \hat{u}) - \hat{f}\left(M_x(x_0), \hat{u}\right) \right\|. \tag{2}$$

B. Quantifying the Gap

To start, we express identification of the sim2real gap Λ in (2) as an optimization problem:

$$\begin{aligned} \Lambda = \underset{r \in \mathbb{R}}{\operatorname{argmin}} \quad & r, \\ \text{subject to} \quad & r \geq \left\| O(x_0, \hat{u}) - \hat{f}\left(M_x(x_0), \hat{u}\right) \right\|, \\ & \ldots \forall\ (x_0, \hat{u}) \in \Pi(M_x) \times \hat{\mathcal{U}} \end{aligned} \tag{3}$$

This problem is (likely) impossible to solve as posed, as it may have infinite constraints. So, taking inspiration from prior work, we aim instead to find a "good" solution to (3) via a randomized, sample-based approach (See Section 5 in [30]). This solution will hold with minimum probability $\epsilon \in [0, 1)$, but we can make $\epsilon \to 1$ with enough samples.

Description of our Approach: Our approach hinges on the ability to independently draw state and input samples from a static distribution π over the combined state and input spaces $\mathcal{X} \times \hat{\mathcal{U}}$. In the experimental demonstrations to follow, we will argue and show evidence that our chosen method produces independent samples from an unknown distribution π. However, the specifics of crafting such a distribution will likely be different for different system/model pairs - this is where we anticipate a large portion of future work in this vein to lie. So, for the moment, we will simply assume the existence of such a distribution π formalized as follows:

Definition 3. The *comparison distribution* π maps subsets of the combined state and model input space $\mathcal{X} \times \hat{\mathcal{U}}$ - as defined in (SYS) - to $[0, 1]$ *i.e.* $\forall\ A \subseteq \mathcal{X} \times \hat{\mathcal{U}},\ \pi(A) \in [0, 1]$. Furthermore, π "covers" the space of states and inputs generating constraints for (3), *i.e.*, $\pi\left(\Pi(M_x) \times \hat{\mathcal{U}}\right) = 1$.

Using this comparison distribution π, we can construct a scenario program for sets of N samples $\left\{\left(x_0^l, \hat{u}^l\right)\right\}_{l=1}^N$ of state and input pairs $(x_0, \hat{u})$ distributed by π:

$$\begin{aligned} \Lambda_N^* = \underset{r \in \mathbb{R}}{\operatorname{argmin}} \quad & r, \\ \text{subject to} \quad & r \geq \left\| O(x_0^j, \hat{u}) - \hat{f}\left(M_x(x_0), \hat{u}^j\right) \right\|, \\ & \ldots \forall\ (x_0^j, \hat{u}^j) \in \left\{\left(x_0^l, \hat{u}^l\right)\right\}_{l=1}^N. \end{aligned} \tag{4}$$

The resulting solution Λ_N^* is an ever-increasing lower bound on Λ as expressed in the following theorem.

Theorem 2. *Let Λ_N^* be the solution to* (4) *with O as defined in* (OBS)*, $\hat{f}$ as defined in* (SYS)*, and π as defined in Definition 3. Then, $\forall\ \epsilon \in [0, 1]$*

$$S_1 \triangleq \mathbb{P}_\pi\left[\Lambda_N^* \geq \left\| O(x_0, \hat{u}) - \hat{f}\left(M_x(x_0), \hat{u}\right) \right\|\right],$$
$$\mathbb{P}_\pi^N\left[S_1 \geq 1 - \epsilon\right] \geq 1 - (1 - \epsilon)^N.$$

In other words, Λ_N^* is larger than the sim2real gap for any sampled state and input pair $(x_0, \hat{u})$ from π with minimum probability $1 - \epsilon$ and confidence $1 - (1 - \epsilon)^N$.

Proof: This proof follows almost directly from Theorem 1, noting that the violation probability,

$$V(\Lambda_N^*) = 1 - S_1.$$

Then, reversing inequalities and changing arguments appropriately provides the desired result. ∎

In other words, Theorem 2 states that the solution Λ_N^* to (4) is a decent approximation of the sim2real gap Λ.

IV. Safety-Critical Controller Verification

Now, we can leverage this approximate sim2real gap Λ_N^* to facilitate controller synthesis and verification in simulation. This section details those efforts.

Constructing a Valid Uncertain Model: To start, we can use the resulting probabilistic sim2real gap Λ_N^* from (4) to augment our nominal model $\hat{x}$ in (SYS) and define an uncertain system denoted via $\tilde{x}$. Specifically, we will first define a feasible space of disturbances, with 2-norm $\|\cdot\|$:

$$D = \{d \in \mathbb{R}^n \mid \|d\| \leq \Lambda_N^* \ \text{(as per (4))}\}, \tag{5}$$

and use D to define our uncertain system:

$$\tilde{x}_{j+1} = \hat{f}(\tilde{x}_j, \hat{u}_j) + d_j,\ d_j \sim \mathrm{U}[D], \tag{6}$$

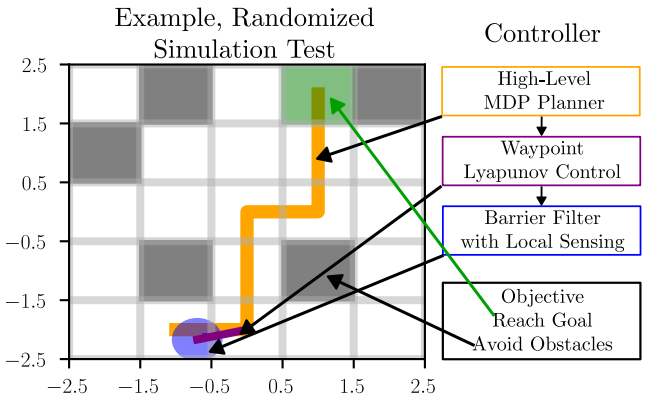

Fig. 2. As part of our pipeline, we verify controllers against our uncertain model. Shown above is an example, randomized test scenario for controller development for the quadruped. The test scheme remains the same for the two different systems (Robotarium and quadruped) as their controller objectives are similar. This information is further explained in Section V.

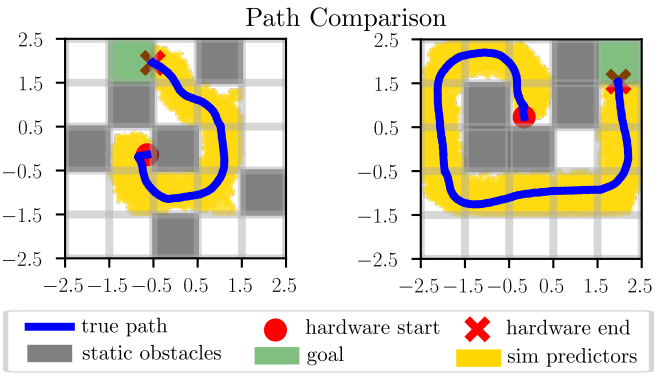

Fig. 3. Our procedure increases the confidence that those controllers that pass the verification step will exhibit similar performance on hardware as they did in simulation, even if we did not directly verify the controller on hardware. This increased confidence arises through our verification of the uncertain model, whose reachable set we prove encapsulates true system evolution to high probability. This can be seen in the figures above, as the quadruped's evolution (blue) lies within its associated uncertain simulator's predictions (gold). This information is further explained in Section V.

with $\mathrm{U}[D]$ the uniform distribution over D. If we define a one-step reachable space provided a model state and input $(\hat{x}, \hat{u}) \in \hat{\mathcal{X}} \times \hat{\mathcal{U}}$,

$$\mathcal{R}(\hat{x}, \hat{u}) = \left\{ \hat{f}(\hat{x}, \hat{u}) + d, \ \forall \ d \in D \right\}, \tag{7}$$

then we have the following result regarding the evolution of the true system and this reachable space.

Corollary 1. *Let O be as defined in* (OBS)*, $\mathcal{R}$ be as defined in* (7)*, M_x be as defined in* (MAPS)*, π be as per Definition 3, and Λ_N^* be as defined in* (4)*. Then, $\forall\ \epsilon \in [0, 1]$,*

$$S_2 \triangleq \mathbb{P}_\pi \left[O(x_0, \hat{u}) \in \mathcal{R}(M_x(x_0), \hat{u}) \right],$$
$$\mathbb{P}_\pi^N \left[S_2 \geq 1 - \epsilon \right] \geq 1 - (1 - \epsilon)^N.$$

In other words, the probability that the observed evolution of the true system $O(x_0, \hat{u})$ lies in the reachable space of our uncertain model $\mathcal{R}(M_x(x_0), \hat{u})$ is at least $1 - \epsilon$ with confidence $1 - (1 - \epsilon)^N$.

Proof: This is a direct application of Theorem 2, as for any state and input pair $(x_0, \hat{u}) \in \mathcal{X} \times \hat{\mathcal{U}}$, $\Lambda_N^* \geq \left\| O(x_0, \hat{u}) - \hat{f}\left(M_x(x_0), \hat{u}\right) \right\|$ iff $O(x_0, \hat{u}) \in \mathcal{R}(M_x(x_0), \hat{u})$, by definition of $\mathcal{R}$ in (7) and D in (5). ■

In other words, Corollary 1 tells us that even though a single step of our uncertain model (6) may not be an accurate representation of our true system's evolution, the space of all possible single-step evolutions does, to high probability, contain the evolution of our true system at the next time-step.

For controller synthesis and verification then, Corollary 1 tells us that our uncertain model is a decent approximator of true system behavior. Therefore, any controller that exhibits good performance on the uncertain model, should likewise exhibit good performance on the true system. Quantification of "good" performance and verifying a controller's ability to realize "good" performance is the subject of the next subsection, which follows from the risk-aware probabilistic verification procedure detailed in Section 3 in [29].

Verifying against Safety Metrics: First, provided a parameterized controller $\hat{\mathrm{U}} : \hat{\mathcal{X}} \times \Theta \to \hat{\mathcal{U}}$ with parameter $\theta \in \Theta$ and noise sequence ξ where $\xi_j = d \sim \mathrm{U}[D]$, we define ϕ to be the closed-loop trajectory to our uncertain model (6), *i.e.* with $J > 0$ and $\tilde{x}_0 = \hat{x}_0$,

$$\phi^{\hat{\mathrm{U}}}(\hat{x}_0, \xi, \theta, J) = \tilde{x}_J, \ \tilde{x}_{j+1} = \hat{f}\left(\tilde{x}_j, \hat{\mathrm{U}}(\tilde{x}_j, \theta)\right) + \xi_j. \tag{8}$$

Second, inspired by traditional safety measures, *e.g.* barrier functions over the system state, we define safety metrics h to be functions over system trajectories that only output positive numbers for trajectories exhibiting the desired safe behavior:

$$\mathrm{h}\left(\phi^{\hat{\mathrm{U}}}\left(\hat{x}_0, \xi, \theta\right)\right) \geq 0 \iff \begin{matrix} \phi^{\hat{\mathrm{U}}}\left(\hat{x}_0, \xi, \theta\right) \text{ exhibits} \\ \text{desired safe behavior.} \end{matrix} \tag{9}$$

Examples of safety metrics include robustness measures from Signal Temporal Logic [34] or the minimum value of a barrier function over a finite-time horizon [35].

Ideally, we would like for our controller $\hat{\mathrm{U}}$ to only ever realize trajectories $\phi^{\hat{\mathrm{U}}}(\hat{x}_0, \xi, \theta)$ with a positive evaluation under this safety metric h. To check for this positivity, we will first draw $(\hat{x}_0, \theta)$ uniformly from $\hat{\mathcal{X}} \times \Theta$. Then, we will evaluate the safety of one trajectory emanating from that initial condition $\hat{x}_0$ with that parameter θ, *i.e.* record $s = \mathrm{h}\left(\phi^{\hat{\mathcal{U}}}\left(\hat{x}_0, \xi, \theta\right)\right)$ for some valid noise-sequence ξ. Repeating this procedure N times to take N such samples s_i and create the dataset $\{s_i\}_{i=1}^N$, we then define $s_N^* = \min\{s_i\}_{i=1}^N$. Then, via Corollary 2 in [29], we have the following result:

Corollary 2. *Let the uncertain system trajectory $\phi^{\hat{\mathrm{U}}}(\hat{x}_0, \xi, \theta)$ be as defined in* (8)*, the safety metric h be as defined in* (9)*, $\{s_i\}_{i=1}^N$ be the safety values of N sampled trajectories $\phi^{\hat{\mathrm{U}}}(\hat{x}_0, \xi, \theta)$ with initial conditions and parameters (x_0, θ) drawn from $\mathrm{U}[\hat{\mathcal{X}} \times \Theta]$, and $s_N^* = \min\{s_i\}_{i=1}^N$. Then, $\forall\ \epsilon \in [0, 1]$ and abbreviating $\mathrm{U}[\hat{\mathcal{X}} \times \Theta] = \pi_0$,*

$$S_3 \triangleq \mathbb{P}_{\pi_0,\ \xi_j \sim \mathrm{U}[D]\ \forall\ j=1,2,\ldots} \left[s \geq s_N^* \right],$$
$$\mathbb{P}_{\pi_0,\ \xi_j \sim \mathrm{U}[D]\ \forall\ j=1,2,\ldots}^N \left[S_3 \geq 1 - \epsilon \right] \geq 1 - (1 - \epsilon)^N.$$

In other words, the probability that s_N^* will be smaller than any sample-able safety value s is at minimum $1 - \epsilon$ with confidence $1 - (1 - \epsilon)^N$.

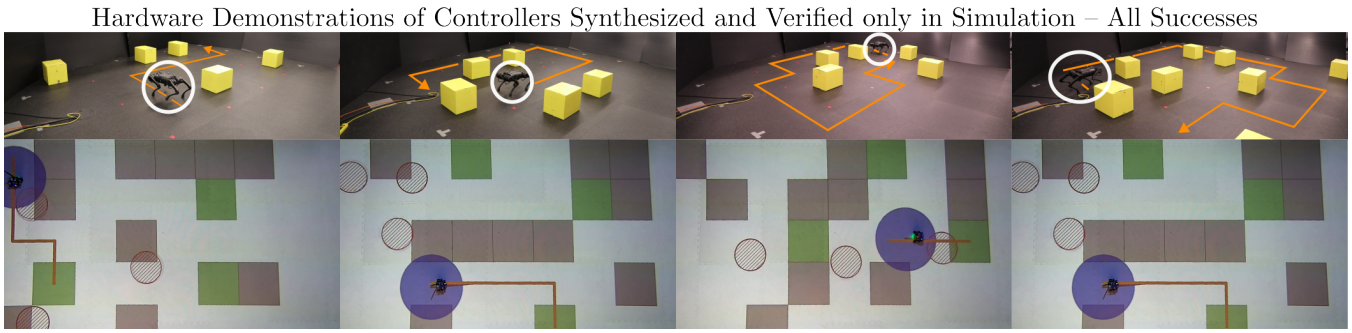

Fig. 4. Since we verified our controllers against the uncertain model produced by our procedure, we expect that the closed-loop hardware systems should realize similar, satisfactory behavior. Indeed, for the first 10 runs on the quadruped and the first 40 runs on the Robotarium, the agents were able to avoid static/moving obstacles and navigate to their goals successfully, despite a wide variety of randomized test scenarios. The first four runs for both systems are depicted above. This ability to synthesize and verify controllers in simulation, with confidence that similar behaviors will manifest in the true system without requiring additional testing, is the main benefit of our proposed approach. Paths for all tests are shown in orange, and the quadruped is highlighted in white. The multi-level control architecture is depicted in Figure 2.

Proof: This is an application of Corollary 2 in [29]. ■

This completes the theoretical statement of our proposed pipeline for safety-critical controller verification via sim2real gap quantification. For the intermediate synthesis step, one can really use any method they like *e.g.* robust control methods [36], [37], input-to-state-stable Lyapunov or barrier functions [23], [25], [38], [39], *etc*. The emphasis here is on verifying the resulting controller against the uncertain model and using Corollary 2 to discriminate between better controllers - those with a higher minimum probability of realizing safe trajectories - and worse controllers - those with a lower minimum probability. By Theorem 2 and Corollary 2 we know that our uncertain model is a decent representor of system behavior. Therefore, those controllers with a high probability of exhibiting desired safe behavior in the uncertain simulator should likely have a high probability of exhibiting desired safe behavior on hardware. We will demonstrate this procedure on two hardware platforms - the Robotarium [31] and a quadruped.

V. Experimental Demonstrations

We will describe our procedure's implementation on both systems simultaneously, as we aim to represent both systems with the same model abstraction, a unicycle:

$$x_{k+1} = f(x_k, u_k),\ x_{k,k+1} \in \mathcal{X},\ u_k \in \mathcal{U},\ t_{k+1} - t_k = \Delta t$$

$$\hat{x}_{j+1} = \hat{x}_j + \Delta\hat{t} \begin{bmatrix} \cos(\hat{x}[3]) & 0 \\ \sin(\hat{x}[3]) & 0 \\ 0 & 1 \end{bmatrix} \hat{u}_j,\ \hat{x}_{j,j+1} \in \hat{\mathcal{X}},\ \hat{u}_j \in \hat{\mathcal{U}}.$$

The specific parameters for each system are as follows (Robotarium (R) and Quadruped (Q)):

(R) $\hat{\mathcal{X}} = [-1.6 \times 1.6] \times [-1, 1] \times [0, 2\pi]$, $\hat{\mathcal{U}} = [-0.2, 0.2] \times [-\pi, \pi]$, $\Delta t, \Delta\hat{t} = 0.033$.

(Q) $\hat{\mathcal{X}} = [-2.5, 2.5]^2 \times [0, 2\pi]$, $\hat{\mathcal{U}} = [-0.15, 0.15] \times [-0.3, 0.3]$, $\Delta t = 0.001$, and $\Delta\hat{t} = 0.1$.

For both systems, we can read true state data x to recover the idealized unicycle state $\hat{x}$. Therefore, M_x is just a projection for both systems. The Robotarium permits unicycle-like commands to their agents resulting in $M_u = I_{2\times 2}$. On the other hand, the quadruped has a lower-level walking controller that operates at 1 kHz to realize commanded forward walking and yaw angular velocities [40]. As a result, M_u for the quadruped is this pre-built walking controller. Finally, our observation maps:

(R) $O_R(x_0, \hat{u}) = M_x(x_1)$ - read the projected true state after one time-step, and,

(Q) $O_Q(x_0, \hat{u}) = M_x(x_{100})$ - read the projected true state after 100 time-steps.

Sampling from the Comparison Distribution: To calculate the discrepancy between the system and our chosen model, we will sample from the comparison distribution π (Definition 3) with the steps listed below:

(x_0) We uniformly randomly sample a planar position x from $\hat{\mathcal{X}}$. Then, we send both agents to the waypoint x using a Lyapunov controller built on top of the input maps M_u for both systems. Once the system reaches a ball of 0.1 m around the desired waypoint, we stop and record the resulting state as x_0.

($\hat{u}$) We uniformly randomly sample an input $\hat{u}$ from $\hat{\mathcal{U}}$. Then, we command the system with this input for 50 true-system time-steps for the Robotarium and 1000 true-system time-steps for the quadruped. We command this input for an extended period to approximate the randomized initial location that we had before sampling x_0 so that subsequent samples drawn from this procedure are drawn effectively independently.

Probabilistic Sim2Real Gap Calculation: For the Robotarium, we collected 2400 observations and used the first 600 to calculate constraints for (4). This provided a sim2real gap constant $\Lambda_N^* = 0.198$ which, according to Theorem 2, is greater than any sampled sim2real gap with minimum probability 99.5% with minimum confidence 95%. Figure 5 (a) shows the probabilistic sim2real gap Λ_N^* overlaid on a histogram of sampled sim2real gaps to approximate the underlying distribution. Notice that since Λ_N^* is indeed greater than the 99.5% cutoff, this corroborates Theorem 2. Figure 5 (b) shows similar results for the quadruped after taking 100 observations. The true cutoff value is not shown as we did not exhaustively sample observations to approximate the underlying distribution. Although, with Theorem 2 and the Robotarium results, we are confident that the calculated

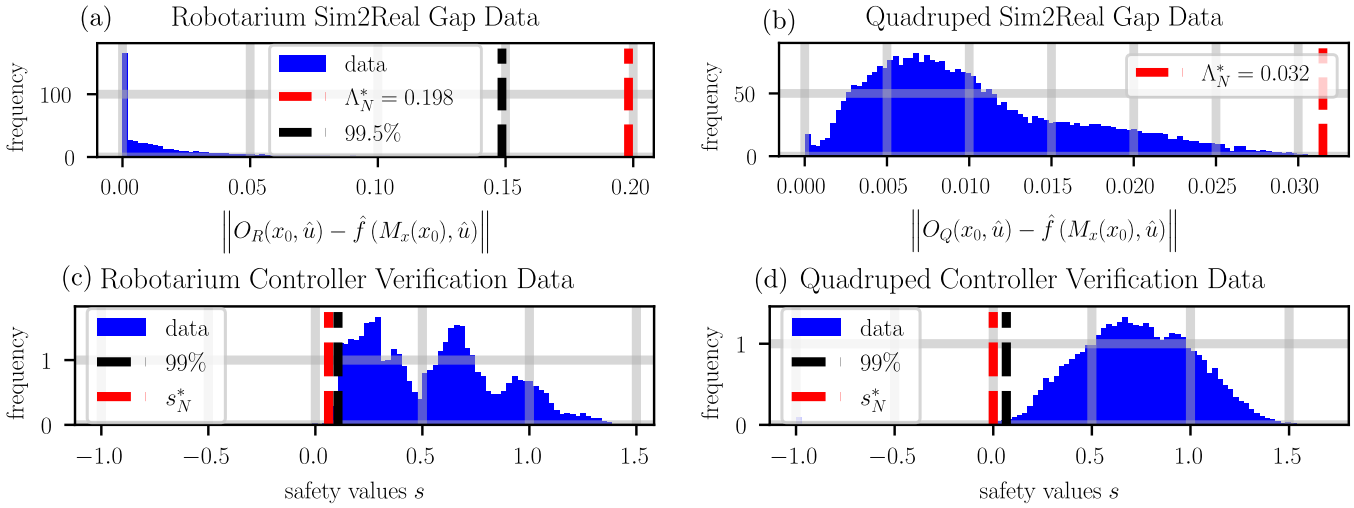

Fig. 5. All data for our experimental pipeline. (Top Left) Data for calculation of a probabilistic sim2real gap Λ_N^* for the Robotarium. Per Theorem 2, we expect that after observing 600 randomly sampled errors, our reported sim2real gap $\Lambda_N^* = 0.198$ is greater than any sample-able gap with minimum probability 99.5% with 95% confidence. Comparing Λ_N^* to the true cutoff after taking 2400 samples verifies this inequality and supports Theorem 2. (Top Right) Data for sim2real gap calculation for the quadruped. True cutoffs are not shown as we did not exhaustively sample gaps. (Bottom) Verification data for both controllers against their respective uncertain models. Note that in both cases, we sampled 300 trajectories to calculate a minimum safety value s_N^* which, according to Corollary 2, should be less than any sample-able safety value with minimum probability 99% with 95% confidence. Taking 20000 safety samples and calculating the true cutoffs against the sampled data shows that this inequality holds verifying Corollary 2.

sim2real gap is greater than any sample-able gap with minimum probability 97% and with confidence 95%.

Safety-Critical Controller Verification: Figure 2 shows the general controller architecture for which we aim to identify parameters such that the resulting controller $\hat{U}$ has a high probability of rendering satisfactory behavior on the uncertain model. To synthesize such a controller, we use a safety metric h as per (9) that outputs -1 if the agent crashes into either a static or moving obstacle and outputs the Manhattan distance traveled along the shortest feasible path to a goal - the orange line in Figure 2 - if it successfully avoids crashes within 200 time-steps. We do not formally define this metric as it is not central to the paper's concept.

For both systems, we iterated through at least 10 different sets of controllers $\hat{\mathrm{U}}$ with parameter spaces, $\Theta =$

(R) All possible setups of 10 static obstacles, 3 goals, and 1 initial condition cell in an 8×5 grid such that a feasible path exists between the initial cell and at least one goal. This is in addition to the starting locations and movement directions of 3, uncontrolled moving obstacles.

(Q) All possible setups of 5 static obstacles, 1 goal, and 1 initial condition cell in a 5×5 grid such that a feasible path exists between the initial cell and the goal.

Once we found controllers that, according to Corollary 2, had a minimum satisfaction probability of at least 99% with 95% confidence, we implemented these controllers $\hat{\mathrm{U}}_R, \hat{\mathrm{U}}_Q$ on their respective systems, the Robotarium and the quadruped. In each verification step, we evaluated the controllers under 500 randomized scenarios - 500 test scenarios we would have otherwise had to run on real systems were we not using our uncertain model to approximate true-system behavior. Figures 5 (c) and (d) show the verification data for the finalized controllers $\hat{\mathrm{U}}_R, \hat{\mathrm{U}}_Q$, respectively, with the distribution approximated by evaluating 20000 randomized scenarios.

According to our pipeline then, for the fact that we verified our controllers - Corollary 2 - against an uncertain model which has a high probability of representing true system behavior - Theorem 2 and Corollary 1 - these controllers should similarly exhibit satisfactory behavior on their true systems when implemented. Figure 4 depicts the first four tests underwent by both systems - they successfully completed their tasks in these tests as expected. We ran 40 more randomized tests for the Robotarium, drawing from the same parameter space Θ as described earlier - all successes. Similarly, we ran 10 more tests for the Quadruped drawing from its respective parameter space Θ - all successes. We expected this level of performance since the controllers exhibited a high probability of realizing desired safe behaviors on their respective uncertain models that encapsulated true system behavior. Furthermore, we did not have to run any tests on hardware to gain this level of confidence in our controller, except for those tests required to calculate the sim2real gap.

VI. Conclusion

We present a pipeline for safety-critical controller verification via sim2real gap quantification. Our pipeline starts by augmenting the nominal model with an uncertainty set generated via a probabilistic sim2real gap analysis between the model and the system it represents. By using this uncertain model for synthesis and verification, we limit the number of hardware tests we have to run to develop effective, safe controllers that exhibit satisfactory performance on hardware. We showcase our procedure successfully developing satisfactory controllers on two hardware platforms - the Robotarium and a quadruped. In future work, we hope to add a randomized synthesis step to our pipeline, to automate the generation of safe, effective controllers while minimizing the need for extensive/expensive hardware tests throughout.

REFERENCES

[1] A. K. Tangirala, *Principles of system identification: theory and practice*. Crc Press, 2018.

[2] E. A. Morelli and V. Klein, *Aircraft system identification: theory and practice*. Sunflyte Enterprises Williamsburg, VA, 2016, vol. 2.

[3] L. Ljung, "System identification," in *Signal analysis and prediction*. Springer, 1998, pp. 163–173.

[4] K. J. Keesman and K. J. Keesman, *System identification: an introduction*. Springer, 2011, vol. 2.

[5] M. Schetzen, *The Volterra and Wiener Theories of Nonlinear Systems*. Krieger Pub., 2006. [Online]. Available: https://books.google.com/books?id=hgFVAAAAYAAJ

[6] T. Koh and E. Powers, "Second-order volterra filtering and its application to nonlinear system identification," *IEEE Transactions on acoustics, speech, and signal processing*, vol. 33, no. 6, pp. 1445–1455, 1985.

[7] S. A. Billings, *Nonlinear system identification: NARMAX methods in the time, frequency, and spatio-temporal domains*. John Wiley & Sons, 2013.

[8] K. Rodriguez-Vazquez and P. J. Fleming, "A genetic programming/narmax approach to nonlinear system identification," in *Second International Conference on Genetic Algorithms in Engineering Systems: Innovations and Applications*. IET, 1997, pp. 409–414.

[9] S. Chen, X. Wang, and C. J. Harris, "Narx-based nonlinear system identification using orthogonal least squares basis hunting," *IEEE Transactions on Control Systems Technology*, vol. 16, no. 1, pp. 78–84, 2007.

[10] O. Nelles, *Nonlinear system identification: from classical approaches to neural networks, fuzzy models, and gaussian processes*. Springer Nature, 2020.

[11] S. Chen, S. A. Billings, and P. Grant, "Non-linear system identification using neural networks," *International journal of control*, vol. 51, no. 6, pp. 1191–1214, 1990.

[12] S. A. Billings and H.-L. Wei, "A new class of wavelet networks for nonlinear system identification," *IEEE Transactions on neural networks*, vol. 16, no. 4, pp. 862–874, 2005.

[13] J. Kocijan, A. Girard, B. Banko, and R. Murray-Smith, "Dynamic systems identification with gaussian processes," *Mathematical and Computer Modelling of Dynamical Systems*, vol. 11, no. 4, pp. 411–424, 2005.

[14] J. Bernardo, J. Berger, A. Dawid, A. Smith, *et al.*, "Regression and classification using gaussian process priors," *Bayesian statistics*, vol. 6, p. 475, 1998.

[15] I. D. J. Rodriguez, U. Rosolia, A. D. Ames, and Y. Yue, "Learning to control an unstable system with one minute of data: Leveraging gaussian process differentiation in predictive control," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2021, pp. 3896–3903.

[16] M. Abdalmoaty, "Identification of stochastic nonlinear dynamical models using estimating functions," Ph.D. dissertation, KTH Royal Institute of Technology, 2019.

[17] F. Giri and E.-W. Bai, *Block-oriented nonlinear system identification*. Springer, 2010, vol. 1.

[18] G. Garofalo, C. Ott, and A. Albu-Schäffer, "Walking control of fully actuated robots based on the bipedal slip model," in *2012 IEEE International Conference on Robotics and Automation*. IEEE, 2012, pp. 1456–1463.

[19] M. Thieffry, A. Kruszewski, C. Duriez, and T.-M. Guerra, "Control design for soft robots based on reduced-order model," *IEEE Robotics and Automation Letters*, vol. 4, no. 1, pp. 25–32, 2018.

[20] X. Xiong and A. D. Ames, "Bipedal hopping: Reduced-order model embedding via optimization-based control," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2018, pp. 3821–3828.

[21] S. L. Herbert, M. Chen, S. Han, S. Bansal, J. F. Fisac, and C. J. Tomlin, "Fastrack: A modular framework for fast and guaranteed safe motion planning," in *2017 IEEE 56th Annual Conference on Decision and Control (CDC)*. IEEE, 2017, pp. 1517–1522.

[22] C.-Y. Su, Y. Stepanenko, and A. A. Goldenberg, "Reduced order model and robust control architecture for mechanical systems with nonholonomic pfaffian constraints," *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans*, vol. 29, no. 3, pp. 307–313, 1999.

[23] S. Kolathaya and A. D. Ames, "Input-to-state safety with control barrier functions," *IEEE control systems letters*, vol. 3, no. 1, pp. 108–113, 2018.

[24] A. Taylor, A. Singletary, Y. Yue, and A. Ames, "Learning for safety-critical control with control barrier functions," in *Learning for Dynamics and Control*. PMLR, 2020, pp. 708–717.

[25] X. Xu, P. Tabuada, J. W. Grizzle, and A. D. Ames, "Robustness of control barrier functions for safety critical control," *IFAC-PapersOnLine*, vol. 48, no. 27, pp. 54–61, 2015.

[26] A. Mesbah, "Stochastic model predictive control with active uncertainty learning: A survey on dual control," *Annual Reviews in Control*, vol. 45, pp. 107–117, 2018.

[27] R. Soloperto, M. A. Müller, S. Trimpe, and F. Allgöwer, "Learning-based robust model predictive control with state-dependent uncertainty," *IFAC-PapersOnLine*, vol. 51, no. 20, pp. 442–447, 2018.

[28] J. Schneider, "Exploiting model uncertainty estimates for safe dynamic control learning," *Advances in neural information processing systems*, vol. 9, 1996.

[29] P. Akella, M. Ahmadi, and A. D. Ames, "A scenario approach to risk-aware safety-critical system verification," *arXiv preprint arXiv:2203.02595*, 2022.

[30] P. Akella, A. Dixit, M. Ahmadi, J. W. Burdick, and A. D. Ames, "Sample-based bounds for coherent risk measures: Applications to policy synthesis and verification," *arXiv preprint arXiv:2204.09833*, 2022.

[31] S. Wilson, P. Glotfelter, L. Wang, S. Mayya, G. Notomista, M. Mote, and M. Egerstedt, "The robotarium: Globally impactful opportunities, challenges, and lessons learned in remote-access, distributed control of multirobot systems," *IEEE Control Systems Magazine*, vol. 40, no. 1, pp. 26–44, 2020.

[32] M. C. Campi and S. Garatti, "The exact feasibility of randomized solutions of uncertain convex programs," *SIAM Journal on Optimization*, vol. 19, no. 3, pp. 1211–1230, 2008.

[33] ——, "Wait-and-judge scenario optimization," *Mathematical Programming*, vol. 167, no. 1, pp. 155–189, 2018.

[34] A. Donzé and O. Maler, "Robust satisfaction of temporal logic over real-valued signals," in *International Conference on Formal Modeling and Analysis of Timed Systems*. Springer, 2010, pp. 92–106.

[35] A. D. Ames, X. Xu, J. W. Grizzle, and P. Tabuada, "Control barrier function based quadratic programs for safety critical systems," *IEEE Transactions on Automatic Control*, vol. 62, no. 8, pp. 3861–3876, 2016.

[36] K. Zhou and J. C. Doyle, *Essentials of robust control*. Prentice hall Upper Saddle River, NJ, 1998, vol. 104.

[37] M. Green and D. J. Limebeer, *Linear robust control*. Courier Corporation, 2012.

[38] E. D. Sontag and Y. Wang, "On characterizations of the input-to-state stability property," *Systems & Control Letters*, vol. 24, no. 5, pp. 351–359, 1995.

[39] J. P. Hespanha, D. Liberzon, and A. R. Teel, "Lyapunov conditions for input-to-state stability of impulsive systems," *Automatica*, vol. 44, no. 11, pp. 2735–2744, 2008.

[40] W. Ubellacker and A. D. Ames, "Robust locomotion on legged robots through planning on motion primitive graphs," in *2023 IEEE International Conference on Robotics and Automation (ICRA), submitted*, 2023.