Closing the Sim-to-Real Gap: Falsification-Driven Testing on a Pololu 3pi+ Robot

Marcus Koh, Jinhong Zhao, Hagen Haeussler
marcuskoh@berkeley.edu, asg2rg@berkeley.edu, hagen.haeussler@berkeley.edu

ABSTRACT

We investigate the Sim-to-Real gap through falsification-driven testing on a Pololu 3pi+ 2040 differential-drive robot executing multi-waypoint navigation in matched Webots simulation and hardware. Systematically varying terrain and velocity parameters, we quantify trajectory divergence and demonstrate condition-dependent gaps. Our results show that falsification-based exploration effectively identifies operational regimes where physics modeling fails, providing targeted guidance for simulator refinement. This work demonstrates falsification as a practical and generalizable framework for diagnosing Sim-to-Real failures in mobile robotic systems.

1 Introduction

Deploying autonomous systems in simulation has become increasingly common, offering a safe, rapid, and cost-effective method to test complex robotics policies before real-world deployment. However, *Sim-to-Real* transfer faces a persistent challenge: systematic behavioral differences between simulated and physical systems caused by imperfect physics modeling, sensor noise, and unmodeled dynamics.

Our work reframes the Sim-to-Real gap as a structured failure discovery problem. We adopt a falsification-driven approach to expose these failure modes by systematically searching parameter space for conditions for worst-case divergence, revealing failure modes that would be difficult to observe through nominal testing.

Our work integrates key cyber-physical systems concepts: matched simulation environments, feedback control via dual-loop PID controllers, wireless networking with bluetooth communication, and synchronous, parallel machines with precise motion capture sensing using overhead cameras and robot control. This methodology provides actionable insight into which parameters most strongly degrade simulation accuracy, enabling targeted model refinement rather than broad parameter randomization.

2 Related Works

Past attempts to close the Sim-to-Real gap rely on domain randomization [1], [2]. This process randomizes the simulation parameters across many trials, attempting to verify or improve the robustness of a robot task in many potential real-world scenarios. Methods such as adaptive domain randomization further incorporate real-world statistics to iteratively update parameter distributions, demonstrating improved transfer performance in certain tasks [1].

While effective for increasing robustness, domain randomization methods primarily optimize for average-case performance and typically require manual selection of parameters and ranges to randomize. As a result, these approaches often obscure what is actually responsible for Sim-to-Real failure, leaving open the question of which modeling assumptions most critically limit simulation fidelity.

Falsification-based methods offer an alternative perspective by explicitly searching for worst-case scenarios that violate system-level specifications. Recent work has applied falsification techniques to autonomous driving simulations, where metrics are modified to extensively search for failure scenarios in autonomous driving [3]. Our paper draws inspiration from this but applies falsification to the Sim-to-Real transfer problem in isolation. Rather than evaluating policy safety, we use falsification to define parameter sets that maximize the divergence from simulated and physical robot trajectories, enabling targeted analysis of modeling errors.

3 Falsification with Sim-to-Real Gap

For a real-world robot task there are typically constraints on its behavior due to safety or other considerations. Tying this to falsification, this can come in the form of a threshold, where errors below the threshold are considered successes, and errors above the threshold are considered failure. In this paper, we use the **geometric deviation from path** as this constraint, which we quantify with the Sim-to-Real Gap.

3.1 Estimating the Sim-to-Real Gap

To quantify the gap between Simulation and Reality, we choose to compare geometric differences in paths, with the largest difference being the Sim-to-Real Gap. In this paper we represent path data as a collection of time-stamped poses.

As mentioned earlier, our primary definition of the Sim-to-Real Gap is defined as the maximum geometric deviation between the simulated and real trajectories. We selected this since it removes differences in factors such as latency, absolute timestamps, and speed. Instead, we reparameterize by *normalized arc length*, which quantifies the Sim-to-Real Gap as a single scalar value and enables a purely spatial comparison.

Given a planar trajectory

$$\mathbf{p}(t) = (x(t), y(t))$$

its arc length is defined as

$$s(t) = \int_{t_0}^{t} \left\| \frac{d\mathbf{p}(\tau)}{d\tau} \right\|_2 d\tau$$

or in discrete form

$$s_i = \sum_{k=1}^{i} \|\mathbf{p}_k - \mathbf{p}_{k-1}\|_2 = \sum_{k=1}^{i} \sqrt{(x_k - x_{k-1})^2 + (y_k - y_{k-1})^2}$$

then normalizing to standardize comparisons

$$u = \frac{s(t)}{s_{\text{total}}}, \qquad u \in [0, 1]$$

and find the error at normalized arc length u, interpolating if necessary to obtain values between timesteps

$$\|\mathbf{p}_{\text{sim}}(u) - \mathbf{p}_{\text{real}}(u)\|_2 = \sqrt{(x_{\text{sim}}(u) - x_{\text{real}}(u))^2 + (y_{\text{sim}}(u) - y_{\text{real}}(u))^2}$$

Then, we take the max value across N evenly samples (2000 used for the experiments in this paper) to get the Sim-to-Real Gap.

3.2 Falsification

We then evaluate how this gap varies across changes in environmental and control parameters through the lens of falsification. We formalize this definition below

Let $\theta \in \Theta$ denote a d-dimensional vector of parameters, where d denotes the number of different parameters being modified. For each parameter setting θ, we compute the Sim-to-Real Gap (Δ(θ)):

$$\Delta(\theta) = \max_{u \in [0,1]} \|\mathbf{p}_{\text{sim}}(u; \theta) - \mathbf{p}_{\text{real}}(u)\|_2$$

Large values of Δ(θ) falsify the task under the parameter θ.

This falsification method provides several advantages:

1. By using the *maximum*, we emphasize failure modes in the worst-case scenario, which is more applicable to real-world robotics tasks for safety and robustness
2. Permuting over the set Θ gives a structured results in d-dimensions, which allows for efficient comparison of both modifying one parameter at a time and modifying multiple parameters together.
3. Falsification highlights simulation parameters that are both good and bad, informing future decisions. For example, if a parameter θ_i always results in successes, it is generally safe to deploy into reality for these conditions, while for a parameter θ_j that contains failures, this points to further testing and verification before deployment.

4 System Architecture

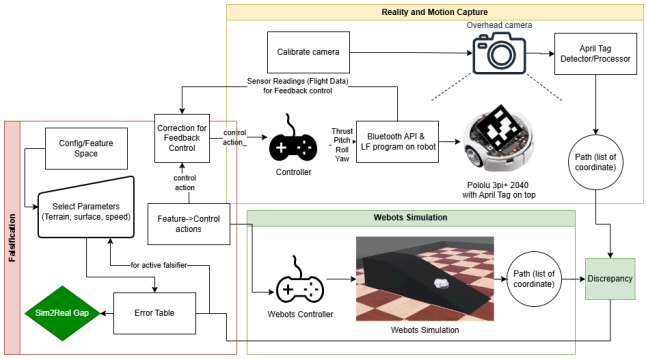

Figure 1: **Flowchart diagram describing our system**

4.1 Hardware Control Stack

Figure 2: **The Pololu 3pi+ 2040 robot**

We use a Pololu 3pi+ 2040 robot, a small, simple differential driving robot. To communicate with the robot, we connected a UART bluetooth receiver to the robot, and ran

a Lingua Franca coded listener which waits for motor commands (power from 0 to 1) for its two wheels.

The control system operates on a distributed architecture with a Python script running on a host computer managing both sensing and control. An overhead camera thread provides robot localization, which feeds into a dual-loop PID controller that computes translational and rotational velocity commands for waypoint navigation. The controller implements separate PID loops for distance and heading errors, with tunable gains enabling both precise stopping at waypoints and smooth through-waypoint trajectories with measured Bluetooth latency incorporated into the control loop, while differential drive commands are rate-limited and normalized to ensure physical motor constraints are respected.

To input waypoints, we defined a yaml configuration containing a list of waypoints. For each waypoint, we define a x,y (2D) position and a “stop” boolean, which defines whether a robot should go through the waypoint without stopping, or stop, then move to the next waypoint. For the first waypoint (the start waypoint), we also defined a starting heading of the robot.

4.2 Motion Capture

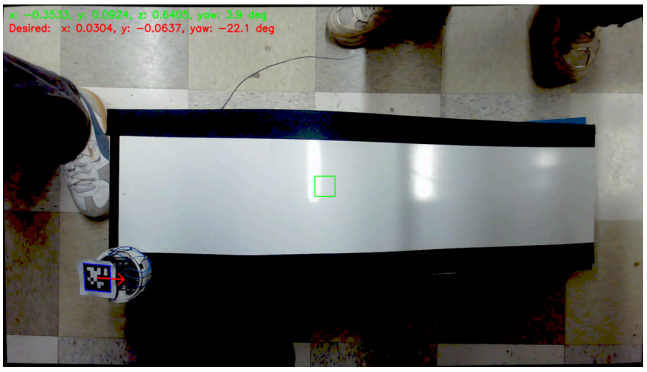

Figure 3: **GUI of the camera feed and AprilTag detection**

Our motion capture system uses an overhead Victure SC30 wide-angle camera (1080p, 30 fps) with OpenCV-based AprilTag detection to track robot pose. We choose the tag36h11 family with id #0 as our AprilTag. This family was chosen due to its robustness and balance at high resolutions. The camera is calibrated using a chessboard pattern with known geometric properties to correct for lens distortion and establish accurate spatial measurements. A Python thread processes the video stream in real-time, detecting AprilTags mounted on the robot and logging 6-DOF pose data (position and orientation) with timestamps at up to ~25 Hz. Furthermore, we update the current state of the robot, providing continuous feedback for the waypoint navigation controller.

4.3 Simulation Environment

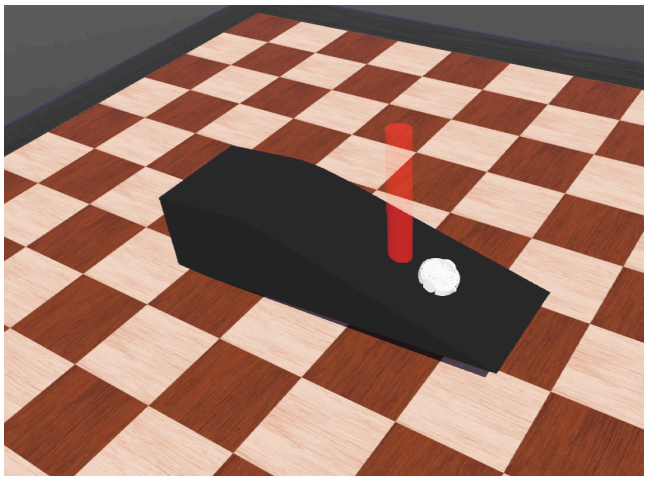

Figure 4: **The Webots simulation environment**

We implemented a matched Webots simulation environment mirroring our physical waypoint-navigation platform. The robot and environment were reconstructed from CAD/Blender assets to minimize modeling mismatch, with identical waypoint navigation and controller logic ensuring differences stem from real-world physics rather than control variations.

To replicate real-world sensing, we simulate motion capture by logging the robot's global pose directly from the simulator at a fixed rate, matching the state feedback from our overhead camera and AprilTag pipeline. Bluetooth communication delay is modeled by applying control commands with a fixed time offset, approximating the latency in real motor velocity command transmission. The simulation supports controlled variation of three falsification parameters—terrain layout, surface friction, and robot speed—enabling systematic search for conditions that maximize Sim-to-Real divergence.

4.4 Supplementary Encoder-Based Analysis

As a complementary investigation, we developed an encoder-based approach to quantify systematic deviations between idealized motion models and real-world behavior under controlled experimental conditions. The Pololu's onboard incremental quadrature encodes logged motor displacement during prescribed trajectories. The same

motion capture system (4.2) provided ground-truth pose for comparison against encoder-derived odometry. Experiments systematically varied commanded speed and surface friction. For each parameter combination, multiple trajectory repetitions characterized inter-trial variance and deviation from ideal programmed paths, quantifying slip and model mismatch as functions of operating conditions. Unlike the simulation-based falsification framework, this encoder analysis was conducted exclusively in hardware to provide empirical measurements of physical mechanisms contributing to Sim-to-Real divergence.

5 Experiments

5.1 Camera-Based Waypoint Navigation

We evaluate sim to real using a multi-waypoint navigation task executed in both simulation and the real world. In real experiments, we first record the waypoints and put them in a yaml file. Then, the robot's trajectory is recorded using an overhead camera with AprilTag tracking to obtain global position, which will then be saved as a csv file for later analysis. Motor velocities are transmitted over Bluetooth and executed onboard. The simulated experiments follow the same controller structure and log comparable pose trajectories for direct comparison. Our experimental methodology follows a falsification-driven approach. We treat three variables as "knobs" for falsification. They are terrain layout, surface friction, and robot speed. We select a combination of these parameters and run the waypoint task in simulation once and in the real world for 3 trials, then compute the resulting sim to real gap. We permute over parameter combinations to identify cases that produce the largest divergence between simulated and real trajectories. We test multiple scenario families, including flat ground, cardboard group, parchment paper ground, uphill on a ramp, downhill on a ramp, and bump configurations. For each scenario we evaluate on multiple commanded speeds, such as with minimum speed of 0.15 or 0.2 and maximum speed of 0.3, 0.4, and 0.5. Each configuration is repeated 3 times to reduce run-to-run variance from sensing noise and actuation variability. To quantify the Sim-to-Real gap, we compute the Euclidean distance between the simulated and real trajectories after aligning them using normalized arc length, which makes the comparison independent of timing differences due to slip or latency. We report the gap over the trajectory for each configuration, and analyze which falsification parameters most strongly increase divergence.

Figure 5: **Waypoint Initialization**

5.2 Encoder-Based Trajectory Experiments

We conducted controlled encoder odometry experiments to quantify trajectory deviation and inter-trial variance as functions of surface properties and commanded velocity. Three surface conditions were tested: square-textured bump surface, dry parchment paper, and wet parchment paper. For each surface, the Pololu executed prescribed trajectories at 25%*, 50%, and 75% of maximum velocity, with five repetitions per configuration to ensure statistical reliability. Initial conditions were precisely controlled: robot orientation was calibrated to ° and position to millimeter precision using the motion capture system and marks on the surfaces. During execution, both encoder displacement and ground-truth pose (position and orientation with timestamps) were logged synchronously. For visualization purposes a GUI enabled the experimenter to see each AprilTag detection while running each experiment trial. *The 25% speed condition provided insufficient torque to traverse the bump surface and was omitted from that configuration.

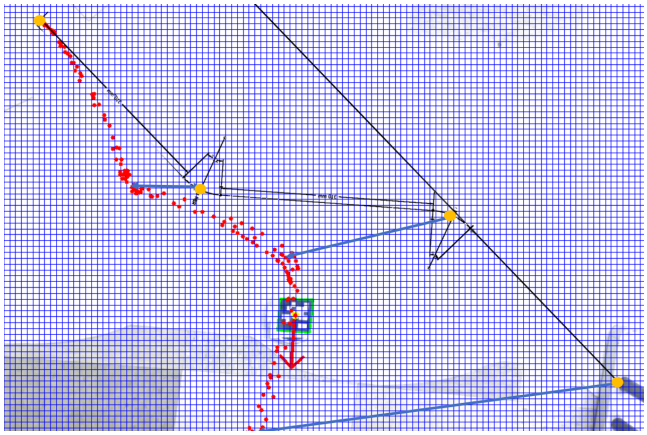

Figure 6: **Encoder-based trajectory tracking GUI displaying real-time AprilTag pose estimation for the Pololu robot. The current robot pose is indicated by the green bounding box and red orientation arrow. Red dots denote historical detection points along the executed trajectory. The black line represents the programmed reference trajectory with**

yellow waypoints marking target positions. Blue perpendicular lines illustrate instantaneous lateral deviation from the reference path, showing increasing trajectory error over time.

6 Results

6.1 Sim-to-Real Gap Analysis

The Sim-to-Real gap proved highly condition-dependent across experimental configurations. From Figure 7, we can tell that the Cardboard surfaces yielded the smallest average discrepancy (0.046 m), while standard ground and downhill ramp conditions exhibited significantly larger gaps (0.062 m and 0.060 m respectively), identifying contact dynamics and slope effects as primary contributors to divergence. Low-friction parchment paper surfaces, positioned farther from the camera, demonstrated elevated inter-trial variability, indicating increased sensitivity to slip, localization uncertainty, and tracking noise.

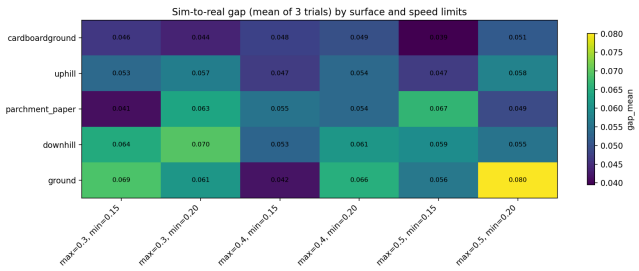

Figure 7: **Sim-to-Real gap by surface and speed limits**

Most notably, higher commanded velocities amplified Sim-to-Real divergence. The smallest gap occurred at low speed on parchment paper, while the largest emerged at high speed on standard ground, suggesting that velocity-dependent dynamic effects and unmodeled contact phenomena are critical sources of simulation error.

In addition to the average trends reported above, Figure 8 breaks down the Sim-to-Real gap per trial for every [v_min, v_max] and surface/terrain setting. The relative ordering of conditions is largely consistent across trials, which means the same settings repeatedly appear with a low gap. This suggests the dominant gap drivers are systematic rather than one-off noise. Within each surface family, increases in v_max tend to shift the gap upward more reliably than changes in v_min. This indicates that peak-speed dynamics are a primary source of divergence.

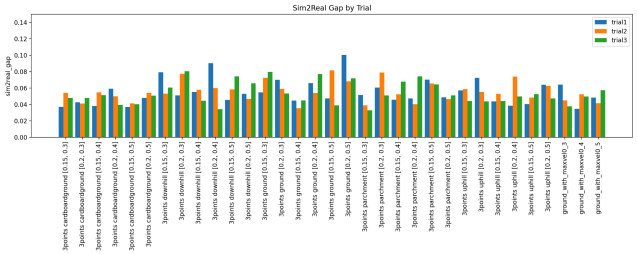

Figure 8: **Overall Sim-to-Real gap by trials**

Figure 9 shows how the Sim-to-Real position error evolves along the trajectory as a function of normalized arc length u. The error is lowest near the start around 0.026m, then gradually increases and reaches its maximum near u=0.73, which is also being marked by the dashed line, with the Sim-to-Real gap around 0.0499m. This non-monotonic profile suggests the gap is localized to a specific portion of the path, rather than uniformly accumulating.

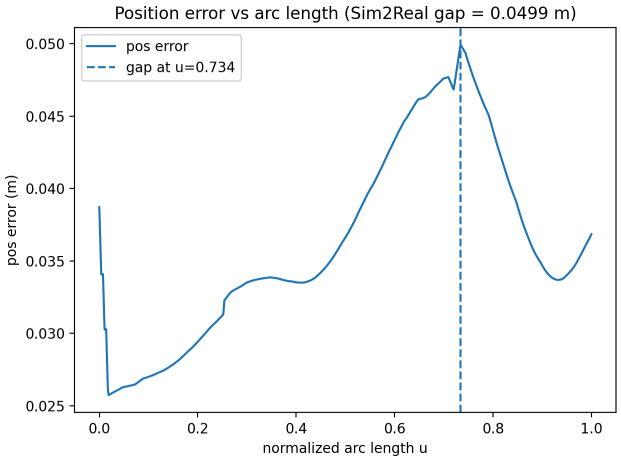

Figure 9: **Position error plotted over normalized arc length of robot trajectory of one trial**

Figure 10 shows the mean Sim-toReal gap as a function of both surface and [v_min, v_max]. The largest mean gaps cluster in high-speed settings. Most notably ground at [0.20,0.50], while cardboard remains relatively low across the velocity sweep. This highlights that the gap is not explained by surface or speed alone, but by their interaction.

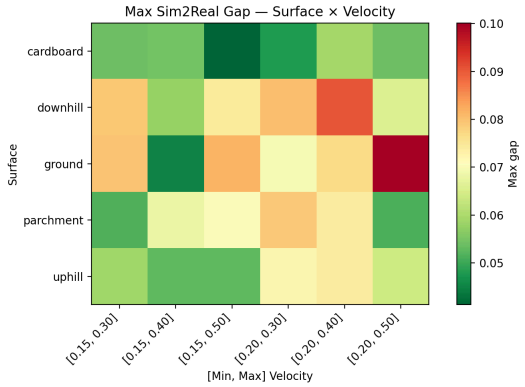

Figure 10: **Mean Sim-to-Real Gap: Surface × Velocity**

Figure 11 summarizes the average Sim-to-Real gap aggregated across velocity settings and trials for each surface and terrain. The gap is lowest on cardboard, increases through uphill and parchment, and is highest on downhill and ground, indicating that contact conditions and slope significantly affect simulation fidelity. This plot provides a ranking of which environments are most and least consistent between sim and real.

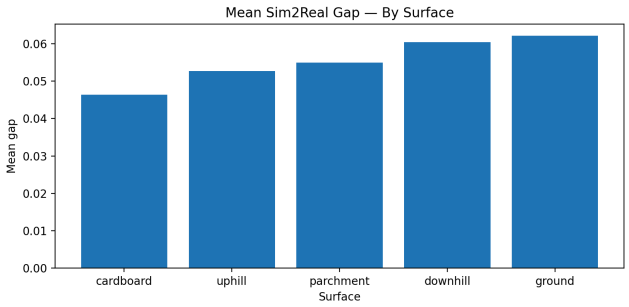

Figure 11: **Mean Sim-toReal Gap by surface**

Figure 12 reports the number of falsification failures, especially for trials with a Sim-to-Real gap >0.06. It aggregated over all velocity settings for each surface. Failures are concentrated on ground (9), parchment (7), and downhill (7), with fewer on uphill (4) and none on cardboard (0). This shifts the result from “average gap differences” to a clearer statement of which environments most often violate the falsification threshold.

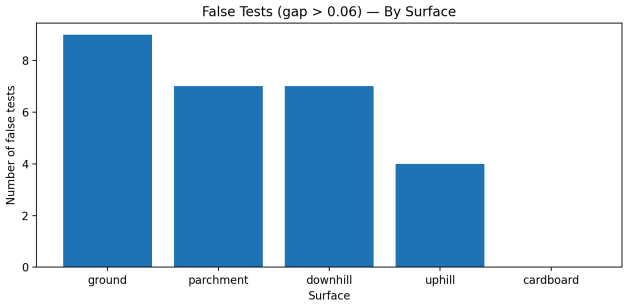

Figure 12: **False Tests (gap > 0.06) by Surface**

Figure 13 shows falsification failures by commanded velocity range [v_min, v_max]. The lowest failure count occurs at [0.15, 0.40] for once, while higher-speed ranges produce substantially more failures, including [0.20, 0.30] and [0.20,0.50] with 6. This supports the falsification framing by showing that increasing the velocity range raises the likelihood of exceeding the 0.06 gap threshold, even after averaging over surface types.

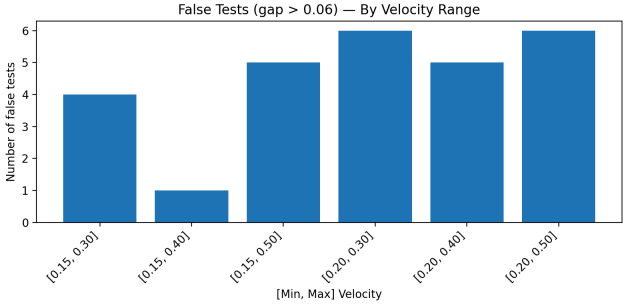

Figure 13: **False Tests (gap > 0.06) by Velocity Range**

6.2. **Secondary Encoder Findings**

Encoder-based trajectory analysis strikingly revealed that average Euclidean deviation from ideal trajectories decreased with increasing speed, indicating encoder drift at low speeds dominates over slip-induced errors at higher speeds. Inter-trial variance exhibited surface-dependent trends. Dry parchment paper produced consistently low variance across all speeds. Wet parchment paper and bump surfaces showed speed-dependent behavior: low variance at lower speeds escalating dramatically at higher speeds as slip and dynamic effects became pronounced. The bumped surface showed a dramatically increasing inter-trail variance

at high speeds while proving consistent at low robot velocities. These findings characterize the same physical mechanisms—encoder drift, surface slip, and contact dynamics—that contribute to model-reality discrepancies in the simulation-based falsification framework, isolating how operating conditions amplify deviations from idealized kinematic predictions.

$\sigma_{RSS} = \sqrt{\sigma_x^2 + \sigma_y^2}$

% of max speed	Material: Parchment Paper	Wet Parchment Paper	Bumps
25	0.035812937	0.022789044	
50	0.024669982	0.06867581	0.007301656
75	0.03029636	0.056048776	0.043678659

$d = \sqrt{(\Delta x)^2 + (\Delta y)^2}$ cm

% of max speed	Material: Parchment Paper	Wet Parchment Paper	Bumps
25	3.831424651	3.719147297	
50	3.648021172	3.545403885	3.726976902
75	3.552857957	3.474147119	3.539338196

Figure 14: **Quantitative analysis of encoder-based trajectory tracking accuracy across surface conditions and velocities. Upper table: RSS standard deviation of inter-trial trajectory variance. Lower table: mean Euclidean distance from programmed path. Each cell represents the average of five trials at the specified velocity (percentage of maximum speed) and surface type.**

7 Conclusion

This work presents a falsification-driven framework for analyzing the Sim-to-Real gap in mobile robotic systems. By reframing Sim-to-Real transfer as a structured failure discovery problem, we demonstrate how systematic exploration of parameter space can reveal condition-dependent discrepancies between simulated and physical behavior. Our approach emphasizes worst-case divergence, enabling interpretable diagnosis of where simulation assumptions break down and offering applicability to critical systems.

We demonstrate this by running experiments in matched simulation and hardware environments using closed-loop feedback control and high-precision motion capture, finding that ground terrain and higher velocities cause the most failures.

Looking ahead, this work suggests a broader role for falsification in robotics and cyber-physical systems. If extended to more complex platforms, such as legged robots, aerial vehicles, or systems with learned policies, falsification-driven Sim-to-Real analysis could serve as a general tool for verifying autonomy under realistic operating conditions. We suggest extensions into automated model adaptation or simulator-in-the-loop learning, as our framework could enable simulators that improve in response to real-world failures. Overall, by prioritizing the discovery of failure regimes, future robotic systems could be developed with stronger guarantees about where and why simulation predictions hold, opening doors to informed, safe simulation tools.

REFERENCES

[1] Yevgen Chebotar, undefined., et al. "Closing the Sim-to-Real Loop: Adapting Simulation Randomization with Real World Experience," in CoRR, vol. abs/1810.05687, 2018.

[2] Mehta, B., et al, "Active Domain Randomization," in Proceedings of the Conference on Robot Learning, 2020, pp. 1162–1176.

[3] F. Indaheng, E. Kim, K. Viswanadha, J. Shenoy, J. Kim, D. J. Fremont, and S. A. Seshia, “A scenario-based platform for testing autonomous vehicle behavior prediction models in simulation,” arXiv preprint arXiv:2110.14870, 2021. [Online]. Available: https://arxiv.org/abs/2110.14870