

INTRODUCTION

Deploying a learned vision policy on a physical robot is limited by the sim-to-real gap: models trained using simulation behave differently in reality. This is especially acute for low-cost platforms with monocular RGB cameras and modest compute capabilities, where large real datasets are expensive to collect but a simulator can generate training data quicker and with less cost.

We ask: how much real data is actually required, and can cheap simulated data make up the difference?

We study this question on a fixed task and platform. Holding the backbone and total data scale constant, we vary the composition of the training set — from real-only, through real + simulated hybrids at controlled real fractions, to simulation-only — and measure the deployed robot's behaviour. Where most sim-to-real navigation work reports qualitative success or lap completion, we measure the robot's trajectory against a motion-capture ground truth and report mean absolute error in millimetres. That resolution is what lets us separate regimes a binary success check would call identical.

Contributions

- A controlled sweep of how much real data to add to a fixed simulated set for sim-to-real visual navigation on a physical robot, showing an optimum at ~90% of the available real data, with the abundant simulation acting as a regularizer rather than a substitute.
- A millimetre-level motion-capture evaluation protocol for line-following policies.
- A robustness finding that sim-containing policies tolerate an unseen lighting shift better than real-only, motivating a domain-randomization follow-up.

METHODS

Task, platform, and control loop

The robot follows a taped line around a tabletop track, choosing at each step among three discrete actions (left, forward, right) from a single monocular RGB camera frame. It runs a discrete perception-action loop: it issues a motion command, stops, captures an image, infers the next action, and repeats. Because inference happens while the robot is stationary, per-step latency does not bound control stability.

Deployment hardware is an NVIDIA JetBot built on the Jetson Nano (version B01) module. Four physical tracks (T1, T3, T4 for training; T2 held out for evaluation) are used throughout; the evaluation track is never seen during training.

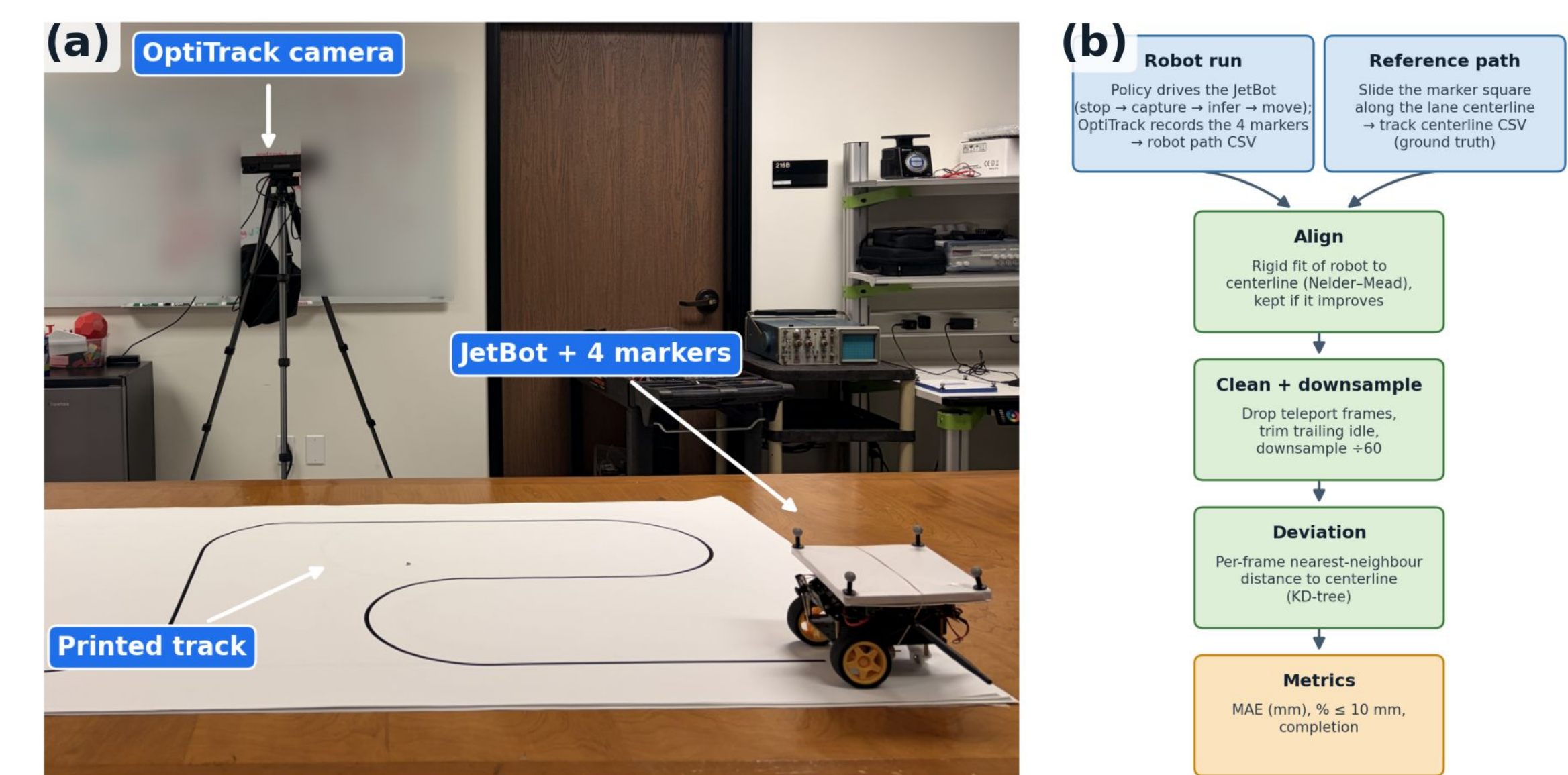


Figure 1: Setup and evaluation pipeline. (a) The JetBot carries a four-marker rigid body and drives the printed track while an OptiTrack camera on a tripod records its trajectory. (b) Each run and a reference centerline are recorded, the robot path is aligned to the centerline, cleaned of motion-capture artifacts and downsampled, then scored per frame as nearest-neighbour distance.

Training data

Simulated: 8,312 images across the four tracks (2,259 / 2,459 / 2,115 / 1,443 on T1–T4), generated in a custom Unreal Engine 5 environment modelled on the physical lab, with the track built from a spline path defined in real-world units. An agent traverses the spline, capturing camera frames and labelling each with the correct next action.

Real: 1,981 labelled frames captured by the JetBot camera at varied positions on the physical tracks, by manually driving the robot and labelling each frame as forward, left, or right. The class distribution is dominated by “forward”, and the per-track counts are imbalanced, a factor that surfaces in the real-only results.

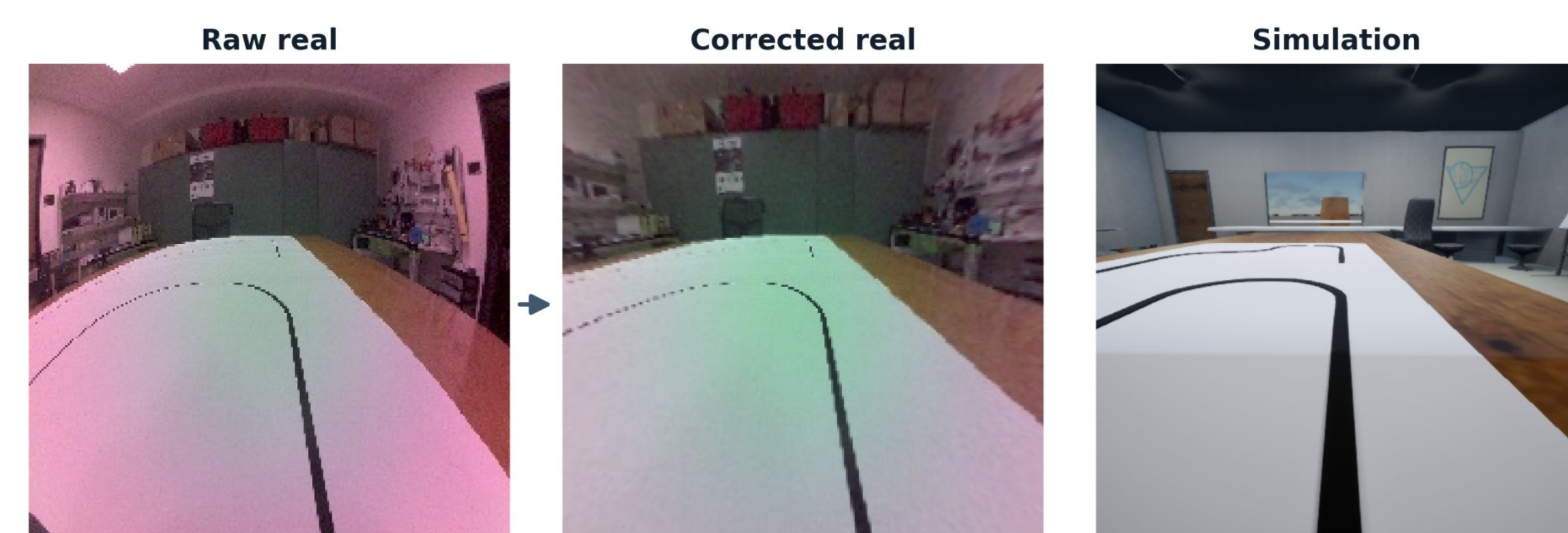


Figure 2: The onboard camera view across domains. Left: a raw real frame — wide-angle fisheye distortion and a strong pink colour cast from the sensor and lab lighting. Middle: the same frame after undistortion and gray-world colour correction, the form the model trains on. Right: a simulated frame (UE5, pinhole camera). A visible appearance gap remains even after correction.

TRAINING

Real frames are preprocessed before training. Each image is undistorted using the camera's calibrated intrinsics to remove the wide-angle lens distortion of the onboard camera, and a gray-world colour correction reduces the strong colour cast introduced by the sensor and lab lighting. Simulated frames, rendered with a pinhole camera in UE5, require no undistortion. Both sources are then resized to 224×224 and normalized with ImageNet statistics.

All models are initialized from ImageNet-pretrained weights and optimized with AdamW (initial learning rate 3×10^{-4} , weight decay 1×10^{-4}) under a cosine-annealing schedule for 25 epochs at batch size 32. Offline augmentation stores task-relevant transformations (brightness, Gaussian noise, Gaussian blur, and horizontal shifts simulating an off-centre robot), doubling the training set; online augmentation applies random resized cropping (scale 0.7–1.0) and colour jitter at load time.

Model selection uses peak validation accuracy. 19 configurations were trained on the Pomona HPC cluster.

The four compared models

M1	Real only
M2	Simulation only
M3	Hybrid (real + full sim)
M14	Hybrid (90% real + full sim)

All ResNet18 unless noted. Every hybrid contains the full 8,312-image simulated set; the percentage refers to how much of the 1,981-frame real pool is added on top.

RESULTS

Data composition: real vs simulated vs hybrid

Simulation-only training fails to transfer: M2 never completes an evaluation run (0% completion) — qualitatively it fails on curved sections, either not initiating a turn or stalling mid-curve. Real-only training transfers but overfits: M1 attains the lowest error of any model on Track 1 (6–8 mm) yet degrades elsewhere and completes only 75% of runs, consistent with the training-set imbalance toward Track 1. The hybrid regime is best: M3 completes every run at ~16 mm across tracks, and M14 (a real-saturated hybrid) matches it at 16.0 mm overall with the highest within-10 mm rate. Only the hybrid models complete every run.

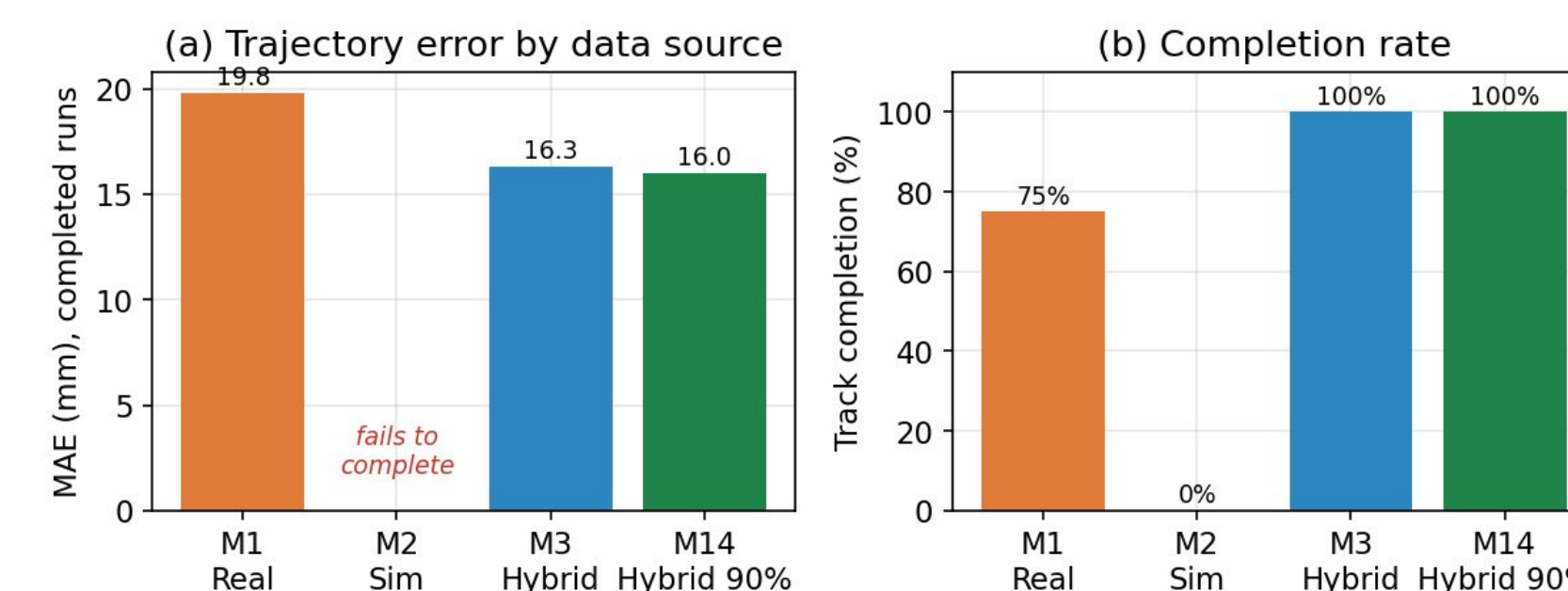


Figure 3: Trajectory error (a) and track completion (b) by training-data source, three-trial re-evaluation. Simulation-only (M2) never completes a run; hybrid (M3, M14) is the most accurate and the only regime to complete every run.

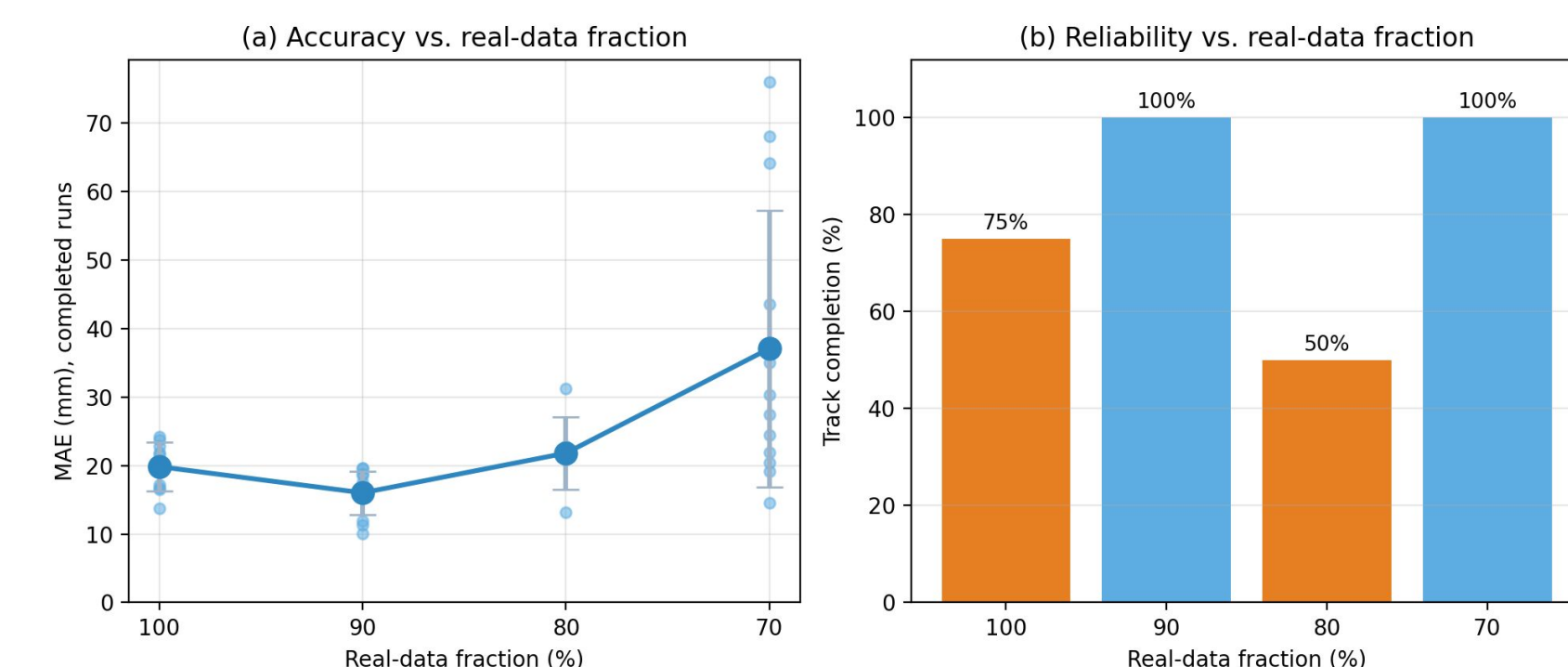


Figure 4: Hybrid models mix $n\%$ of the real dataset with the full simulated set. (a) Trajectory error vs real-data fraction: each light dot is one completed run, the dark marker the mean. (b) Track completion — only the 90% model completes every run.

The real-data ratio matters, and simulation regularizes rather than substitutes

Every hybrid includes the full simulated set; what we vary is how much of the scarce real pool (1,981 frames) we add on top, so “90% real” means 90% of the available real data, not 90% of the training set. Simulation in fact dominates every hybrid by count: even the most real-rich, M14, is only ~18% real by frame. With that axis, performance peaks at 90% of the real pool. M14 is the most accurate and reliable model: 16.0 mm overall MAE at 100% completion across all four tracks, and moving either way is worse. Dropping simulation entirely (the 100% point is real-only M1) overfits and completes only 75% of runs (19.8 mm, failing Track 3); withholding real data degrades sharply and unevenly at 80% (M15) it completes only half its runs, and at 70% (M16) it completes every track but far less accurately (37.1 mm overall). The scarce real data therefore carries transfer and cannot be replaced by abundant simulation, yet simulation is not idle: adding the full sim set to the real-only data fixes that model's overfitting.

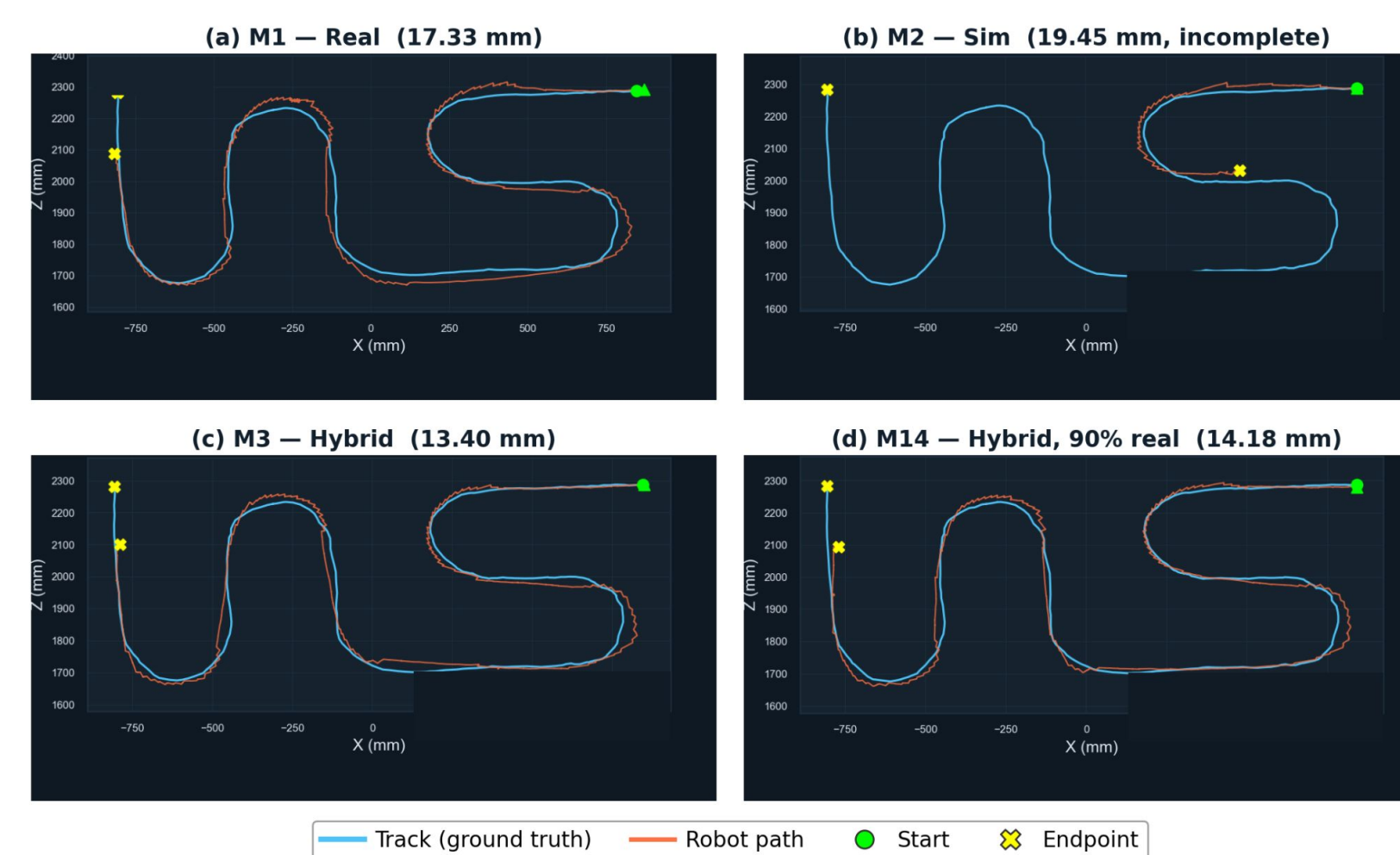


Figure 5: Track-2 path overlays by training-data source (representative run). Blue: ground-truth centerline; orange: driven path; green marker: start; yellow x: run endpoints. (a) M1 (real) follows the track but wanders and overshoots. (b) M2 (sim) fails to complete and leaves the track pathway, so most of the lap is never driven. (c) M3 (hybrid) and (d) M14 (real-saturated hybrid) track the centerline closely.

Robustness to unseen lighting

We re-ran the Track-2 evaluation under dim lighting applied only at test time; no model was trained on dim-lighting data, so this is an out-of-distribution test. The policies that include simulated data are robust: M14 improves from 17.3 to 13.6 mm and M3 is essentially unchanged (15.1 to 15.8 mm), while real-only M1 degrades from 15.8 to 24.0 mm. All three still complete the track.

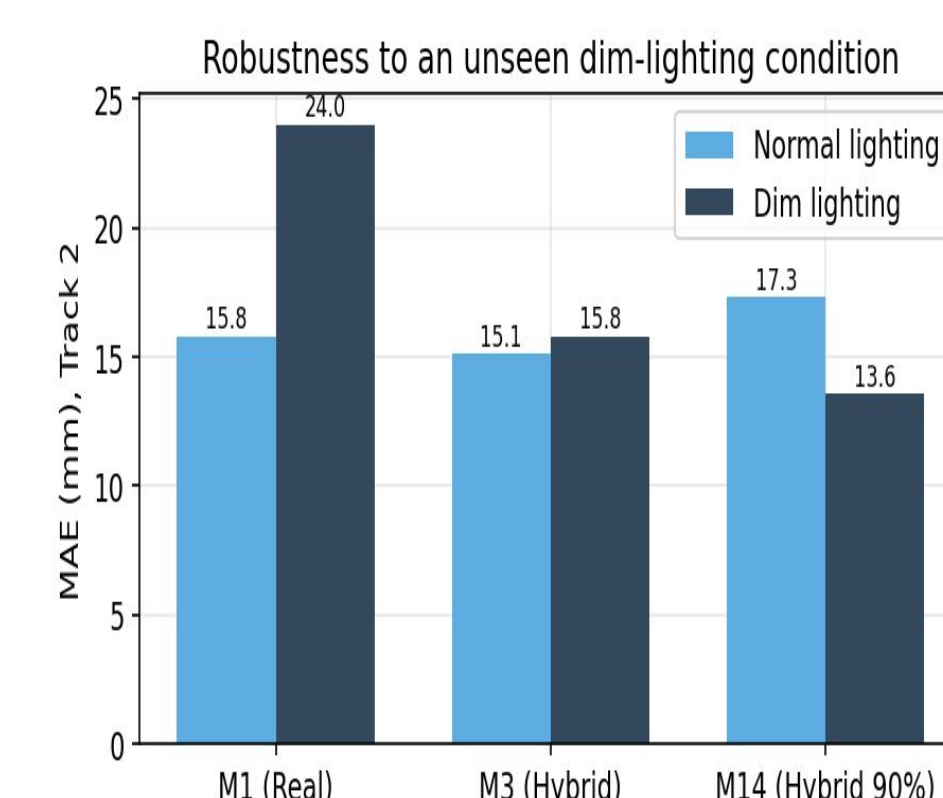


Figure 6: Under an unseen dim-lighting condition on Track 2, the hybrid models hold or improve their accuracy while the real-only model degrades. Three-trial means.

CONCLUSIONS & NEXT STEPS

For sim-to-real visual navigation on a low-cost JetBot, the amount of real data in the training set matters more than any architectural or augmentation trick. A hybrid that keeps the full simulated set and ~90% of the available real data is the only regime that transfers both reliably and accurately; simulation-only fails outright and real-only overfits, and how much of the real pool to include has a clear optimum — with the abundant simulation acting as a regularizer rather than a substitute for the scarce real data. Policies that include simulated data are also more robust to an unseen lighting shift, motivating the domain-randomization study now underway. Throughout, a millimetre-level motion-capture protocol makes these distinctions measurable where task-success metrics cannot.

Next steps

- Domain randomization: isolate whether randomized simulation is the specific mechanism behind the lighting robustness, and whether it buys robustness beyond what plain simulated data already provides.
- Multi-seed statistics: add means, variance, and significance tests to the ratio sweep and to the architecture and augmentation results, which currently rest on single runs.
- Generalization breadth: additional held-out layouts or cross-validation, and more than one lighting condition and surface for the robustness test.

Exploratory, single-run: a ConvNeXt-Tiny backbone generalizes markedly better to the unseen track than ResNet18 (8.39 mm vs 16.57 mm on Track 2, a 49% reduction) but costs ~3.2× more per inference step on the Jetson Nano. TensorRT FP16 optimization trades roughly 3× faster runs for roughly 2× larger trajectory error — the speed-up buys experimental throughput rather than better driving, because per-step latency does not affect control quality under our stop-and-go loop.

Key references: [1] Tobin et al., IROS 2017 · [2] Tremblay et al., CVPR-W 2018 · [3] Bayraktar et al. (ADORESet), MVA 2019 · [4] Suomela et al. (FAINT), 2025 · [5] Kadian et al., IEEE RA-L 2020 · [6] Giusti et al., IEEE RA-L 2016 · [7] Bojarski et al., 2016 · [8] Liu et al. (ConvNeXt), CVPR 2022