ORBITUS ROBOTICS / WHITE PAPER

Continuous arm retargeting without competing trajectory workers

A reproducible controller-handoff case study from ROB

OR-WP-2026-001 | Version 1.1 | November 4, 2026

Orbitus Robotics · Project lead: Rodolfo Aramayo

Abstract

Changing an arm target while it is moving requires a consistent motion state and a single owner of the output. Inspection of ROB's installed Amber controller identified detached trajectory workers, separately sampled derivatives and a cancellation-to-publication race. An isolated numerical example reproduced a large backward excursion from a deliberately inconsistent seed; the historical incident state was not recorded. We describe a replacement owner that retains commanded position, velocity and acceleration, replaces pending targets in one mailbox, and serializes Stop with publication. Recorded offline tests cover 4,200 targets, 3,898 moving retargets and 19 regression groups in three build configurations. The contribution is a source-bound diagnosis, an offline-tested ownership design and a commissioning boundary. These results do not establish real-arm tracking, stopping performance or collision safety.



RESEARCH RECORD

How a detached-worker handoff can corrupt motion continuity, and how a persistent owner was tested before live integration.

Evidence originates in the 2026-11-01 report. Technical white paper · Not peer reviewed.

ROB at the table during the October 29, 2026 can session. Still from Kierie's video. This earlier physical scene provides application context; it does not show the replacement controller operating.

Keywords: motion control, trajectory generation, retargeting, Ruckig, robot reliability

ROB: history, purpose and software

ROB (R.O.B.) is Orbitus Robotics' experimental tracked humanoid: a platform for robotics education, human-robot interaction and supervised manipulation. Independent treads, a rotating torso, two seven-joint arms and grippers, cameras and range sensing connect software to a physical machine. Can-handling practice, stage demonstrations and spatial teleoperation give the project concrete tasks to investigate. [5,6,7]

The project's archive reconstruction records ROB-specific Mac serial-control work in 2017-2018 and a surviving Cerebro Git root on January 1, 2018. Later histories add speech, perception experiments and controller arbitration. Public galleries document builds and appearances from 2019 onward. The fresh Cerebro repository of August 5, 2025 imported an existing v5 application; it was not the project's beginning. The pre-v5 originals remain local, with provenance summarized in the supporting record. [5,7]

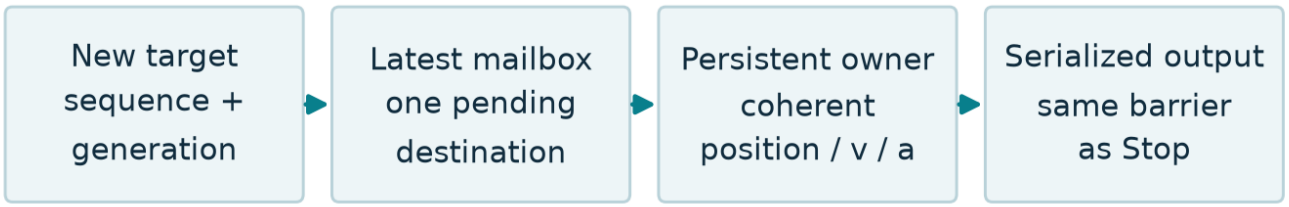
Codebase / platform	Role in ROB
Cerebro / macOS	Swift, Objective-C/Objective-C++, C++ and Python helpers coordinate perception, dialogue, command admission and robot I/O.
ROBController / iPhone, iPad, Watch	Human command, maps, telemetry and explicit action approval.
ROBControllerVision / visionOS	SwiftUI spatial controls, live video and time-limited arm authority.
ROBArduino + RPLidar project	Base firmware handles low-level drive; the lidar application supplies scans and map observations.
Amber host + ContinuousArm	Existing Ubuntu arm runtime; the C++ replacement owner and codec live in Cerebro/Tools/ContinuousArm.

The public source index links six repositories and suggests starting files. The architecture separates human intent and AI-assisted planning from motor-facing ownership. This paper examines that latter boundary. It does not establish that every project feature is active together, or that the replacement controller has been physically commissioned. [5,6]

1. The problem: a new target must not create a new owner

ROB combines a tracked base, a rotating torso and two seven-joint arms. Can handling and headset control both need a moving arm to accept a revised destination. A command acknowledgement only shows that a message exchange finished. It does not establish that the previous trajectory completed or that its worker lost permission to publish.

The inspected gateway's UDP command 4 path acknowledges before motion completion. Static analysis of the installed controller shows a position-task handler creating a detached trajectory worker. Replacement planning combines commanded position with velocity and acceleration sampled separately. An older worker can check its cancellation condition and then publish after a replacement has been accepted. The check and write do not form one ownership transaction. [1,2]



Design proposal: one owner of commanded motion, including replacement and Stop.

Figure 1. Proposed ownership boundary. New targets replace one mailbox value; only one persistent owner can publish. This is a design diagram, not a timing trace.

These observations establish a concurrency hazard and a plausible mechanism for discontinuity. They do not recover the exact state during the earlier physical rotation. Joint-reference errors, sign conventions and collision-model errors are separate possibilities that still need their own evidence.

> Application-level waiting can reduce exposure to this handoff hazard. Removing a completion guard before replacing the motor-facing owner would remove that containment without demonstrating a repaired controller.

2. Method and numerical counterexample

The investigation compared three initial states using the installed Ruckig 0.9.2 baseline, with J6 starting at +0.34 rad and a target of +1.05 rad. The launch constraints observed after unit conversion were 5,400/4,200 deg/s, 90,000 deg/s² and 5,000 deg/s³. These were software configuration values, not established actuator ratings. A minimum requested duration is also not a substitute for a speed limit. [1,2,4]

Initial state	Minimum calculated position	Interpretation
v = +0.2 rad/s; a = 0	+0.340000 rad	Coherent moving state
v = 0; a = -40 rad/s²	-5.262596 rad	Constructed inconsistent state
v = 0; a = 0	+0.340000 rad	Stationary handoff

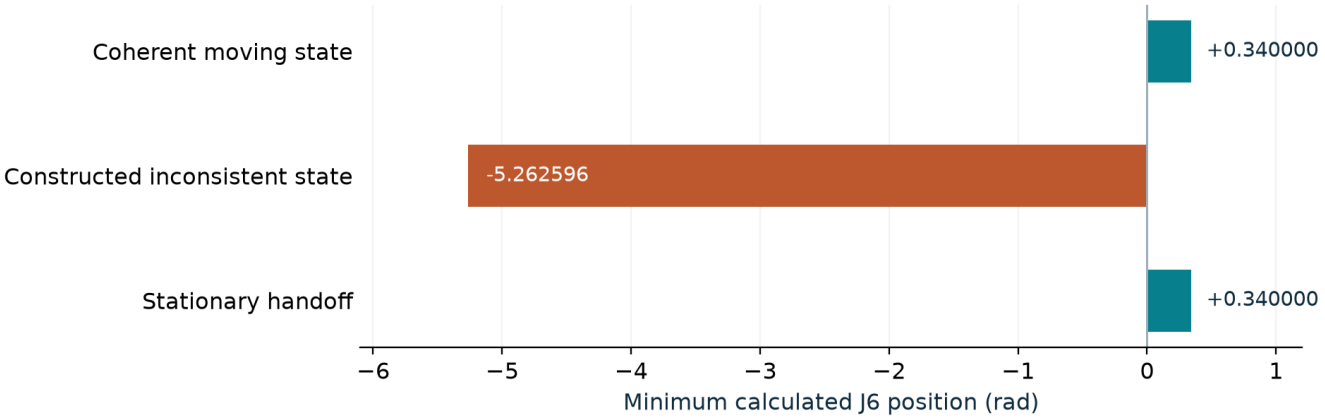


Figure 2. Minimum calculated J6 position in the three recorded numerical cases. Values are from the source-bound release record [2]. They are simulated extrema, not measured robot angles.

The hypothetical -40 rad/s² value corresponds to a 0.2 rad/s speed sample dropping to zero across 5 ms. It was deliberately chosen to expose the effect of an inconsistent derivative. It is not an estimate of the historical seed, a claimed physical -5.26 rad rotation, or a measured failure frequency.

This comparison isolates state handoff from geometry. Direct joint command 4 bypasses the Drake/URDF solver. Editing meshes or adding a table to a viewer cannot serialize trajectory workers in that path.

3. Replacement owner and recorded validation

The replacement keeps the last successfully published commanded position, velocity and acceleration together. A new target changes the destination while preserving that coherent state. A single latest-target slot bounds backlog. Generation and sequence checks reject stale or foreign commands, and publication shares the same ownership barrier as Stop. Measured feedback remains a separate observation rather than being silently substituted for commanded derivatives. [1,2]

Recorded test	Result	What it establishes
Main seven-joint replay	14,000 samples over 70 simulated seconds; 4,200 targets; 3,898 moving retargets	Continuity under this synthetic input
Illustrative dynamic bounds	$v \le 0.3$ rad/s; $a \le 0.6$ rad/s²; discrete jerk ≤ 2 rad/s³; 5 ms step ≤ 0.0015 rad	Configured simulation bounds
Command rates and bursts	20/60/90/120 Hz pose input; 1,000-target burst gives one plan; 100,000 concurrent submissions retain one slot	Bounded target ownership and backlog
Regression configurations	19 groups pass in Release, ASan/UBSan and TSan	Recorded functional and instrumented coverage

All joints converged to the final 0.1 rad target in the main replay. Feedback was idealized from the previous output unless a test injected a fault. The replay therefore does not measure actuator lag, friction, backlash, current, bus latency or collisions. The publication reuses recorded results; it does not count them as new physical tests.

Failure cases cover Stop/publication races, sequence replay, foreign generations, stale tracking or feedback, loss of authority, timing gaps, blocked braking paths and uncertain output. An ambiguous write latches a fault rather than retrying a sample that might already have been accepted.

4. Bounds, integration and limitations

Checking trajectory samples alone can miss an extremum between samples. The project's checker integrates initialized polynomial segments and evaluates the relevant roots, braking phases and unequal-duration tails. Testing included 526,750 joint-state observations from 250 randomized trajectories, plus an exact zero-jerk, nonzero-acceleration interior-maximum case. This addressed two identified assumptions in the pinned 0.9.2 profile helpers; it is not a claim about every later Ruckig release. [1,2,4]

The model audit found each vendor arm description contained eight links: a base and one seven-joint chain. It omitted the other arm, torso, head, jaws, payload, table and floor. Its joint speed/effort values were placeholders. The separate Cerebro preview had 45 links and 44 joints, but its listed speed/effort pairs were zero. Neither model supplied a commissioned whole-robot dynamic envelope. [3]

Integration requirement	Evidence still needed
Exclusive output	Replace the legacy writer; do not stream 200 Hz targets into command 4 or run two motor writers.
Units and references	Verify physical RIGHT = L-10/CAN10 and LEFT = R-11/CAN11; establish signs, zeros, limits and tool transforms.
Stopping and feedback	Measure tracking, transport age, current, output deadlines and independent stop behavior.
Headset and world model	Calibrate pose deltas and whole-body collision geometry; validate tracking-loss braking before continuous VR control.

A valid future headset path would convert a fresh clutched pose into a calibrated position/quaternion delta, use continuous IK seeded from the current motion state, check the full robot/world and stopping path, then submit the latest joint target. A fresh generation is needed after release, reconnect or fault. No such physical commissioning is established by the November 1 results.

> Conclusion: preserving motion continuity is an ownership problem as well as a trajectory-math problem. The offline owner removes the demonstrated software failure class within its tested model; a physical driver and measured safety envelope remain separate deliverables.

5. Reproducibility and publication record

The November 1 implementation and test records are pinned to project commit f4f569ba979df443a37c553d266533a5e64e870e. The Ruckig baseline is 7de6b1035e1cad509251b31b3287d10fbcda6baf. The linked README describes the isolated build and regression procedure. A machine-readable evidence summary accompanies this paper, preserving the distinction between hardware_io=false, simulated output and physical observations.

This paper adapts the five-page arm-transition report into a public case study. Personal session messages, local machine paths and live process IDs are omitted. The photograph is historical context and is explicitly dated. The companion controller paper examines the host boundary; the recovery paper records subsequent staging and the remaining physical takeover gates.

> Research questions for the next edition: What stop latency and travel occur under actual process/bus failure? Which tracking limits are supportable by each joint? Can measured full-body clearance be maintained during moving retargets? Those answers require new, controlled physical evidence.

References and supporting records

[1] Continuous owner design and reproduction instructions (f4f569b)

https://github.com/RudyAramayo/Cerebro/tree/f4f569ba979df443a37c553d266533a5e64e870e/Tools/ContinuousArm

[2] Release, sanitizer and numerical records (November 1)

https://github.com/RudyAramayo/Cerebro/tree/f4f569ba979df443a37c553d266533a5e64e870e/docs/calibration/evidence/2026-11-01-continuous-arm

[3] Controller and model investigation (f4f569b)

https://github.com/RudyAramayo/Cerebro/blob/f4f569ba979df443a37c553d266533a5e64e870e/docs/calibration/continuous-arm-retargeting-2026-11-01.md

[4] Pinned Ruckig profile implementation (0.9.2 baseline)

https://github.com/pantor/ruckig/blob/7de6b1035e1cad509251b31b3287d10fbcda6baf/include/ruckig/profile.hpp

[5] ROB project chronology and public source index (v1.1)

https://www.orbitusrobotics.com/whitepapers/rob-project-background.json

[6] ROB software platform and operator interfaces

https://www.orbitusrobotics.com/products/

[7] ROB development and Maker Faire photo archive

https://www.orbitusrobotics.com/makerfaire/

Web edition and supplements: https://www.orbitusrobotics.com/whitepapers/arm-transition-fix-2026-11-01/

Version 1.1 adds ROB history, purpose, functionality and a software-stack overview. Recorded test results and physical commissioning status are unchanged.

Prepared with AI assistance from project reports, source code and recorded test evidence. Technical claims are tied to the cited records. No new physical robot trial was performed for this publication.

Photography: Kierie, October 29, 2026. Diagrams: Orbitus Robotics. Correspondence: orbitus@orbitusrobotics.com