Porting NVIDIA Sonic to the AgiBot X2 Ultra: A Sim-to-Sim-to-Real Bridge for a Non-G1 Humanoid

Claude Opus, Sitarama Raju Chekuri, Zeeshaan Mohammed, Dhruv Diddi, Samarth Shukla

ABSTRACT

We deploy a learned whole-body controller on the AgiBot X2 Ultra (31 DoF, 14-DoF dexterous hands) — to our knowledge the first publicly documented Sonic-family policy on a non-G1 humanoid. Reaching the real robot from upstream Sonic required closing two sequential transfers: a *sim-to-sim* transfer from the training simulator (IsaacLab) to the deployment evaluation simulator (MuJoCo), and a *sim-to-real* transfer from MuJoCo to hardware through a closed-API vendor motion controller. Adapting upstream Sonic surfaced gaps at every layer. Motion retargeting introduced arm-flip and wrist-clamp failures rooted in inverse-kinematics assumptions specific to the original platform. The sim-to-sim transfer was blocked by three independent bugs spanning the embodiment, observation, and encoding layers that masqueraded as physics-tuning problems: a **foot-collision URDF mismatch** in the training distribution; a **6D-rotation channel-order error** that slipped past tolerance-based parity checking because aggregate norms are blind to positional differences; and a **tokenizer-layout error** at the runtime-to-export adapter boundary. The sim-to-real handoff produced an audible ***dual-publisher whir*** — our deploy publisher and the vendor motion controller briefly fighting over the joint-command stream because the vendor exposed only coarse start/stop primitives and no graceful-overlap command. We close the gap with a finite-state handoff protocol built on those same primitives, replacing the 1.6 s audible burst with a single continuous handoff. We document each bug's controlled fine-tune validation, the smooth-handoff state machine, and a two-layer validation substrate — sim-side parity gates plus hardware preflight envelopes — that made debugging survivable on a non-trivially-expensive humanoid. A committed sim-to-real *anchor archive* — three matched simulator–hardware recordings from a single canonical checkpoint — validates the URDF/MJCF kinematic chain and torso inertial model to within roughly five degrees per-DoF RMS and base IMU angular velocity to within roughly five percent. We close with a set of policy-free bench tests that would convert this indirect closed-loop evidence into direct, component-level validation, and a four-class failure-mode taxonomy — bug, physics-divergence, handoff, and tuning — intended as an organising principle for subsequent humanoid deployments.

1. Introduction

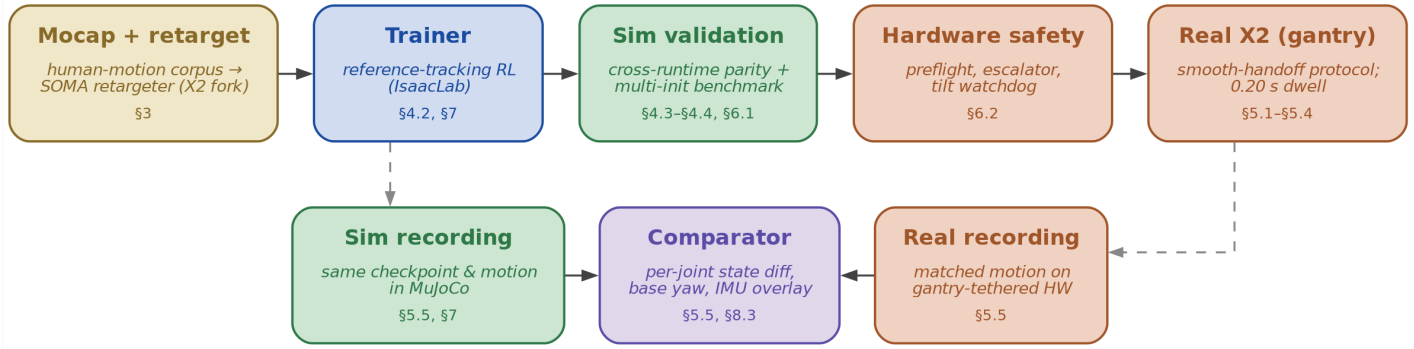

Figure 1. End-to-end pipeline. Top row: the forward path from human motion-capture data, through retargeting, training, and sim-side validation, to the gantry-tethered hardware. Bottom row: the closed-loop measurement that feeds calibrated fidelity claims back into the next training round. Section references locate each stage of the discussion. The two transfer bridges of the paper sit between sim-side validation and hardware safety gates (Bridge 1, §4) and between hardware safety gates and the real robot (Bridge 2, §5).

Humanoid whole-body controllers trained in simulation routinely promise sim-to-real transfer and routinely fail to transfer cleanly when the simulator, the policy, the deploy stack, and the real robot are integrated by different teams. Joint orderings, observation layouts, rotation conventions, contact geometries, motion-controller APIs, mode-transition primitives — any one mismatched contract can produce failure modes that look identical to bad training. Distinguishing a training problem from a deployment-side bug under these conditions is the first hard problem; doing it without bricking expensive hardware is the second.

This paper documents the integration of NVIDIA's reference-tracking Sonic policy onto the AgiBot X2 Ultra, a 31-DoF humanoid with 14-DoF dexterous hands and a sphere-foot lower-body variant. Sonic was originally developed and demonstrated on the Unitree G1; ours is the first publicly documented deployment of a Sonic-family policy on a non-G1 platform. The work fell into two sequential transfer problems — sim-to-sim and sim-to-real — connected by a validation substrate that prevented either of them from destroying hardware. Figure 1 places those two transfers, the upstream embodiment-integration step that precedes them, and the closed-loop measurement that calibrates the resulting digital twin into a single picture.

1.1 Two bridges, one substrate

Bridge 1 (IsaacLab → MuJoCo). Round-1 training (mesh-foot URDF in IsaacLab, sphere-foot MJCF in deploy) reached roughly sixteen thousand iterations. Interim MuJoCo evaluation looked acceptable near **~2k iterations** and **worsened monotonically** as training tracked the IsaacSim-matched distribution more tightly; by late training the policy **failed entirely in MuJoCo** and was **not deployable to hardware** — a costly lesson documented in our engineering logs. A trainer-side mirror ablation isolated a **foot-collider mismatch in the training distribution** as a major axis (G18); two additional bugs lived in the **deploy-side observation and ONNX adapter stack** (G20 6D-rotation flatten; G21 tokenizer layout) and produced MuJoCo failures that **tolerance-based parity checks could not see** because permutations preserve aggregate norms. We validate each fix by controlled fine-tune comparison; most starkly, the rotation fix alone unlocks an already-trained checkpoint — sub-four-second collapse becomes a full-clip walk — without any retraining (§4).

Bridge 2 (MuJoCo → real robot). The vendor motion controller exposes only a coarse start/stop pair, with no graceful-overlap command in the path we shipped. Naive parallel publishing produces roughly **1.6 s of audible *dual-publisher whir*** as two control sources fight the joint-command bus during ramp-down and MC resume. We compose the start/stop primitives into a finite-state handoff protocol — takeover-detector subscriber (with sensor-matching QoS), sentinel-file orchestration, recorder-backed MC-mode timeline as ground truth, and a **persistent-client mode escalator** at 20 Hz with **polled controller-reported mode** as confirmation — that brings the measured dwell to **0.20 s**, operator-imperceptible (§5).

Substrate. Both bridges sit on a two-layer validation substrate. On the simulator side, four cross-axis observation comparators check the policy's input pipeline against frozen-reference dumps. On the hardware side, a master preflight script gates bring-up behind pose, velocity, effort, IMU, gravity, and tilt envelopes, plus a ground-truth-verified mode-transition escalator and a tilt watchdog. One concrete substrate insight emerged from the rotation bug above: tolerance-based comparators silently pass channel permutations — the same numbers in different positions — and that is precisely how the bug survived months of *passing* parity checks. Element-wise positional assertions against frozen reference dumps are the rung that was missing (§6).

Closed-loop validation. A committed sim-to-real anchor archive of three matched simulator–hardware recordings, produced by a single canonical checkpoint, calibrates the resulting digital twin. The archive validates the kinematic chain and torso inertial model to within roughly five degrees per-DoF RMS and base IMU angular velocity to within roughly five percent. It also surfaces a large end-of-clip base-heading gap on the locomotion anchor, which we are careful to read as *consistent with* unmodelled contact / friction / free-base integration error rather than as direct evidence of any one of them: the hardware base is gantry-pinned and the simulator base is free, so the present anchor archive cannot disambiguate the physics hypothesis from the boundary-condition asymmetry (§5.5, §9.1). The closed-loop anchors do not, and cannot, validate foot contact, floor friction, actuator saturation, joint friction at low velocity, sensor latency or noise, or free-base dynamics, and we name those limitations explicitly (§9, §10).

1.2 Contributions

1. *First public Sonic-family policy on a non-G1 humanoid*, trained on an X2-curated subset of **BONES-SEED** (~2,550 SOMA-retargeted clips; §3.2), with X2-specific motion-retargeting fixes that surface and remedy two structural issues in the upstream tooling (§3).
2. *Three independent sim-to-sim bugs identified, fixed, and validated by controlled fine-tune comparison*, together with a generalisable methodology for distinguishing training-side from deployment-side failure modes when the symptoms are indistinguishable (§4).
3. *Measured dual-publisher motor whir and its remedy:* naive handoff produced **~1.6 s** audible bus conflict; a persistent-client escalator with **controller-reported mode** as ground truth reduced measured **joint-default dwell** to **0.20 s** (8×), documented on live hardware. A finite-state smooth-handoff protocol built atop start/stop primitives only, with the wire-level subtleties that make it non-trivial against a closed-API motion controller (§5.1–§5.4).
4. *A two-layer validation substrate* — sim-side parity gates and hardware-side safety envelopes — with the documented insight that tolerance-based comparators miss channel permutations and that element-wise positional assertions are the rung that closes the gap (§6).
5. *A committed sim-to-real anchor archive* with calibrated, falsifiable digital-twin fidelity claims and an explicit catalogue of the model components for which the archive is, and is not, evidence; together with a four-class failure-mode taxonomy — bug, physics-divergence, handoff, tuning — intended as an organising principle for subsequent humanoid deployments (§5.5, §8, §9).

2. Related work

NVIDIA's public Sonic stack targets the **Unitree G1** with the same reference-tracking recipe we adopt unchanged: architecture, rewards, and training loop are upstream defaults. That setup has already demonstrated strong **real-world** whole-body behaviour on G1, which is the empirical anchor we inherit.

This paper applies that same training and deployment methodology to the **AgiBot X2 Ultra** — a different kinematic chain, different contact assets, and a different vendor motion-controller surface — and documents what had to change for the port to remain safe and measurable. Our related-work stance is therefore not a new algorithm but a **cross-embodiment integration** story: when the robot is no longer the seed platform, integration artefacts dominate the critical path.

3. The X2 Ultra integration

The X2 Ultra is a 31-DoF humanoid: twelve-DoF legs, three-DoF waist, two-DoF head, and seven-DoF arms (including a three-DoF wrist), with fourteen further actuated dexterous-hand degrees of freedom that we hold static at neutral pose for all reported runs. From a distance the kinematic chain resembles that of the Unitree G1, the platform on which most of the upstream tooling we use was originally developed. Up close, three differences in the upper body matter, because they intersect with assumptions baked into the upstream

retargeting and observation pipelines: the elbow rotates about the lateral axis on the X2 rather than the longitudinal one; the wrist chain is ordered yaw-pitch-roll rather than roll-pitch-yaw; and the wrist pitch and roll ranges are roughly a third of the comparison robot's. Foot collision geometry also differs between the two simulators we use, but that difference is a Bridge-1 issue and is treated in §4 rather than here.

3.1 Adapting SOMA (NVIDIA retargeter)

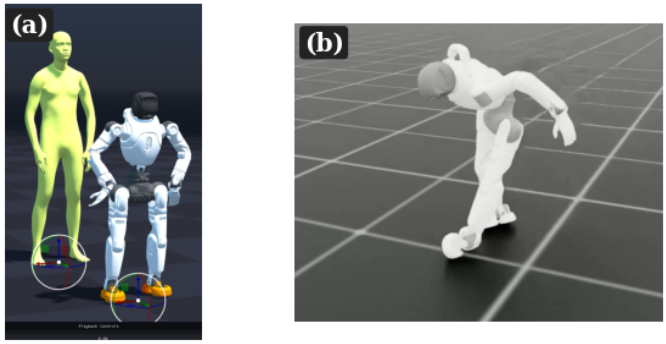

Figure 2. Adapting NVIDIA **SOMA** (inverse-kinematics retargeter) to the X2. (a) Side-by-side pose-match viewer used to verify, before any objective-weight tuning, that the X2 (right) is kinematically capable of the desired source pose (left). The wrist axis indicators on both feet visualise the orientation discrepancy induced by the wrist chain ordering and reduced wrist range discussed in §3. (b) The *arm-flip* failure mode discussed in §3.1: walking motions retargeted under the comparison robot's high rotation-objective weights drive the solver to a kinematically valid but visually wrong solution at the negative limit of shoulder pitch. Lowering rotation weights by an order of magnitude resolves the flip without retraining.

We use NVIDIA's **SOMA** retargeter — a publicly available tool that maps human motion-capture data onto humanoid skeletons by solving a constrained inverse-kinematics problem with weighted position and rotation objectives per body segment — configured per-robot through joint-offset and weight files. Adapting the configuration files to the X2 surfaced two structural failure modes that recur, in our experience, on any humanoid whose configuration is initialised from one developed for a robot with appreciably different upper-body kinematics.

Arm-flip-over-head. Walking motions retargeted onto the X2 from configuration files seeded by the comparison robot produced solutions in which both arms inverted fully over the head rather than swinging at the sides (Figure 2b). The cause is that the upstream configuration encodes high-weight rotation objectives on the forearm and hand segments, calibrated against world-frame orientations natural for an elbow rotating about one axis. On the X2's lateral-axis elbow those orientations are not reachable by the natural arm configuration, and the inverse-kinematics solver finds a kinematically valid but visually wrong alternative at the negative limit of shoulder pitch. The remedy is structural rather than numerical: lowering the rotation weights on those segments by an order of magnitude, so that the position objectives dominate, allows the solver to find natural arm configurations regardless of elbow axis. The general lesson — that rotation-weighted objectives bake in axis assumptions and should be retuned per-robot when the axis assumptions change — applies broadly, and is the diagnostic to apply first when porting any inverse-kinematics-based retargeter across humanoid kinematic chains.

Wrist hard-clamping. After the arm-flip was resolved, retargeted wrist trajectories were stuck at one or both physical limits for the majority of every clip. The proximate cause is that the offset frames for the hand segments had been carried over from the comparison robot, where the wrist range is wider; on the X2 those offsets place the natural neutral wrist orientation outside the physical envelope, causing the joint-limit clamper to fire on every frame. The fix is a pair of small Euler-angle corrections to the hand offset frames, one per hand, found by a manual axis sweep. Gradient-based optimisation of these offsets failed because the limit-clamper's discontinuity destroys the local gradient signal — a useful diagnostic in its own right when porting offset-based retargeting across robots with different limit envelopes.

URDF/MJCF mismatch. The vendor's robot descriptor drop ships three side-by-side variants — a mesh-foot URDF, a sphere-foot URDF, and a sphere-foot MJCF for MuJoCo. Our integration grabbed the mesh-foot URDF for IsaacLab training because its filename was the obvious candidate, and the sphere-foot MJCF for MuJoCo deployment; nobody opened the third file. The geometric mismatch between mesh and sphere feet is the root cause of the dominant Bridge-1 physics gap (§4.2) and is, with hindsight, the single most consequential precursor decision documented in this paper. We recommend that future descriptor drops ship a short URDF↔MJCF coherence note documenting contact-relevant link parity on every contact-relevant body before any training run.

Both retargeting fixes were verified by playback in side-by-side viewers of the trainer and the retargeting tool, with manual pose-matching to confirm that the X2 was kinematically capable of the desired pose before any objective weights were adjusted (Figure 2a). Neither fix is invasive; both are quotable verbatim by anyone porting the same retargeter to a humanoid with G1-derived configurations and different upper-body kinematics.

3.2 Datasets and assets (BONES-SEED and the X2 subset)

Human motion enters our stack through **BONES-SEED**, an open corpus hosted on Hugging Face by Bones Studio: **142,220** human motion clips with accompanying processed artefacts (see References). We do **not** train on that full enumeration on the X2. Instead we curate an **X2-specific working subset of roughly 2,550** clips — approximately **2,000** locomotion-and-manipulation and **550** standing-manipulation motions — each passed once through the SOMA retargeting procedure of §3.1 to the X2 skeleton, then through the IsaacLab motion-PKL pipeline used by this repository's **GR00T-style** Sonic training configuration (four motion encoders, including the SOMA skeleton channel). The distinction matters for reproducibility: the **full BONES-SEED cardinality** is a property of the upstream dataset; the **~2,550** figure is the materialised X2 bundle we actually optimised against.

Asset variants used during the work include a reference URDF with mesh foot collision, a fine-tuning URDF with sphere foot collision matched to the deployment simulator, and the deployment MJCF itself. The geometric mismatch between the reference URDF's mesh feet and the deployment MJCF's sphere feet is one of the three Bridge-1 issues discussed in §4.

4. Bridge 1: Sim-to-Sim (IsaacLab → MuJoCo)

The first deployment bridge connects the training simulator (IsaacLab) to the evaluation simulator (MuJoCo). Failures on this bridge are easy to misread as physics-tuning problems but turn out, on inspection, to be observation- and encoding-layer bugs: the right numerical values arriving at the policy in the wrong positions, or under the wrong layout convention. We encountered three independent instances of this pattern — a foot-collision-geometry mismatch between simulators, a transposed flatten convention on the policy's heading representation, and a layout-adapter error at the boundary between the runtime and the exported model. Each initially appeared to be a sim-to-real physics gap, and each absorbed weeks of compliance and PD-tuning effort before the underlying cause was localised.

4.2 Foot collision geometry across simulators

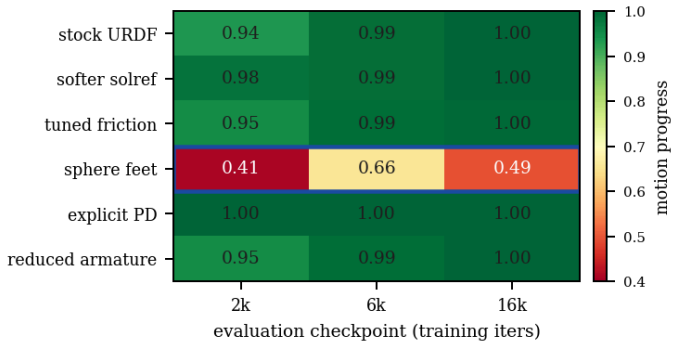

Figure 3. Trainer-side ablation isolating foot-collider geometry. Six modifications of the trainer URDF, evaluated against three checkpoints; cell value is mean motion progress. The boxed row, in which the trainer is forced onto the deployment simulator's foot geometry, is the only modification that reproduces the deployment failure inside the trainer.

Round-1 training used the upstream URDF, which models each foot as a single mesh collision body. The MuJoCo deploy environment, mirroring an earlier hand-tuned configuration, modelled each foot as a cluster of small spheres distributed across the sole. Both representations are geometrically reasonable; they differ in their contact regime — sphere clusters produce many simultaneous contact points whereas the mesh produces broader distributed contact — and the policy was trained against only one of them.

For most of round 1 we attacked the resulting transfer gap from the deployment side, softening MuJoCo's contact parameters in successive rounds. None of these reproduced the trainer's behaviour. The diagnostic that finally localised the cause was the inverse of the usual move: rather than soften MuJoCo, we *hardened* IsaacLab to reproduce the MuJoCo failure inside the trainer. A six-row ablation of the IsaacLab URDF, crossed with three checkpoints, isolated foot-collider geometry as the only modification that reproduced the deployment failure within the trainer (Figure 3, Table 1). Refining the URDF to match the deployment foot model and continuing training for an additional two thousand iterations roughly doubled time-to-fall on the deployment benchmark; a clean retraining round on the matched assets later saturated our evaluation cap entirely.

*Table 1. Trainer-side ablation that localised the foot-collider mismatch. Each row toggles a single deploy-side axis on inside the trainer, evaluated against three checkpoints from the same training run; **progress** is the mean fraction of motion completed across fifteen motions and **term** is the fraction of motions terminated by the fall criterion. The matched-foot-geometry row (bold) is the only single-axis modification that reproduces the deployment-side failure within the trainer; row A5 combines all three axes and is strictly less catastrophic than A3 alone, showing that the deployment-side PD regime partially masks the foot-collider failure rather than compounding it. The 16k checkpoint is the worst of the three under spheres — the same monotonic-degradation direction as observed in MuJoCo — further evidence that the cause is asset-side rather than training-stage-dependent.*

Row	2k checkpoint		6k checkpoint		16k checkpoint		What it toggles inside the trainer
	progress	term	progress	term	progress	term	
`A0_isaac_stock`	0.94	0.07	0.99	0.07	1.00	0.00	stock training configuration
`A1_no_dr_no_noise`	0.96	0.07	1.00	0.00	1.00	0.00	training DR and observation noise disabled
`A2_frictionloss`	0.96	0.07	1.00	0.00	1.00	0.00	A1 + joint friction-loss term
`A3_sphere_feet`	**0.41**	**0.93**	**0.66**	**0.53**	**0.49**	**0.67**	**A1 + sphere-foot collision URDF (matches deployment)**
`A4_explicit_pd`	1.00	0.00	1.00	0.00	1.00	0.00	A1 + explicit-PD actuator regime, ankle KP×1.5
`A5_full_mirror`	0.66	0.60	0.71	0.60	0.63	0.67	A1 + all three deploy-side axes together

We initially attributed nearly all of round 1's transfer gap to this asset mismatch. Subsequent work showed this attribution was overstated — a second observation-layer bug, discussed next, accounted for most of the residual gap and had been partially absorbed into the asset-mismatch fix. The retraction is recorded here because it shaped the methodology that follows: any single-cause attribution made before all observation-layer bugs are isolated is provisional, and asset audits are necessary but not sufficient.

4.3 Heading-frame encoding across pipelines

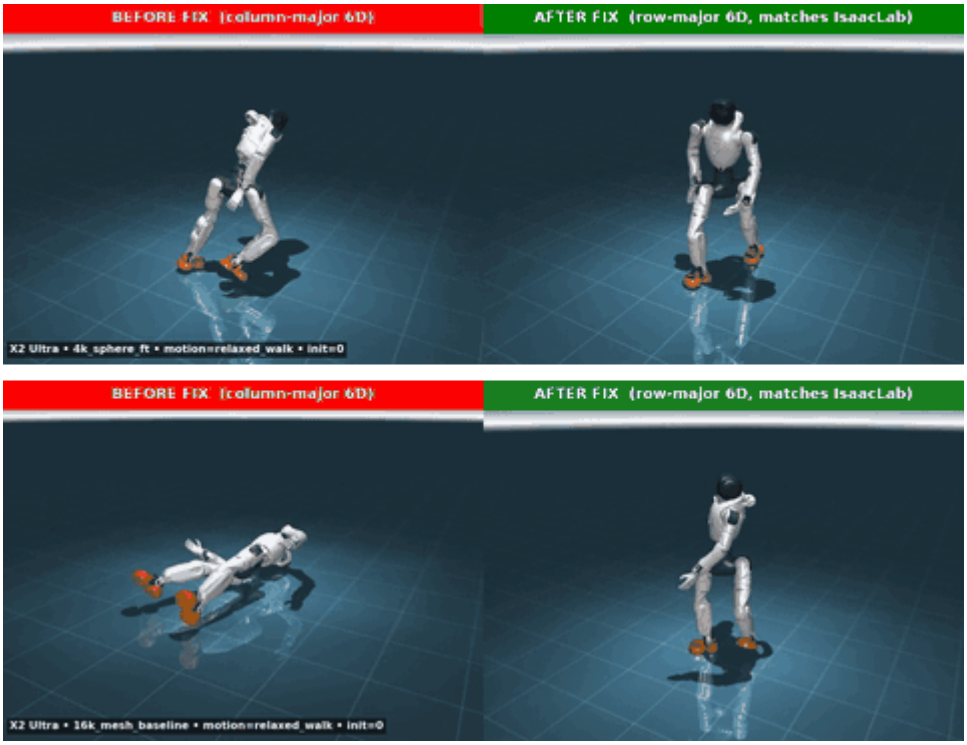

Figure 4. The heading-encoding correction, qualitatively. End-of-clip frames before (left, column-major flatten) and after (right, row-major flatten matching the trainer) the one-line correction, on two checkpoints from the same training run. Top: a four-thousand-iteration sphere-foot fine-tune. The pre-fix policy collapses within 3.8 s; the same weights, with the flatten order corrected, walk the full clip without retraining. Bottom: a sixteen-thousand-iteration mesh-foot baseline. Pre-fix collapse at 2.22 s reproduces the original ablation measurement to the centisecond, confirming bit-exact reproduction; post-fix survival lifts to 12.24 s, with the residual gap attributable to the foot-collider mismatch of §4.2.

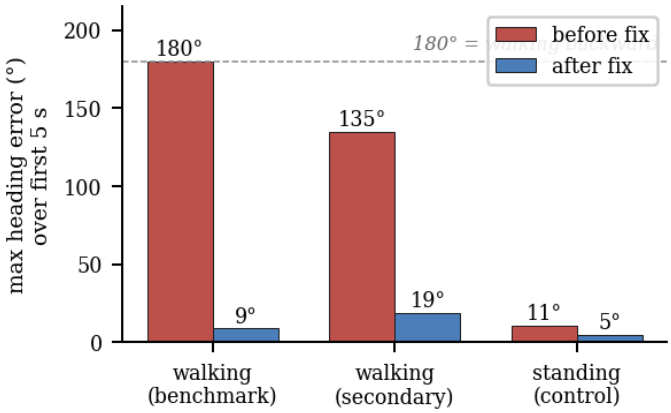

Figure 5. The heading-encoding correction, quantified. Maximum deviation from the reference heading over the first five seconds of three motions, before and after the one-line correction. On the benchmark walking clip, deviation drops from 179.9° to 8.8° without retraining; the same checkpoint that had collapsed at 2.22 s on the same clip subsequently sustained 12.24 s.

The policy consumes a flattened representation of the desired heading rotation as part of its tokenised observation. The first two columns of a rotation matrix are flattened into a six-element vector — the standard six-dimensional rotation representation used in this family of policies. There are two natural ways to perform this flatten: row-by-row across the two columns, or by concatenating the columns end to end. The training pipeline used the former; the deployment observation builder, written independently and validated against tolerance comparators, used the latter. Both produce a six-element vector containing the same scalar entries; they differ only in which entry sits at which position.

The policy treats positions in its observation as semantically distinct features. From the policy's point of view, the heading channel was therefore, throughout deployment, a permuted (and no longer rotation-valid) vector. Standing motions placed no demand on the heading channel and concealed the bug; walking motions exercised it immediately. On a representative benchmark clip, deviation from the reference heading reached 179.9° within the first half-second and the policy walked confidently in the wrong direction. The bug evaded our cross-pipeline tolerance comparators because they aggregated across vector positions — root-mean-square and maximum-element distances are invariant to permutation when the same entries appear in different positions.

Correcting the flatten order to match training was a one-line change with no retraining required. The same checkpoint that had collapsed at 2.22 s on the benchmark clip subsequently sustained 12.24 s, and yaw error over the affected window dropped from 179.9° to under 9° (Figure 4, Figure 5). The methodological lesson generalises beyond this particular channel: an observation-pipeline bug that produces correct values in incorrect positions silently passes any comparator that aggregates across positions. Element-wise positional assertions against frozen reference dumps, captured at a known-good initial state, are the only check we found that catches this class of error reliably.

4.4 The runtime-to-export adapter boundary

The deployed policy is exported via ONNX from its PyTorch reference implementation. The exported graph expects its tokenizer input in a per-frame interleaved layout: at each timestep the command sub-vector and the orientation sub-vector are concatenated, and successive timesteps are stacked. The deployment runtime, in contrast, naturally constructs a grouped layout: all command sub-vectors first, then all orientation sub-vectors. A small adapter wrapper sat between the deployment observation builder and the ONNX session to bridge

the two layouts. The wrapper's reshuffle was the inverse of the one actually required.

Both halves of this seam had been independently validated. The export step produced agreement against the PyTorch reference to within roughly half a microradian, and the live observation builder agreed with a frozen IsaacLab dump to within a similar tolerance. Validating the components individually did not validate their composition. With the prior heading-encoding bug fixed but the wrapper still in place, action-level comparison against the PyTorch reference exhibited disagreements of several radians per timestep over a 150-tick rollout. Removing the wrapper recovered the documented sub-microradian agreement. A separate observation worth recording is that ONNX export under the relevant tooling performs constant folding sufficient to bake wrapper layout assumptions into the exported graph, so re-exporting from the same checkpoint after a code-side change is a precondition for parity.

The lesson is structural: when two correctly-validated components are connected through an adapter, the adapter is the location most likely to harbour the bug, and the act of validating the components individually inadvertently licenses the assumption that their composition is correct.

4.5 Methodology synthesis

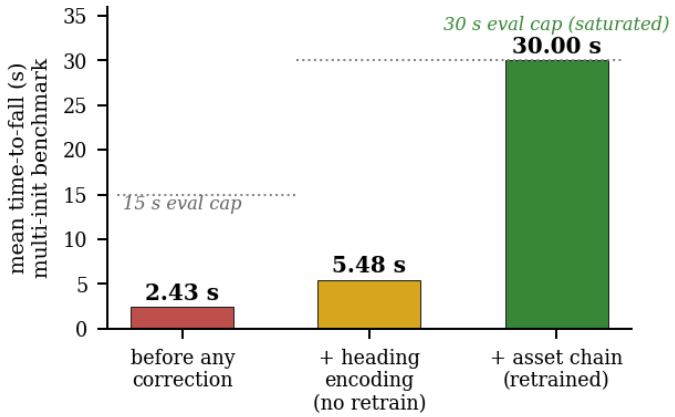

Figure 6. Cumulative impact of the three corrections on the multi-initialisation simulator benchmark. Mean time-to-fall: 2.43 s before any correction; 5.48 s after the heading-encoding correction applied without retraining; thirty-second evaluation cap saturated after retraining on the corrected asset chain.

The three bugs share a generative pattern. Each was concealed by an adjacent activity that had a credible appearance of progress: physics-side compliance tuning concealed the asset mismatch; the asset-mismatch fix partially absorbed the heading-encoding error and licensed an over-confident attribution; per-side validation of the export and runtime concealed the layout error at the boundary between them. Three procedural conclusions follow.

First, deployment-side debugging should isolate observation-layer bugs from physics-tuning hypotheses before any tuning effort is expended. The diagnostic question is not whether the deployment behaves like the trainer, but whether the deployment observation builder reproduces the trainer's observation builder element by element on a frozen reference frame. If the answer is no, no amount of physics tuning will close the gap. Second, when deployment-side softening fails to make progress, hardening the trainer to reproduce the deployment failure is a more productive next step. Third, every observation channel should be subject to element-wise positional comparison against frozen reference dumps. Aggregate comparators are necessary but not sufficient; they are silent on permutations.

The cumulative impact of the three corrections is summarised in Figure 6. On a multi-initialisation evaluation, the pre-correction policy survived a mean of 2.43 s before falling. A single observation-layer correction (heading encoding), applied without any retraining, lifted that to 5.48 s. A subsequent retraining round on the corrected asset chain saturated the 30-second evaluation cap entirely.

5. Bridge 2: Sim-to-Real (MuJoCo → real robot)

With the observation-layer bugs of §4 corrected, what remained of the deployment problem was almost entirely concerned with the *handoff* between our policy and the robot's vendor-supplied motion controller, rather than with policy–physics fidelity. This section describes the constraints imposed by the vendor controller, the finite-state handoff protocol we built within those constraints, the wire-level subtleties that surfaced during commissioning, and the closed-loop sim-vs-real measurements that calibrate how well the digital twin reproduces the real system.

5.1 The motion-controller API constraint

The X2 Ultra ships with a vendor motion controller (MC) that owns the joint command bus by default. Our policy is, from the bus's perspective, a second publisher. The MC's *public* handoff interface exposes only two operations: a request to silence the controller, and a request to restart it. The restart sequence boots the controller through a passive holding state, an intermediate joint-default state, and finally the standing default, over three to five seconds. There is no public operation for overlapping command authority between two sources, no public mechanism for registering an external policy as a recognised input, and no additional documented controls beyond those two operations.

Even within that interface, naive parallel publication of policy commands during the controller's transitions produced **operator-audible *dual-publisher whir*** — two writers on the same joint-command bus during the ramp-down and MC-resume window, with motors audibly fighting as setpoints disagree. The remainder of this section describes how we reduced measured **joint-default dwell** from **1.60 s to 0.20 s** (8×) and how we structured the surrounding orchestration so the human operator sees a single continuous handoff. The authoritative diagnostic for that window is the **recorder's MC-mode timeline** together with **polled controller-reported mode**, not process logs alone.

5.2 The dual-publisher whir and the handoff protocol

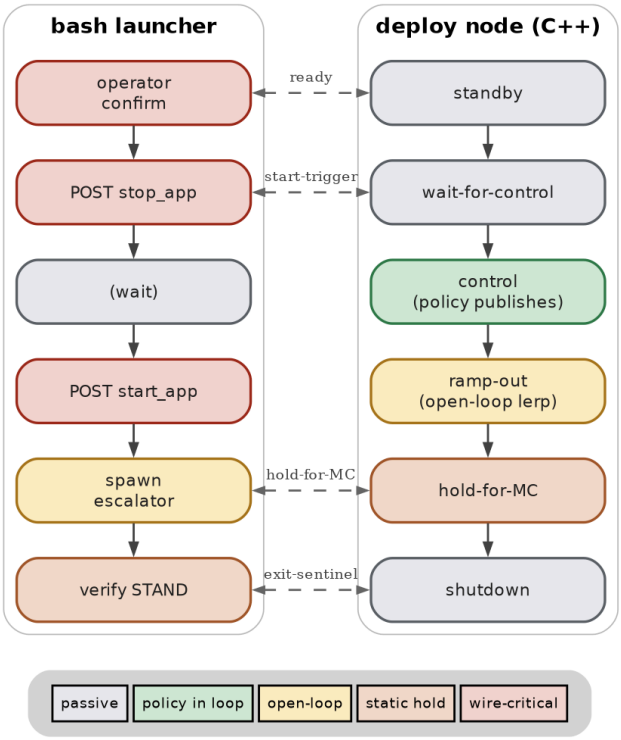

Figure 7. Handoff protocol as a two-lane finite-state machine. The deploy-node lane (right) is colour-coded by what the policy is doing in each state — passive, in the loop, open-loop interpolation, or static hold. The launcher lane (left) is colour-coded by criticality — quiescent, wire-critical, transitional, or static. Horizontal double-arrows mark filesystem sentinels that synchronise the two lanes at protocol boundaries.

Dual-publisher whir. During active policy control, ramp-out, static hold, and motion-controller resume, both our deploy node and the vendor motion controller can briefly command the same bus. The failure mode is not subtle: the operator hears **motor whir** — sometimes a full **~1.6 s** of it when mode transitions were driven from **repeated short-lived controller clients** (with per-invocation overhead), capping effective poll rate near **3 Hz**. That dwell is long enough to be alarming on a humanoid and is unacceptable for a productized handoff. The fix is architectural: a **persistent-client mode escalator** that keeps the vendor mode-set and mode-read interfaces open and polls at **20 Hz**, treating **controller-reported mode** as ground truth. On an otherwise-identical handoff, measured joint-default dwell dropped **1.60 s → 0.20 s** — the shorter window is operator-imperceptible in the audio signature; the longer one was the audible "couple of seconds of whirring" reported on hardware.

Handoff protocol. The protocol is a finite-state machine implemented across two cooperating processes: the deploy node, which holds the policy in its inference loop, and a launcher script, which orchestrates around it. The deploy node passes through a quiescent warm-up phase, an initialisation phase, a wait for subscriber readiness, the active control phase during which the policy publishes joint targets, a fixed-duration ramp during which targets are interpolated towards the controller's standing pose, and a static hold at that pose with controller-matched stiffness and damping. The launcher gates operator confirmation, requests controller silence at the appropriate moment, signals the deploy node to begin publishing, requests controller restart at the moment the policy hands the pose back, and verifies via ground-truth polling that the controller has resumed authority before allowing teardown. Coordination between the two processes is via a small set of filesystem sentinels that mark protocol boundaries — readiness, start authorisation, hand-back, and exit — without coupling the processes to each other's internals. Figure 7 shows both lanes and the sentinel edges between them.

The eight-fold reduction in the dual-publisher window is attributable to that single architectural change — replacing the shell-driven loop with the persistent-client escalator. The escalator decides success by reading back the controller's reported mode rather than relying on the service's response code.

5.3 Wire-level subtleties

Two observations arose during commissioning that materially affect handoff correctness and that are not, to our knowledge, documented elsewhere in the literature.

The first concerns the ramp from policy authority to standing pose. An initial implementation interpolated joint targets and joint stiffness simultaneously toward the standing-mode values over the ramp window. The result was a transiently under-damped system tracking a moving setpoint, manifesting as audible motor whir and visible end-effector ringing. Lerping only the position target and snapping the stiffness change to the moment of pose-error convergence at the end of the ramp eliminated the artefact.

The second concerns the static hold at standing pose. The stiffness values used during active policy operation are lower than what is needed for a robust passive hold against gravity, and a passive hold under those values exhibits visible postural sag on certain joints — most notably the waist and knee. The protocol therefore snaps stiffness to higher static-hold values at the moment of static-hold entry, where pose error is small enough that the stiffness change is not perceptible.

5.4 Output-side filtering and target clamping

Two parameters of the deploy stack proved consequential to operator-perceived motion quality. A first-order low-pass filter applied to outgoing joint targets, after the safety stack and immediately before the bus, allows the operator to trade fine-grained policy expressivity against bus-level smoothness. We measured the trade-off by sweeping the filter's cutoff on a fixed motion at a fixed clamp setting; lowering the cutoff from 8 Hz to 5 Hz reduced upper-body command jitter and peak body angular velocity but increased lower-body command activity as the legs assumed a larger share of the balance work needed to track the wider arm trajectories permitted by a relaxed clamp. We use 8 Hz by default and 5 Hz for runs where upper-body smoothness matters more than the residual leg jitter.

The second parameter, a per-tick safety clamp on the magnitude of joint-target deviation from the standing pose, was set conservatively for early bring-up and turned out to be silently clamping every wrist and shoulder-yaw joint in steady state. The clamp does not register as policy clipping in the deploy log because the log instruments raw policy output rather than the post-clamp target. Widening the clamp from 0.30 rad to 0.80 rad gave the policy access to its trained workspace; the low-pass filter made the wider clamp tractable in terms of bus jitter.

5.5 The closed-loop sim-vs-real anchor archive

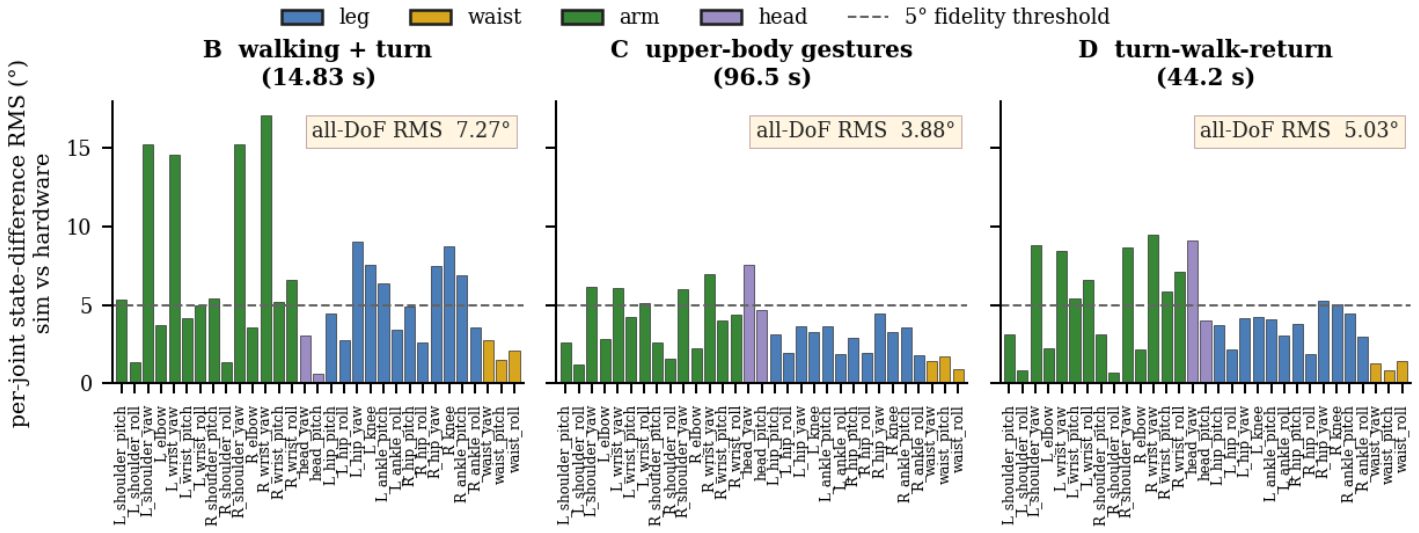

Figure 8. Per-joint state-difference RMS between simulator and hardware across the three matched recordings. Each panel is the 31-DoF kinematic chain ordered consistently (legs, waist, head, arms); colour encodes joint group; the dashed line marks the five-degree fidelity threshold of §8.4. The upper-body gestural recording holds nearly every joint under the threshold. The closed-path locomotion recording does the same with the exception of two leg degrees of freedom near contact transitions. The walking-with-turn recording holds the limbs but is dominated, in its all-frame statistic, by an integrated divergence in base heading addressed in §9.1.

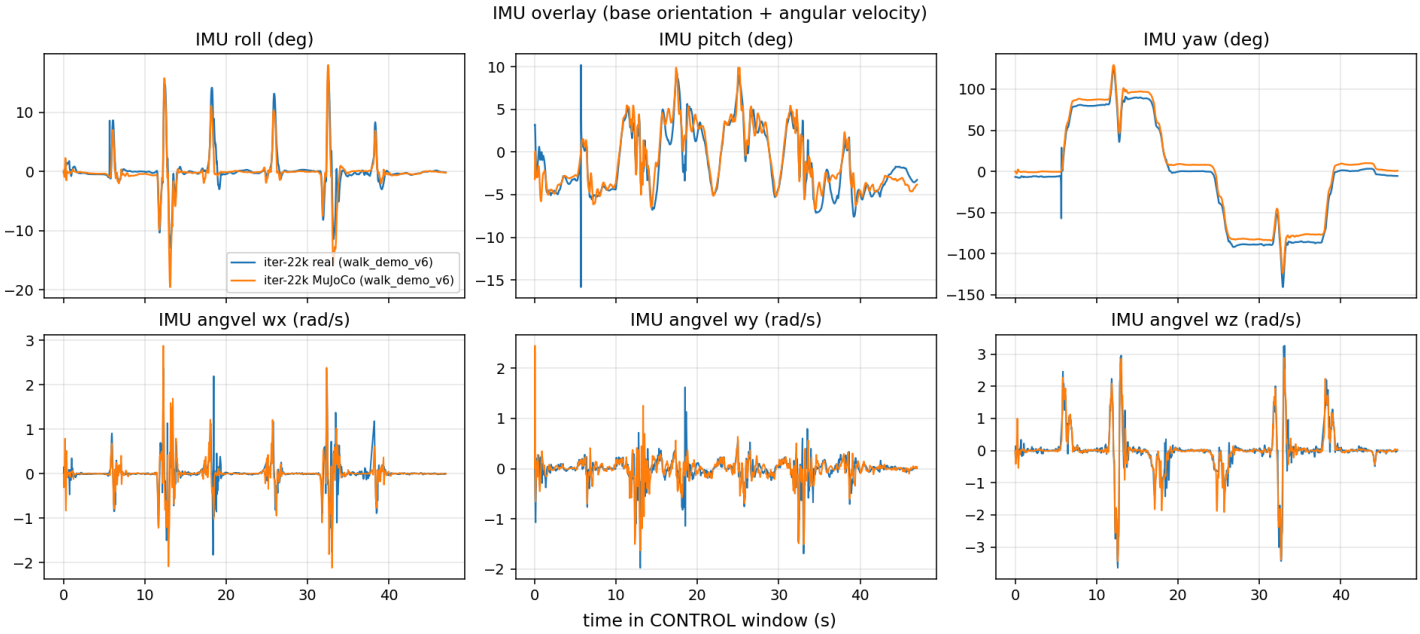

Figure 9. IMU overlay for Anchor D (turn-walk-turn-return on a closed path, 44.2 s). Top row: torso roll, pitch and yaw, in degrees, on the real robot and the matched MuJoCo run. Bottom row: the three components of the base angular-velocity vector. The orientation channel of the IMU and its angular-velocity components track the simulated trajectory closely throughout the recording, qualitatively confirming that the torso inertial model and the orientation channel of the inertial sensor transfer across the bridge.

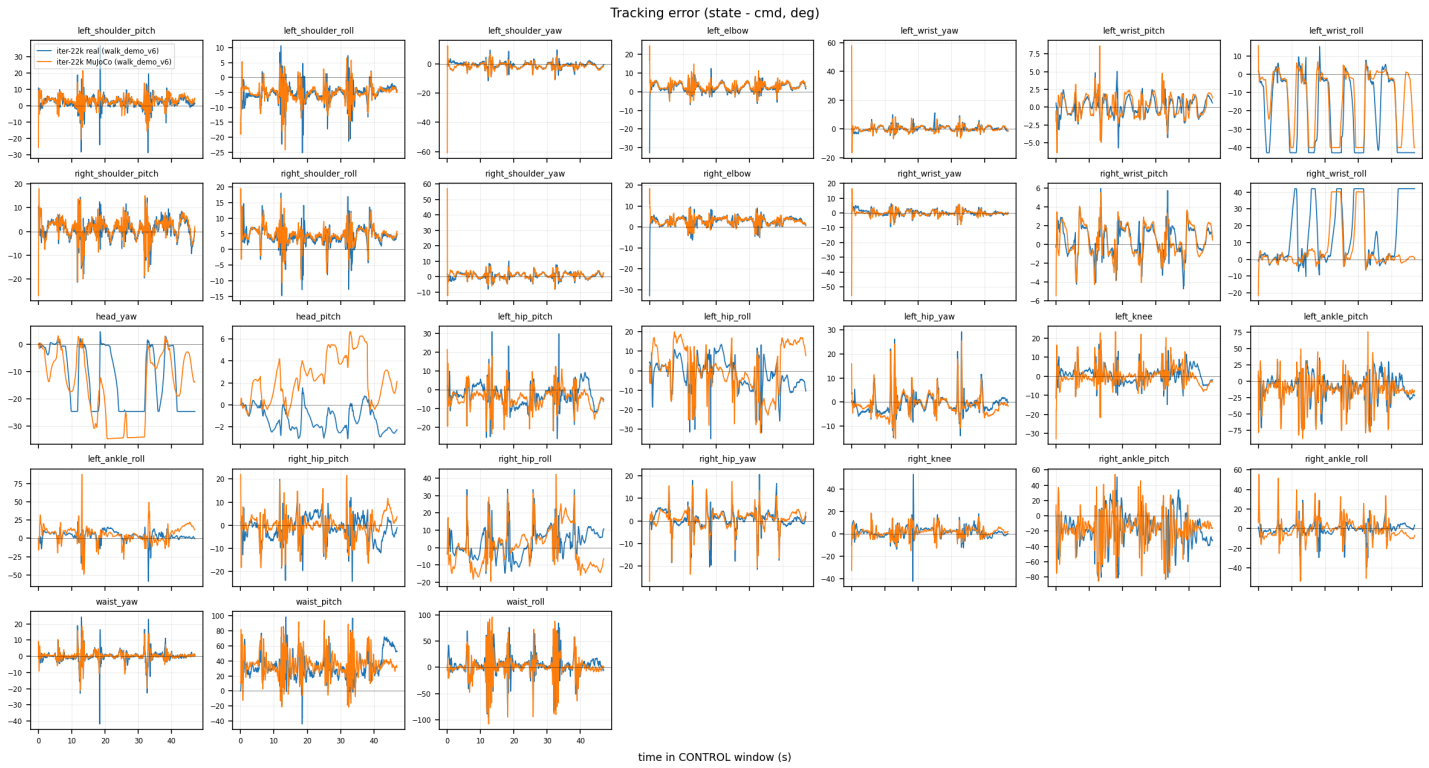

Figure 10. Per-DoF reference-tracking error on Anchor D (44.2 s closed-path locomotion). Each panel plots, for one joint, the instantaneous difference between the policy's commanded position and the joint state actually achieved, in degrees, on the real robot and in the matched MuJoCo run. The two traces are near-identical in shape and amplitude on every joint — evidence that the actuator-level PD response of the digital twin reproduces the real robot's tracking behaviour under the same policy command stream, a stronger claim than the per-joint state agreement of Figure 8 because it isolates controller dynamics from kinematic correctness.

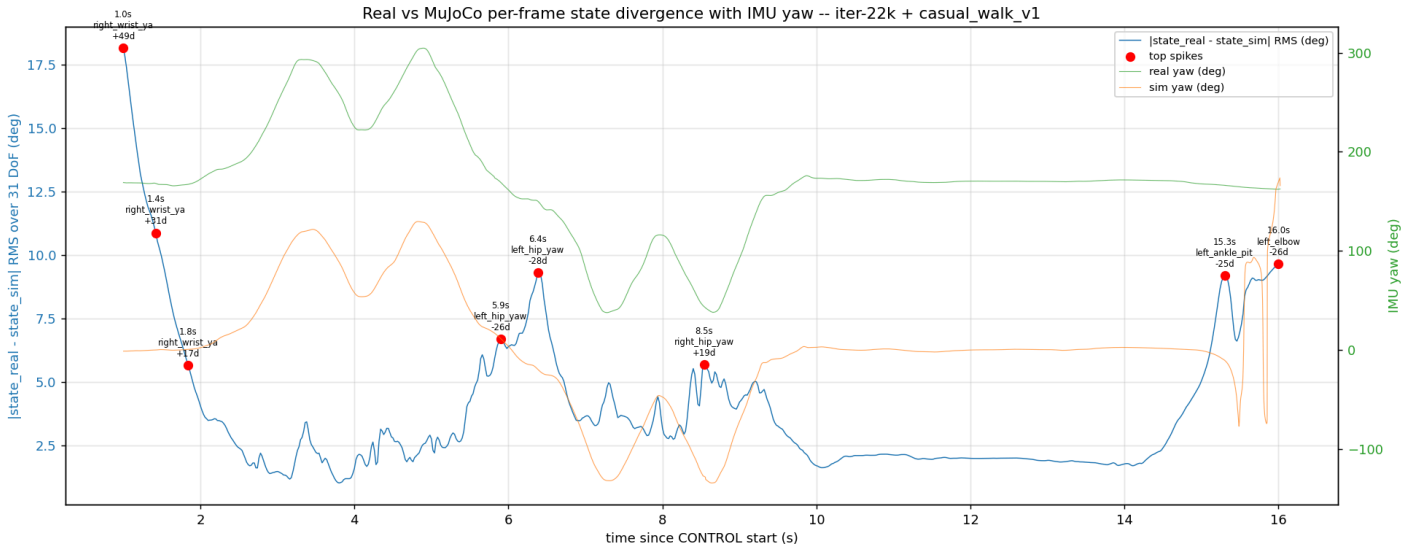

Figure 11. Base-yaw divergence on Anchor B. The blue trace is the per-frame all-DoF state difference (RMS over 31 joints) between simulator and hardware; the orange and green traces are the simulated and real base-yaw angles respectively. Per-cycle limb kinematics agree to within single-digit degrees throughout the sixteen-second recording, but the integrated base trajectories accumulate a 166-degree gap. The hardware base is gantry-pinned and the simulated base is free, so this divergence is consistent with either contact / friction / free-base integration error in the digital twin or with the boundary-condition asymmetry between the two halves of the comparison; §9.1 unpacks why this finding should not be read as a clean falsification of any single component.

The empirical foundation of the digital-twin fidelity claim of §8 and §9 is a committed *anchor archive*: three pairs of matched recordings, each pair differing only in the physics environment. The two halves of each pair share the same exported policy checkpoint, the same observation builder, the same motion playlist, the same target low-pass filter, and the same per-tick safety clamp. The real half is recorded on hardware with the gantry strap engaged; the simulated half is recorded by the same deploy node configured to drive a parity-tuned MuJoCo digital twin in place of the bus. The archive is the only artefact in the paper that quantifies, rather than describes, the residual sim-to-real gap.

The three anchors and the headline numbers are summarised in Table 2.

Table 2. The three matched simulator–hardware recordings that constitute the closed-loop anchor archive. State-diff RMS is the all-frame, all-DoF root-mean-square of the per-joint position difference between the simulated and real recordings, in degrees.

Anchor	Motion	Real / Sim duration	State-diff RMS (all-frame)	Headline finding
B	walk with mid-clip turn-around (14.8 s)	16.03 s / 14.02 s	**7.27°** (4.74° after t>1 s)	First powered walk on hardware. Per-cycle leg kinematics agree, but the integrated base heading diverges by 166° in simulation against −6.5° on hardware. The hardware base is gantry-pinned and the simulator base is free; the divergence is therefore consistent with either contact / friction / free-base integration error or with the boundary-condition asymmetry between the two halves, and we are careful not to read it as a clean falsification (§9.1).
C	upper-body gesture reel, lower body standing (96.5 s)	102.00 s / 102.02 s	**3.88°**	Tightest sim-to-real of the three; upper-body kinematics transfer near-perfectly.
D	turn-walk-turn-return on a closed path (44.2 s)	48.99 s / 52.02 s	**5.03°**	Strongest locomotion result; both worlds traverse the home-loop choreography and return within three degrees of the starting heading.

Figure 8 disaggregates the all-frame statistic across all 31 joints. The gestural recording (Anchor C) and the closed-path locomotion (Anchor D) hold nearly every joint inside the five-degree fidelity threshold defined in §8.4. The walk-with-turn recording (Anchor B) shows tight per-cycle agreement on the limbs but is dominated, in its all-frame statistic, by an integrated divergence in base heading that §9.1 returns to.

Three complementary views of the archive are reproduced from the supplementary material as figures. The IMU overlay of Anchor D, in Figure 9, shows that the orientation channel of the inertial sensor — roll, pitch, and yaw — and its angular-velocity components track the simulated trajectory closely throughout the forty-four-second walking sequence, qualitatively confirming that the torso inertial model and the orientation channel of the IMU transfer across the bridge. The per-DoF *tracking error* of the same anchor, in Figure 10, addresses a question that the state-difference statistic cannot: it overlays the policy's commanded-versus-achieved joint trajectory in simulation and on hardware and shows that the two are near-identical on every joint, evidence that the digital twin reproduces not only the kinematic chain but the actuator-level PD response under the policy's command stream. The combined state-divergence and yaw plot of Anchor B, in Figure 11, makes the most discussed asymmetry of the archive visible at a glance: the per-frame state divergence (blue) is bounded by single-digit degrees over the full sixteen seconds of the recording, while the simulated and hardware base-yaw traces (orange and green) accumulate a 166-degree gap over the same interval. Per-cycle leg kinematics are matching; the integrated base trajectory is not. Section 9.1 unpacks why this gap is consistent both with contact / friction / free-base integration error in the digital twin and with the boundary-condition asymmetry between a gantry-pinned hardware base and a free simulator base, and why we therefore decline to read it as a clean falsification of any single component of the model.

The matched recordings validate the kinematic chain, the torso inertial model, the position-control signal path, and the orientation channel of the inertial sensor. They do not validate, and we are careful not to claim they validate, the foot contact model, friction, free-base dynamics, actuator saturation under high acceleration, or the noise spectrum of any sensor. §8.4 and §9 catalogue the unvalidated components exhaustively; §10 lists five policy-free bench tests that would convert the indirect, closed-loop evidence offered by the archive into direct, isolated, component-level validation of the digital twin.

5.6 The exit ramp as a physics observation

A consistent feature of all three matched recordings is a divergence at end-of-clip that is not policy-attributable. When the deploy node exits its active control phase, the protocol interpolates joint targets open-loop from the last policy command toward the standing pose over a fixed two-second window. The policy is not in this loop. The simulated robot, lacking the gantry that supports the real robot during operation, falls during the open-loop window: in the gesture-loop recording, simulated body pitch reaches roughly −85° by end of clip while the real robot, supported by the gantry, settles within a few degrees of vertical.

This is not a policy failure. It is, however, the clearest single piece of evidence in our corpus that the gantry is doing balance work that the digital twin does not yet model: kinematics and signal paths transfer cleanly across the bridge; free-base dynamics and contact, when the policy is briefly removed from the loop, do not. We trim the open-loop window from the comparison statistics reported above; the trimmed and untrimmed numbers are both reported in the supplementary archive.

6. Validation and safety substrate

The two transfer bridges discussed in §4 and §5 are the substantive content of the deployment effort. The infrastructure that surrounds them — the cross-runtime parity comparators that gate every deployment, and the hardware-side safety envelope that guards every live run — is the part of the work that made debugging the bridges survivable on a non-trivially-expensive humanoid. This section describes that infrastructure.

6.1 Pre-deployment parity gates

A small number of comparators run before any live deployment. Each compares two views of the same observation pipeline. One compares the deployment-side observation builder against the trainer's, on a static initial state, with a tolerance appropriate to a quiescent system. A second compares the deployment-side observation against a reference dump captured at a known reset frame, with a tighter

tolerance suitable to the determinism of that frame. A third compares per-term observation envelopes between the two simulators on identical reset states. A fourth runs a heuristic check that the exported model produces sensible standstill behaviour and is not in some grossly broken state. Together these comparators triangulate the observation pipeline by parties that can fail independently — observation builder, exported model, simulator — and localise a regression to the responsible party.

Built around these comparators is a small grid of cross-runtime evaluation drivers spanning the two simulators and the two model formats (the PyTorch reference and the exported deployment model). The published action-level agreement between the reference and the exported model on a 150-tick rollout, after the corrections of §4, is below two-times-ten-to-the-minus-six radians; this is the comparator that surfaced the wrapper-boundary bug of §4.4 in the first place, and is the contract maintained on every subsequent export.

The framework-level limitation of these comparators is that all of them aggregate across vector positions. A bug that produces correct values in incorrect positions, of the form discussed in §4.3, leaves the comparator response unchanged because root-mean-square and maximum-element distances are invariant to permutation. The principled remedy, partially implemented and listed under §10 as a hardening priority, is element-wise positional comparison against frozen reference dumps captured on a known-good initial state, asserted at deployment startup before any policy publication.

6.2 Hardware-side safety gates

A preflight check runs immediately before every live deployment, between operator confirmation and the first policy publication. It enforces an envelope on per-joint pose, per-joint velocity, per-joint effort, inertial sensor stillness, body tilt from vertical, and gravity magnitude in the body frame. The numerical envelope values are calibrated for the X2's natural neutral pose and joint-group structure: legs and arms admit wider pose deviations than waist or head; velocities are constrained tightly because the preflight is run from rest; the inertial threshold is loose enough to accommodate gantry sway. A topology check confirms that all joint groups are publishing their state and that no third-party publisher is competing on the command bus.

A mode escalator service polls the vendor motion controller's actual reported mode at twenty hertz and uses the readback as ground truth for mode transitions. A C++ tilt watchdog runs at every control tick and forces a safe-hold pose if the body tilt exceeds a permissive threshold; the threshold is deliberately set wide enough to tolerate legitimate motion-induced leans during expressive gestures and narrow enough to fire before any plausible fall completes.

A final operational discipline ties these together: every live unconstrained run on the gantry-tethered robot is preceded by a gantry-bound dry-run on a static motion that publishes only standing-pose targets with stiffness and damping zeroed. The dry-run validates the entire stack — preflight, parity gates, finite-state handoff, mode escalator, watchdog, and recorder — without committing torque to the joints. Only on a clean dry-run does the operator launch the same wrapper without the dry-run flag for the live run. This procedure is the most-tested path in our deployment stack.

7. Experiments

The training methodology used for all results in this paper is reference-tracking reinforcement learning on the upstream Sonic-family architecture, with no changes to the network, reward function, or training loop relative to the upstream release. The simulator and motion-data pipelines are also unchanged at the algorithmic level; the modifications discussed in earlier sections are in observation construction, asset preparation, and post-training deployment.

Parallel simulation count. Throughput is set by the number of **independent Isaac simulator instances per GPU rank** (one rank per GPU in the eight-GPU layout). The **canonical eight-GPU H200 production run** reported in this paper used **16,384** instances per rank, i.e. **131,072** simulators in parallel across the node, with **24** simulation steps collected from each instance between policy updates — the throughput-oriented setting documented for eight-GPU H200 training alongside measured samples-per-second and memory headroom. A lower **4,096**-per-rank footprint (**32,768** total on eight GPUs) remains the conservative choice for smoke tests and memory-tight nodes when operators choose not to match the full production footprint.

Three evaluation suites are reported. The first is a multi-initialisation simulator benchmark spanning representative standing and walking motions with five initial body yaws each, with time-to-fall as the primary metric and a thirty-second cap appropriate to the post-correction policy. The second is a parity ladder of cross-runtime comparators along the lines described in §6.1, reporting agreement between the reference and exported policy formats and between the two simulators. The third is the matched simulator–hardware recording set described in §5.5, comprising three motions on the canonical checkpoint.

7.1 Training rounds, compute budget, and fine-tune gates

The canonical checkpoint used for all hardware demonstrations and matched simulator–hardware recordings was selected from the **second** of two training rounds, after the corrections of §4 had been applied to the asset chain and the observation pipeline. A first training round, on the original asset chain and prior to the observation-pipeline corrections, did not produce a deployable policy; we report comparative numbers from that round in §8 to quantify the cumulative impact of the corrections, but no hardware demonstration uses the round-one checkpoint.

Round-1 behaviour in MuJoCo. Interim MuJoCo evaluation during round-1 was acceptable near roughly **two thousand** environment iterations and **worsened** as training tracked the IsaacSim-matched distribution; late-round checkpoints **failed in MuJoCo** and were not candidates for hardware. The detailed ablation tables, parity methodology, and experimental inventory live in the repository's user-facing sim-to-sim documentation and appendix planning materials rather than in this PDF.

Fine-tune gates before round-2. Two controlled gates preceded the successful round-2 job: a **2k-iteration sphere-foot** fine-tune from the **16k mesh-foot** baseline after the trainer-side G18 foot-geometry correction, and a **4k sphere-foot** path that isolates the G20 rotation-order fix without retraining from scratch. Together they establish that the remaining sim-to-sim gap is decomposable into trainer distribution, observation layout, and export-adapter issues rather than a single opaque physics knob.

Headline compute. The successful canonical job is an **eight-GPU NVIDIA H200 SXM** node on Nebius, with experiment-tracking logs supporting an all-in headline of order **500 GPU-hours** (per-rank wall times summed across ranks land near **475 GPU-hours**). That headline is **not** merged with a separate round-1 **mesh-foot** attempt on the same topology that consumed on the order of **~300 GPU-hours** without a clean convergence story and that reinforced the URDF/MJCF foot-mismatch lesson, nor with an exploratory **B200** head-to-head slot that mainly confirmed that the bottleneck was training-distribution and observation contracts rather than raw FLOPs. Sonic's public G1 demonstrations are not directly comparable on published GPU-hour accounting; our position is simply that **non-G1** porting shifts wall time from "train longer" to "find the silent contract bugs first."

7.2 Compute substrate and provider choice

All large-scale jobs ran on **Nebius** bare-metal GPU instances. **H100**, **H200**, and **B200** eight-GPU presets are all subject to regional scarcity: cloud APIs may accept a reservation request even when capacity is exhausted, yielding long queues or instances that never become SSH-reachable. In practice we adopted a **parallel-provision** workflow — launch several region and SKU variants concurrently, keep the first healthy node, and tear down the rest — so that wall-clock time to *any* eight-GPU node did not dominate engineering time.

7.3 Training telemetry

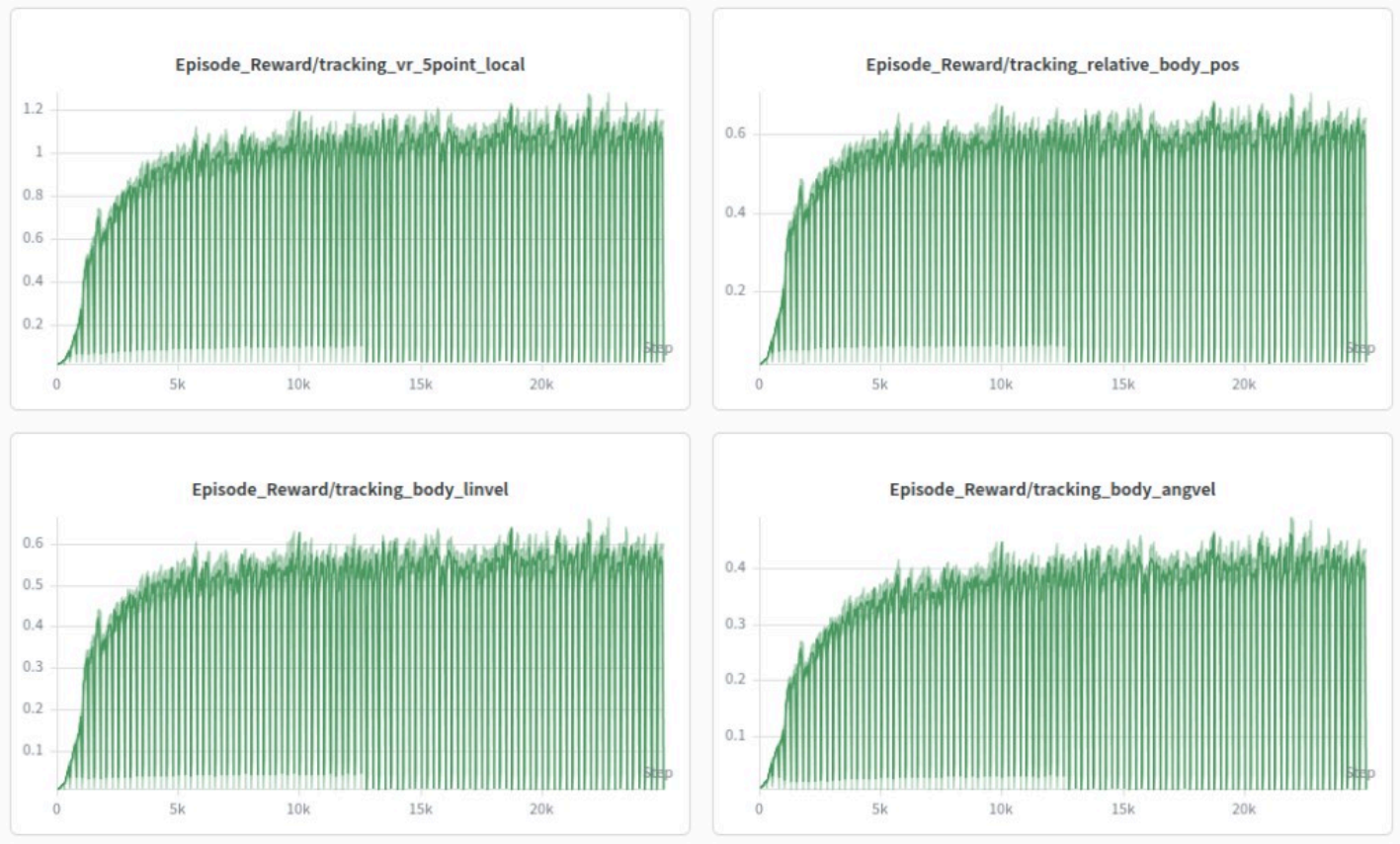

Figure 12. Per-episode reward components vs training step for the same job: local five-point tracking, relative body position, body linear velocity, and body angular velocity (four-panel training-dashboard snapshot).

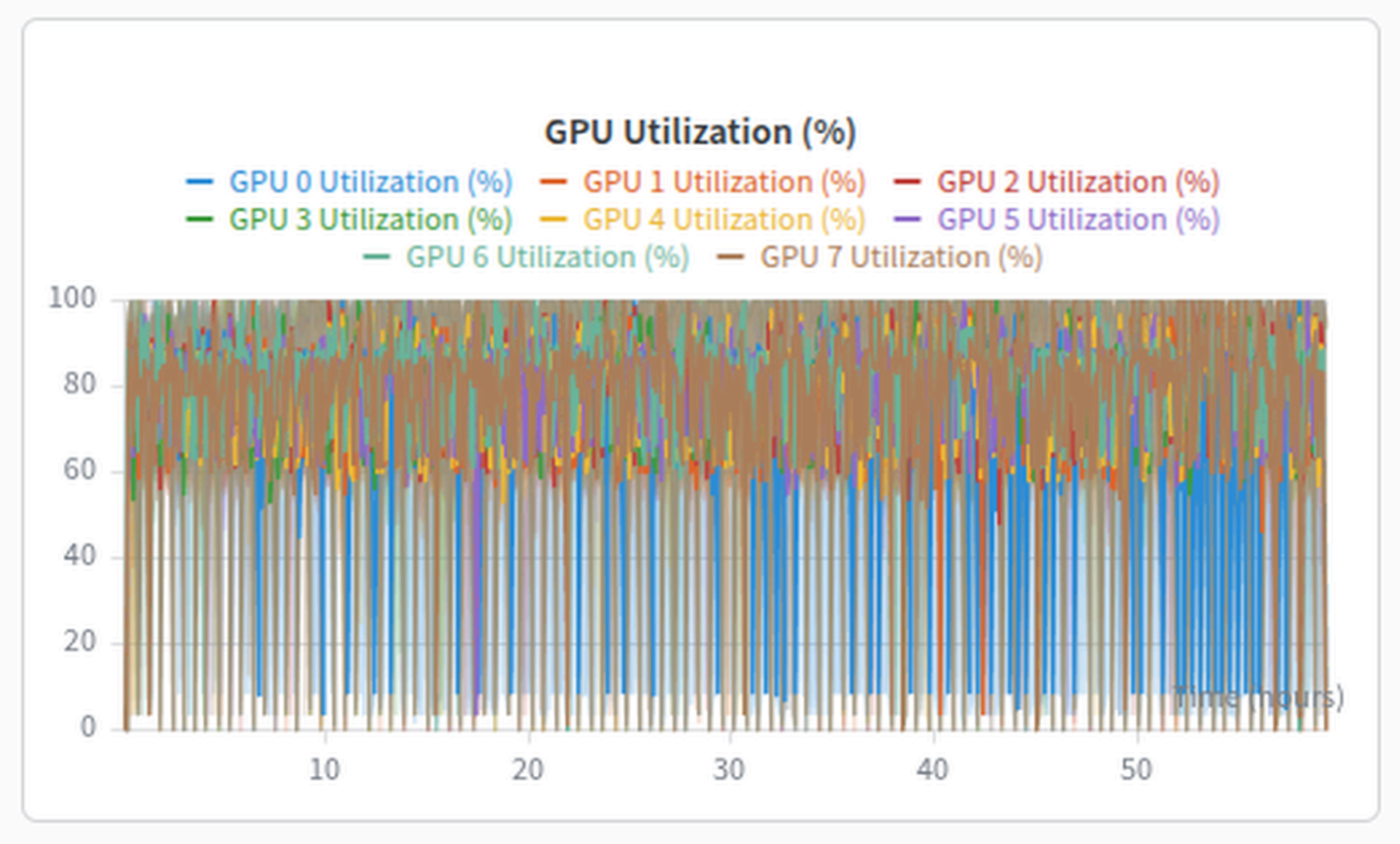

Figure 13. Per-GPU utilisation on the eight-GPU training node over wall time during the canonical round-2 job (training-dashboard snapshot).

Figures 12 and 13 are **dashboard snapshots** from the canonical round-2 training job: Figure 12 shows four per-episode reward components versus training step; Figure 13 shows per-GPU utilisation over wall clock on the eight-GPU node. They are included because they communicate convergence and load more clearly than heavily downsampled API exports; the quantitative headline in §7.1 is independent of the figure styling.

8. Results

8.1 Multi-initialisation simulator benchmark

Prior to the observation-pipeline corrections of §4, the round-one policy survived a mean of 2.43 seconds on the multi-initialisation benchmark before falling. Applying the heading-encoding correction alone, with no retraining, lifted the same checkpoint's mean time-to-fall to 5.48 seconds. The round-two policy, trained on the corrected asset chain and consumed through the corrected observation pipeline, saturated the thirty-second evaluation cap on every cell of the benchmark. Decomposed across the three corrections, the heading-encoding fix and the wrapper-boundary fix together account for the no-retrain improvement; the asset-chain audit and consequent retraining account for the cap-saturation. Both contributions are real, and earlier text that attributed the cap-saturation entirely to the asset-chain audit overstated its share. The retraction of the earlier attribution is recorded in §4.2.

8.2 Cross-runtime parity

The post-correction agreement between the PyTorch reference and the exported deployment model, measured as maximum action-level deviation over a 150-tick rollout, is below two-times-ten-to-the-minus-six radians. This is below the deviation threshold by approximately two orders of magnitude. Two diagnostic baselines are worth recording for context: prior to the wrapper-boundary correction the same comparison reported approximately three radians of disagreement, and an export produced before the correction that was subsequently re-validated against the post-correction reference disagreed at approximately four radians. The export-time constant-folding behaviour responsible for the second baseline is not fully eliminated by the correction; re-export from the same checkpoint is required after any deployment-side wrapper change.

8.3 Matched simulator–hardware recordings

Three matched recordings on the canonical checkpoint constitute the primary empirical content of the paper. Anchor C, ninety-six and a half seconds of upper-body gestural motion with the lower body holding the standing pose, exhibits an all-frame per-joint state-difference root-mean-square of 3.88° between simulator and hardware. Anchor D, forty-four seconds of turn-walk-turn-return locomotion, exhibits 5.03° on the same metric, with both simulator and hardware returning within three degrees of starting heading. Anchor B, fourteen and a half seconds of straight walking with a mid-clip turn-around, exhibits 7.27° all-frame and 4.74° once the open-loop exit-ramp window discussed in §5.6 is excluded. Disaggregated to individual joints (Figure 8), Anchor C holds nearly every joint within five degrees, Anchor D holds the same with the exception of two leg degrees of freedom near contact transitions, and Anchor B holds the

limbs but is dominated in its all-frame number by a base-heading divergence that is the subject of §9.1.

8.4 Calibrated fidelity claim

The matched recordings warrant a calibrated and explicitly bounded statement about the digital twin's accuracy. Confirmed by the recordings: the 31-DoF kinematic chain transfers to hardware within roughly five degrees per joint when the active control window is isolated; the torso inertial model is consistent with hardware over the gestural recording, where the upper body executes substantial motion while the lower body is held still; the position-control signal path tracks commanded joint targets to approximately a quarter of a degree of root-mean-square error on hardware, consistent with a correctly configured proportional-derivative model in the simulator; the inertial-sensor orientation channel matches at the precision the gestural recording demands; the reference-tracking objective transfers as intended.

Not validated, by the same recordings: the foot contact and friction model, against which Anchor B's heading-divergence signature is the most direct evidence; free-base dynamics independent of the gantry, against which the open-loop exit-ramp window of §5.6 is the most direct evidence; actuator saturation under high acceleration, since all three recordings operate within conservative target-clamp and low-pass-filter settings; and the noise spectrum of any onboard sensor, since the comparison is at the level of orientation and joint state rather than raw sensor signal.

This is the calibration any reader applying these results to a different humanoid platform requires. The kinematic and inertial part of the digital twin transfers; the contact, free-base, saturation, and sensor-noise parts must be re-validated on the target platform before any claim about untethered operation can be made.

9. Discussion and limitations

9.1 The base-heading divergence on Anchor B

On the walking-with-turn recording, simulated and real base-yaw agree for roughly the first second and then diverge to a 166° end-of-clip gap, while per-cycle joint kinematics agree to within five degrees throughout. The simulated base is free, the hardware base is gantry-pinned, so this divergence is equally consistent with contact, friction, or free-base integration error in the digital twin and with the boundary-condition asymmetry of the experiment. We do not investigate the question further here; the matched-boundary passive sway test of §10 would isolate the physics hypothesis from the boundary-condition asymmetry.

9.2 A failure-mode taxonomy

Four non-overlapping classes structure diagnosis: **bugs** (wrong values in wrong positions; fix with element-wise frozen-frame parity), **physics-divergence** (mis-modelled contact, friction, dynamics, noise; fix with policy-free bench tests), **handoff failures** (service codes vs ground-truth mode, audible whir; fix with §5 protocol), and **tuning** (filters, clamps; fix with single-knob sweeps). Tighter tuning can mask the other three without fixing root cause — classifying the symptom is most of the work.

9.3 Scoping

One gantry-tethered robot, one retargeter fork, one canonical checkpoint, three matched recordings. Methodology in §4 and §6 is intended to port; §8.4 numbers are platform-specific until re-measured.

10. Conclusion and future work

This deployment was three sim-to-sim observation corrections, a finite-state sim-to-real handoff on a closed motion-controller API, and a calibrated digital twin grounded in three matched recordings. Failures were software, encoding, handoff, or wire-level — not opaque physics tuning.

The defensible claim is §8.4's fidelity statement: kinematic chain and torso IMU within roughly five degrees / five percent where the archive is evidence. Foot contact, friction, free-base dynamics, saturation, and sensor noise remain explicitly out of scope; Anchor B's 166° yaw gap is *consistent with* twin vs gantry asymmetry or contact error, not a resolved root cause.

The highest-leverage follow-on is a small battery of **policy-free bench tests** (passive sway, open-loop joint sweep, hanging-arm gravity, foot-drop, per-joint step response) that turn indirect anchor evidence into isolated component validation. Broader sweeps — domain randomisation, richer hardware motions, overlapped MC handoff — are natural next steps once those benches exist.

References

1. AgiBot. *X2 AIMDK — Get SDK*. Documentation portal. https://x2-aimdk.agibot.com/en/latest/get_sdk/index.html
2. Bones Studio. *BONES-SEED* (human motion dataset, 142,220 clips). Hugging Face. https://huggingface.co/datasets/bones-studio/seed
3. NVIDIA. *SOMA retargeter* (inverse-kinematics humanoid motion retargeting). GitHub repository. https://github.com/NVIDIA/soma-retargeter