



Eric J. Goossen

Computer engineering student at the University of Washington focused on embodied intelligence, where machine learning meets the real world.

ericjgoossen.com · github.com/EricJGoossen · linkedin.com/in/ericjgoossen · ericjgoossen@gmail.com

I'm motivated by turning abstract math and computer science into reliable, real-world systems with measurable impact.

This portfolio is meant to be viewed at ericjgoossen.com

This PDF is only a short summary. The website has every project's full write-up, with videos, code links and more projects. Please read it there.

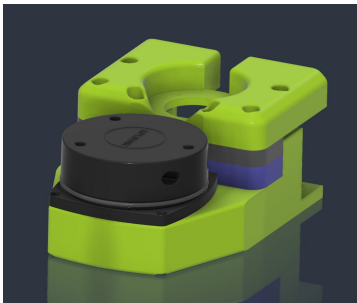
OpenDubs

Personal Robotics Lab · Oct 2025 – present

ericjgoossen.com/projects/opendubs

14,669 lines of code I wrote

A low-cost bimanual mobile manipulator for the Personal Robotics Lab, made for other labs to build on, from drive and localization bring-up to two arms planned and run as one.



Two 2D lidars, merged into one scan

2026-04 · Language: Python · Role: Team

A node that merges two 2D lidars into one scan for slam_toolbox and Nav2, which drove the robot point to point in simulation.

[Full entry, with video and code →](#)



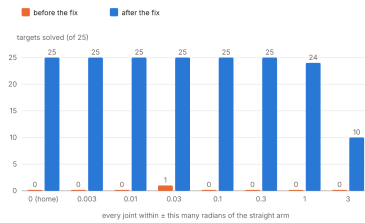
Arm control: three impedance controllers, then a simpler one

2026-07 · 3,525 lines of code I wrote · Language: C++ · Role: Solo

Three impedance controllers for the OpenArms, and why the robot runs the motors' own loop with 500 Hz gravity compensation instead.

Each arm is a 7-DOF OpenArm whose Damiao motors run their own impedance loop: the Thor sends position, velocity and torque targets with gains over CAN-FD, and each motor closes the loop itself. I measured the stock ROS 2 stack first.

[Full entry, with video and code →](#)



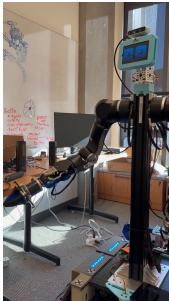
Fixing the IK solver for the OpenArm

2026-07 · 34 lines of code I wrote · Language: Python · Role: Solo

I traced the lab's IK solver failing on the OpenArm to a singularity and fixed it: 184 of 200 targets solved instead of 1.

Teleop and planning both lean on ssik, the lab's analytical IK solver, which I didn't write, and on the OpenArm it kept failing.

[Full entry, with video and code →](#)

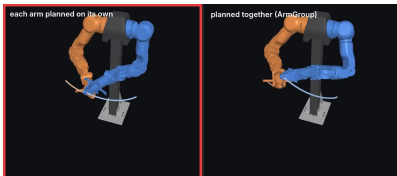


Teleop with a Meta Quest, in the browser

2026-07 · 5,072 lines of code I wrote · Language: Python, JavaScript · Role: Solo

Teleop for both arms with a Meta Quest, built from scratch as a web page the headset opens in AR.

[Full entry, with video and code →](#)



N arms, one planner: ArmGroup

2026-08 · 3,200 lines of code I wrote · Language: Python · Role: Solo

ArmGroup plans every arm's joints together in the lab's planner, so two arms on one torso never collide; it plans both arms on the robot.

The lab plans with its own MuJoCo planning library, mj_manipulator, which planned one arm at a time. With two arms on one torso, two plans that are each safe can still hit each other. I added an ArmGroup to it that plans all arms' joints together, so every step is checked arm against arm.

[Full entry, with video and code →](#)

Benchmarking VLAs for ARCHITECT

Human Centered Robotics Lab · Apr 2026 – present

ericgoossen.com/projects/hil-cap

[2,031 lines of code I wrote](#)

The real-robot benchmark behind a paper on language-guided robot code, from the evaluation harness to what four VLAs can and can't do.

	NO VLA	NO VLA	MolmoAc22 VLA	GR00T M17 VLA	Code as Policies LLM code	ARCHITECT LLM code
Banana → plate	0.0	0.0	0.0	0.0	0.0	0.0
Block → basket	0.0	0.0	0.0	0.0	0.0	0.0
Baseball → basket	0.0	0.0	0.0	0.0	0.0	0.0
Apple, bread → bowl	0.0	0.0	0.0	0.0	0.0	0.0
Bread from box	0.0	0.0	0.0	0.0	0.0	0.0
Close drawer	0.0	0.0	0.0	0.0	0.0	0.0
Fold cloth	0.0	0.0	0.0	0.0	0.0	0.0
Banana under cloth	0.0	0.0	0.0	0.0	0.0	0.0

Success rate over 10 rollouts per task, the objects' placement and the wording change every rollout.

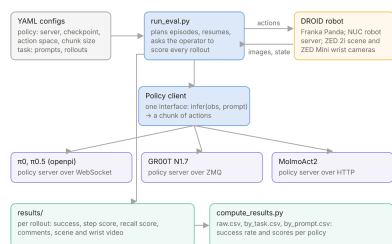
Designing the benchmark: what makes a VLA test fair

2026-05 · 178 lines of code I wrote · Role: Team

How we made the VLA benchmark fair: eight tasks, randomized placements, three scores per rollout, and where the models failed.

A VLA can look great near its training setup and fall apart a few centimeters away, so the hard part is deciding what a fair test is. We picked eight tasks across six kinds of challenge and changed the objects' placement and the wording on each of ten rollouts.

[Full entry, with video and code →](#)



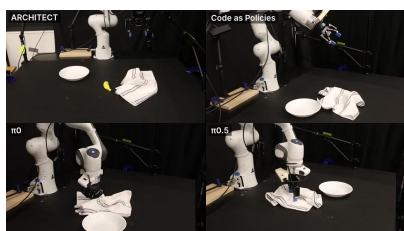
An evaluation harness for any VLA

2026-07 · 1,853 lines of code I wrote · Language: Python · Role: Solo

One open-source harness that runs any VLA on a Franka through DROID, whatever protocol its server speaks.

Every VLA ships its own server, and they don't agree on how to talk: openpi's $\pi 0$ and $\pi 0.5$ use WebSockets, GR00T uses ZMQ and MolmoAct2 uses HTTP. I wrote one harness that hides all of that behind a single interface: a new model, task or whole benchmark is a short YAML file, not new code.

[Full entry, with video and code →](#)



The paper: A Few Words Go a Long Way

2026-07 · Role: Co-author

The paper my benchmark fed: ARCHITECT, which fixes LLM-written robot code with a few words from a person, succeeded 70 to 100% of the time on all eight tasks.

ARCHITECT is the lab's system, not mine: I'm a co-author for the VLA benchmark behind its comparison. The paper asks whether robot programs written by a coding agent, and fixed with a few words from a person, can beat end-to-end VLAs.

[Full entry, with video and code →](#)

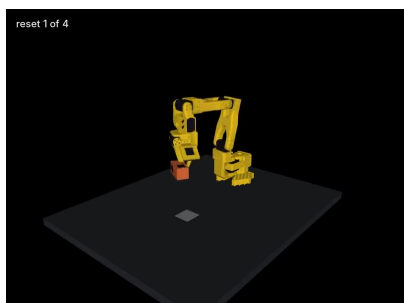
Fleet RL

WEIRD Lab · Sep 2026 – present

ericgoossen.com/projects/fleet-rl

7,208 lines of code I wrote

Twelve low-cost robot arms learning one task together, with resets that need no human.



Resetting the peg without a person

2026-10 · 7,208 lines of code I wrote · Language: Python · Role: Lead

A reset that finds the peg with AprilTags, plans a pick and place, and puts it back, so the arms can run online RL without a person.

In online RL an arm practises a task over and over, so after every try something has to put the peg back where the task starts. I built that reset for the peg-insertion task.

[Full entry, with video and code →](#)

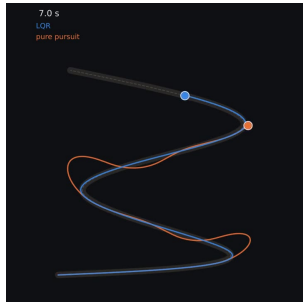
Advanced Robotics at UW

Advanced Robotics at UW · Oct 2025 – present

ericgoossen.com/projects/aruw

10,204 lines of code I wrote

Localization and control for ARUW's competition robots, from Kalman-filter odometry and LQR path tracking to measuring the sentry's localization against motion capture.



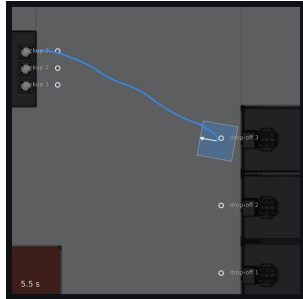
Path tracking: from pure pursuit to LQR

2026-05 · 1,880 lines of code I wrote · Language: C++ · Role: Lead

I led replacing the chassis's pure pursuit with LQR path tracking that models the robot's accelerations: within 3.2 cm of the path in simulation.

Our chassis followed paths with pure pursuit: aim at a point a fixed distance down the path and ramp the speed up toward it.

[Full entry, with video and code →](#)

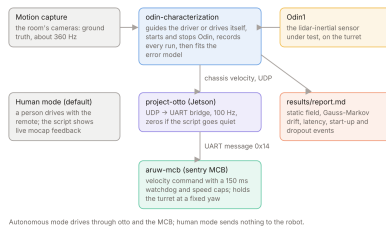


Kalman-filter odometry for three dead wheels

2026-06 · 905 lines of code I wrote · Language: C++, Python · Role: Solo

The engineer robot's Kalman-filter odometry rewritten for three dead wheels, split into three small filters and tuned offline from logged drives.

[Full entry, with video and code →](#)



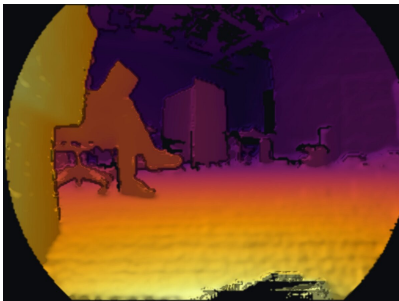
A test platform against motion capture

2026-10 · 5,959 lines of code I wrote · Language: Python · Role: Solo

A test platform that measures the sentry's lidar-inertial localization against motion capture, from guided driving to a fixed campaign protocol.

The sentry localizes with an Odin1, a lidar-inertial sensor that relocalizes into a prebuilt map. Before the team decided whether the whole localization system could rely on it alone, we wanted to know how accurate it is and where it is weak.

[Full entry, with video and code →](#)



An error model for the Odin

2026-10 · 691 lines of code I wrote · Language: Python · Role: Solo

An error model for the Odin, validated on a held-out session by Monte Carlo, that showed the sentry could rely on it alone.

The question wasn't how accurate the Odin is but whether the sentry could rely on it alone, and one accuracy number can't answer that: the nav stack needs to know how the error behaves, whether it grows, and what happens when something goes wrong.

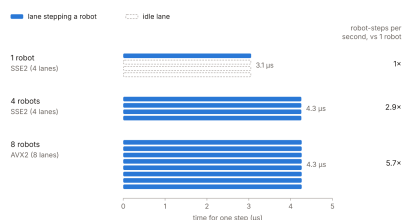
[Full entry, with video and code →](#)

Achilles

Orchid Robotics · 2026 – present
ericgoossen.com/projects/achilles

43,311 lines of code I wrote

A C++20 rigid-body simulator that steps 8 robots per SIMD instruction, about 3× MuJoCo's single-threaded throughput.



SIMD: eight robots per instruction

2026-09 · 6,535 lines of code I wrote · Language: C++20 · Role: Solo

Structure-of-arrays storage sorted by tree depth, so one SIMD instruction steps eight robots: 5.7× the throughput for 1.4× the time.

V3 replaces V2's Eigen with my own spatial algebra on xsimd, and stores every field as a structure of arrays: each component of each field, say the x angular velocity of every joint, is its own contiguous, aligned array.

[Full entry, with video and code →](#)



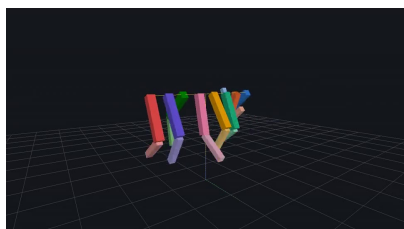
Dynamics: the Articulated Body Algorithm

2026-09 · 4,217 lines of code I wrote · Language: C++20 · Role: Solo

The Articulated Body Algorithm in $O(n)$, checked against MuJoCo and against a double pendulum's exact equations of motion.

V2's dynamics were wrong in ways only a reference could show: brought back to life next to MuJoCo, a one-link arm swung exactly 4× too fast (it used the free-body inverse inertia, dropping the parallel-axis term), and a two-link arm moved in the wrong shape.

[Full entry, with video and code →](#)



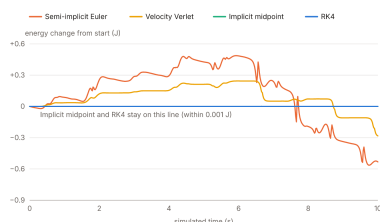
A viewer drawn from the simulation's own state

2026-09 · 1,702 lines of code I wrote · Language: C++20 · Role: Solo

A small OpenGL viewer that draws every body straight from the simulation's state, one cube and one draw call per body.

The viewer is a small OpenGL 3.3 program with GLFW as its only dependency. GLFW opens the window and also loads the handful of OpenGL functions the viewer needs.

[Full entry, with video and code →](#)



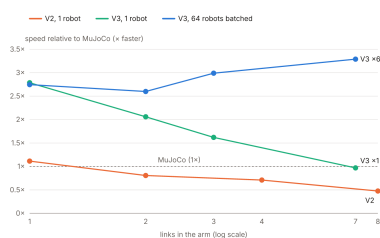
Integrators that hold energy

2026-10 · 1,032 lines of code I wrote · Language: C++20 · Role: Solo

Velocity Verlet and implicit midpoint replace drifting semi-implicit Euler, cutting the double pendulum's energy error by more than half.

With the accelerations right, longer runs exposed the next problem: semi-implicit Euler drifts. A frictionless double pendulum should conserve energy, yet over 10 seconds at a 2 ms step Euler's total energy wanders by up to 0.56 J.

[Full entry, with video and code →](#)



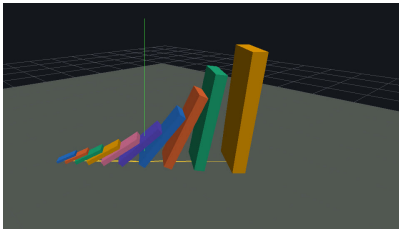
Beating MuJoCo on throughput

2026-10 · 6,022 lines of code I wrote · Language: C++20, Python · Role: Solo

Benchmarked against MuJoCo on the same scenes: batched across SIMD lanes, Achilles steps a seven-link arm 3.3× faster.

To see where V3 stood, I benchmarked it against MuJoCo 3.15 on the same arm scenes, converted from the same .arrow files, at the same 2 ms step, single-threaded on my laptop (Intel Core Ultra 9 285H).

[Full entry, with video and code →](#)



Collision response

2026-10 · 2,041 lines of code I wrote · Language: C++20 · Role: Solo

A soft-contact solver with Coulomb friction that turns contacts into forces, run on a domino scene next to MuJoCo.

Detecting contacts is only half the battle; working out what happens next is the rest. Once accelerations and contacts are known, the solver adjusts the accelerations so bodies respect their contacts.

[Full entry, with video and code →](#)

Swerve Robotics in Botswana

Swerve Robotics · July 2025

ericgoossen.com/projects/botswana

Two weeks in Botswana with Swerve Robotics and Dare to Dream, advocating for STEM funding at a U.S. Embassy event and introducing FIRST LEGO League to students.



Robotics with students in Gaborone

2025-07 · Language: Python · Role: Teacher

A FIRST LEGO League exchange with more than 50 students at a school in Gaborone, where I taught the robot code I'd written.

With Dare to Dream we ran a cultural exchange at Mogoditshane Senior Secondary School in Gaborone for more than 50 students, some visiting from Mogoditshane Junior Secondary.

[Full entry, with video and code →](#)



Advocating for STEM at the U.S. Embassy

2025-07 · Role: Speaker

Advocating to ministers and business leaders for STEM funding and lower tariffs at the U.S. Embassy, and giving the Introduction to FIRST speech.

The trip's main event was Culture Meets Code, which Dare to Dream hosted with Swerve Robotics and the U.S. Embassy in Gaborone.

[Full entry, with video and code →](#)

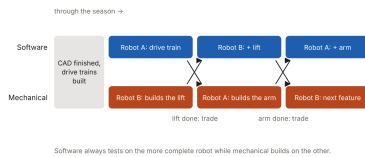
FIRST Tech Challenge

417 SKID, Swerve Robotics · May 2023 – May 2025

ericgoossen.com/projects/ftc

6,090 lines of code I wrote

On FTC team 417 SKID, captain in 2024–25, I built Live View, a rules-legal camera feed for the drivers, and SKIDADDLE, a path follower that plans with physics.



Two robots, so software never waits

2024-09 · Role: Captain

As captain, I had the team build two robots in parallel and trade them at each finished feature, so software always had a robot to test on.

As captain I set up a new way to build our robot: two robots developed in parallel. While the CAD was being finished, we assembled the drive trains. Software then got the first robot, a bare drive train, to test on, while mechanical integrated the first feature, such as the lift, on the second.

[Full entry, with video and code →](#)



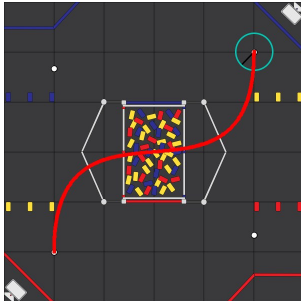
Live View: camera video sent as text

2024-12 · 307 lines of code I wrote · Language: Java · Role: Solo

Camera video sent to the driver station as Braille-character text, within FTC's rules, so drivers could see what the robot saw.

FTC rules ban streaming camera images to the driver station during a match, but telemetry text is allowed, so Live View sends the robot's camera feed as text.

[Full entry, with video and code →](#)



SKIDADDLE: a path follower that plans with physics

2025-10 · 3,091 lines of code I wrote · Language: Java · Role: Solo

A Java path-following library for FTC that plans with a physics model of the robot, with math I derived by hand.

SKIDADDLE is a path-following library for FTC robots, written in Java, that I built to fix inefficiencies in Road Runner, the usual FTC path library, and get better paths.

[Full entry, with video and code →](#)

Publications

A Few Words Go a Long Way: Language Guided Robot Policy Synthesis

D. Chen, A. R. Jain, E. Goossen, E. Romig, M. Murray, N. Walker, M. Cakmak

In preparation for IEEE RA-L · 2026

Tests whether human feedback helps LLM-written robot code beat vision-language-action models: human feedback lifts Code-as-Policies 1.75× over base VLAs, solving tasks VLAs failed completely.

[arXiv](#) · [Website](#)