

Hoverhand: a simulation study of a small humanoid that flies on ducted fans in its feet and uses its hands in the air

castor639

Independent project, October 2026

Every result in this report comes from simulation. No part of the robot has been built, and nothing has flown.
The report describes a design, a simulator, controllers trained and tested in that simulator, and a plan to build the robot.

Abstract

We describe the design and simulation of a 1.47 kg, 30 cm humanoid robot that flies on two 102 mm ducted fans built into its feet and two 127 mm ducted rotors on its back, and carries a one-servo gripper on each forearm. The base is Rhoban’s open-source 30 cm humanoid. We wrote a simulator around the base robot’s MuJoCo model that adds brushless motor and battery electrics, ESC delay, ground effect, gyroscopic torques, servo torque limits, and sensor noise and latency. A hand-written controller (position loop, SO(3) attitude loop, centre-of-mass-aware thrust allocation, arm angular-momentum feedforward) holds position within 1.5 cm while the arms reach and swing, and a version of it flies a four-waypoint 3D route from a standing start and lands. We trained flight control with PPO in two ways. An end-to-end hover policy flies, but drifts up to 9.2 cm in the full simulator against 1.5 cm for the hand-written controller. A residual policy that adds up to $\pm 25\%$ of hover thrust per rotor keeps the robot flying through a 20% thrust loss on one foot fan; without it, the hand-written controller loses control 5.5 s after the fault. On healthy rotors the residual policy tracks worse (5.2 against 4.4 cm RMS) and lands closer to its target (1.1 against 19 cm). Each of these comparisons is a single simulated run, with policy inference on the CPU so that it repeats exactly. The foot fans hover at 77% of their modelled maximum thrust, and the design does not survive a 30% loss on a foot fan. We give an itemised bill of materials (\$4,554 before tax and shipping, including tools).

1 Introduction

A robot that walks into a room, flies up to a shelf and takes something off it needs legs, hands and thrust in one body. Most robots that handle objects in the air are multirotors with an arm mounted under them [26, 30]. A few humanoids have flown. IIT lifted its 70 kg iRonCub3 on four jet turbines in 2025, hung from a crane and tethered [12, 15]. Jet-HR2, a 17 kg biped from Guangdong University of Technology, took off on ducted fans at its waist and feet [20]. A 3.5 kg humanoid from the University of Tokyo flew on a three-fan backpack and used its arms for aerial manipulation [34].

This project asks whether a small humanoid can carry its lift in its feet, keep its legs free, and use its hands in the air. We start from Rhoban’s open-source humanoid, about 30 cm tall with 19 Dynamixel XL330 servos [22]. Each foot becomes a 102 mm ducted fan printed around the base robot’s ankle bracket. Two 127 mm ducted rotors sit behind the chest, placed so that their wake misses the feet, and each forearm gets a gripper driven by one more XL330 (Fig. 1). A humanoid is an awkward airframe: when an arm swings, the centre of mass moves, the inertia changes, and the reaction torque lands on the body, and with the thrust in the feet the main lift acts at the end of two six-servo chains. At 1.5 kg a crash costs props and a servo, which makes this a cheap scale at which to test the idea.

Everything reported here is simulation. Within that limit, the contributions are:

1. **A design with thrust in free feet and working hands:** a 1.47 kg humanoid built on Rhoban’s open-source base robot with a ducted fan in each free-moving foot, two backpack rotors and two grippers, with CAD for every added part. Among the robots we found (Section 2), Jet-HR2 has fans in its feet but weighs 17 kg and has no arms, and the Tokyo humanoid uses its arms in flight but lifts from a backpack.
2. **A simulator and a physics-based controller** that fly the robot through arm and gripper motion, a 3D route and a landing.
3. **Flight control trained with reinforcement learning**, end-to-end and as a residual on the physics-based controller. The flying humanoids we found all use hand-designed or model-predictive control.
4. **Flight through a damaged fan:** in the full simulator, the residual policy finishes a route after a 20% thrust loss on one foot fan that the hand-written controller alone does not survive.
5. **An open design that costs a few thousand dollars**, with an itemised bill of materials.

We do not claim the first humanoid to use its hands in flight: the University of Tokyo humanoid did aerial manipulation with its arms in 2023 [34].

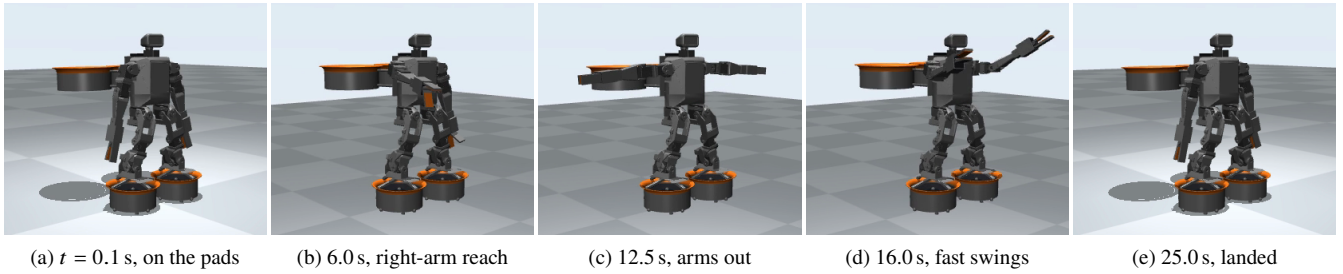


Figure 1: The final design (V4, Table 3) in the simulator during the 27 s scripted mission flown by the hand-written controller. Foot fans and duct lips in orange; backpack rotors behind the trunk. Frames are crops of the rendered mission video (MuJoCo, 500 Hz). The grippers open and close during the reaches; no object is modelled.

Table 1: Prior flying and thrust-assisted humanoids (sources in the text, checked 2026-10-01).

Robot	Group, year	Size	Thrust	Flight	Arms used in flight
iRonCub3	IIT, 2025	~70 kg	4 jet turbines (2 forearms, 2 back)	lifted ~50 cm, tethered	no (forearms carry jets)
LEONARDO	Caltech, 2021	2.5 kg, 75 cm	4 shoulder propellers	flies, walks	no arms
Jet-HR1	GDUT, 2017	6.5 kg	ducted fan in each foot	no; thrust-assisted steps	no arms
Jet-HR2	GDUT, 2022	17 kg, 83 cm	4 fans (2 waist, 2 feet)	took off, above 1 m	no arms
JSK humanoid	U. Tokyo, 2023	~3.5 kg, 35 cm	3-fan backpack	free flight, motion capture	yes
This work	2026	1.47 kg, 30 cm	2 fans in feet, 2 on back	simulation only	grippers, in simulation

2 Related work

Flying humanoids and thrust-assisted legged robots. Table 1 summarises the closest robots. iRonCub3 weighs about 70 kg and carries four JetCat P250 turbines, two on the forearms and two in a jetpack; its first flight on 18 June 2025 lifted it about 50 cm while it hung from a crane on a tether [12, 15]. The IIT group learned its aerodynamics with neural networks [27] and has applied data-fused model predictive control to it [13]. Jet-HR1 (6.5 kg, legs only) used a ducted fan in each foot to step over a 37 cm gap but did not fly [1]. Jet-HR2 (17 kg, 830 mm) has four 90 mm tilting fans, two at the waist and two at the feet, and took off at a thrust-to-weight ratio of 1.17, rising above 1 m [20]. LEONARDO (2.5 kg, 75 cm) flies on four tilted shoulder propellers, walks, slacklines and skateboards, and has no arms [2, 18]. The JSK humanoid (about 3.5 kg: a 1,691 g humanoid built on a Kondo KHR and a 1,652 g flight unit with three tilting fans) flies free under motion capture and uses its arms for aerial manipulation [34]. A 2026 follow-up walks a 2.0 kg, 0.4 m humanoid with two vectorable torso thrusters on a tether, using whole-body MPC [35]. Harpy uses thrusters to assist walking [29].

Aerial manipulation. Aerial manipulators are mostly multirotors that carry arms or grippers [26, 30]. The arm’s motion disturbs the platform, and compensating for it is a central control problem. A humanoid’s own arms play the same role here, except that the lift acts at the end of the legs.

Learned flight control and residual RL. Reinforcement learning has produced quadrotor controllers that transfer to hardware [14, 24] and reach champion-level racing [17], and learned models adapt flight control to wind [25]. Domain randomisation [36] is the usual way to cover the gap between simulator and hardware. Residual RL trains a policy that adds a correction to a conventional controller [16, 33], so learning starts from a system that already works. RL has also been applied to hybrid air-ground robots, including the DoubleBee [21] and a hopping quadcopter in simulation [6]. For flying humanoids we found hand-designed or model-predictive controllers [13, 34, 35], and no flight controller trained with RL.

3 How the robot works

Base robot. Rhoban’s open-source humanoid is about 30 cm tall, has 19 XL330-M288-T servos and a Raspberry Pi Zero 2W, and its bill of materials comes to about \$567. Its hardware is licensed CC BY-NC-SA 4.0 and its software GPL v3 [22]. We use its MJCF model and meshes, and the XL330 joint friction and armature identified by Rhoban [4, 7]. Rhoban does not publish the robot’s mass; with the two 18650 cells (92 g) and the soles removed, the bodies in the model sum to 694 g.

Foot fans. Each sole is replaced by a printed duct around a 102 mm (4 in) three-blade prop on an EMAX ECO II 2306-2400KV motor, with its own ESC at the motor. The duct is fused to Rhoban’s ankle bracket, cut from the base robot’s foot part (Fig. 2a), and the fan axis passes through the point where the ankle pitch and roll axes cross, so thrust puts no torque on the ankle servos.

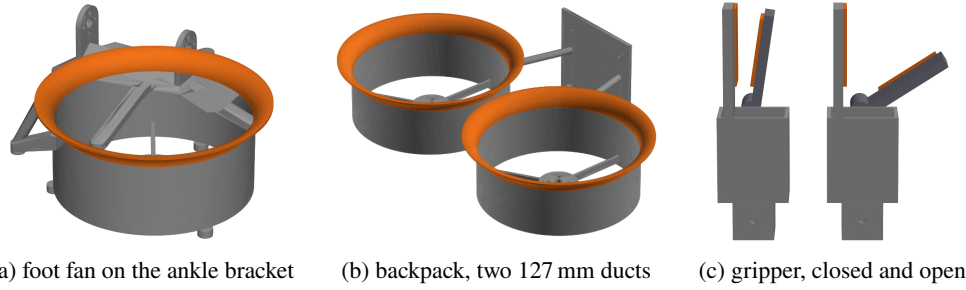


Figure 2: Printed parts of the final design, rendered from the STL files used in the simulator (motors, props and servo not shown; orange marks duct lips and gripper pads). (a) The 102 mm duct fused to Rhoban’s ankle bracket, heel spur at left. (b) Back plate, carbon booms and the two 127 mm ducts. (c) Gripper at the scripted closed (0.15 rad) and open (1.0 rad) jaw angles.

Table 2: Mass budget of the final design, summed from the simulation model. Printed parts are CAD volume times PLA density (solid).

Group	Mass (g)	Share
Base robot (cells, soles removed)	693.5	47%
Flight battery (4S 2200 mAh)	240.0	16%
Foot fans (ducts, motors, ESCs)	214.5	15%
Backpack (2 rotors and frame)	211.9	14%
Grippers (2, with XL330s)	80.2	5%
Avionics (estimate)	30.0	2%
Total	1470.1	

Table 3: Design versions. Power and flight time from endurance runs to 3.3 V per cell on a 4S 2200 mAh pack, arms cycling.

Layout	Mass	Power	Time
V1 two 152 mm foot pods, ankle-steered, bar between shins	1.21 kg	272 W	6.86 min
V2 V1 feet plus two 127 mm backpack rotors	1.46 kg	323 W	5.66 min
V3 102 mm foot fans, free legs, no grippers	1.39 kg	invalid	invalid
V4 V3 plus grippers (final)	1.47 kg	422 W	4.14 min

The duct has a 124 mm outer diameter, 50.2 mm length, 0.8 mm tip gap and a lip of $0.10 D$; the printed foot weighs 65.8 g in PLA. The robot stands on four pads under the duct rim and a heel spur that extends 88 mm to the rear. In flight the hips roll out 0.26 rad (15°), the smallest spread that keeps 10 mm between the ducts. No bar joins the shins, so each leg can move on its own.

Backpack rotors. Two 127 mm ducted rotors (154 mm outer diameter) sit at $x = -149.7$ mm, $y = \pm 81$ mm, $z = 70$ mm in the trunk frame, on 6 mm carbon booms from a 3 mm back plate (Fig. 2b). Their position comes from a layout study of the V2 design, in which each back rotor’s wake, modelled as a cylinder of the duct’s radius, misses the foot ducts by 10 mm in plan view; the final design keeps that position with smaller feet, and we did not repeat the check. Diagonal rotor pairs spin the same way so that their reaction torques cancel. A software limit keeps backward shoulder pitch below 0.3 rad so the hands stay out of the backpack ducts.

Grippers. A printed palm on each forearm holds one XL330, which swings a 40 mm jaw on a hinge 40 mm from the forearm tip against a fixed 46 mm jaw; both jaws carry 1.5 mm pads (Fig. 2c). The jaw’s range is 0 to 1.1 rad, and the scripted routine uses 0.15 rad closed and 1.0 rad open. The simulator moves the grippers but models no object and no grip force.

Battery, electronics and mass. A 4S 2200 mAh LiPo (240 g) sits in a tray at the back of the trunk. It feeds the four ESCs directly and the servos through a 6 V regulator. The simulator carries 30 g for the flight controller, regulator and wiring (an estimate) and a 2.5 W avionics load. The parts list (Section 8) names a Holybro Pixhawk 6C Mini flight controller, four Flycolor X-Cross 2 50 A ESCs, and three Arducam OV9281 global-shutter cameras with printed AprilTags for indoor tracking. Fig. 3 shows how the parts connect, and Table 2 gives the mass budget, summed from the bodies in the simulation model.

Design history. The design went through four versions in simulation (Table 3), and some figures come from earlier ones; each caption says which. V1 replaced the soles with 152 mm ducted pods, steered attitude by tilting them with the ankle servos, and joined the shins with a 10 g carbon bar. The author judged that a layout that depends on the ankle servos for attitude would not hold attitude on hardware when an arm moved, so V2 added the backpack rotors and controlled attitude by differential thrust, with the ankles only holding the stance. V3 replaced the 152 mm feet with 102 mm fans so the legs could move without the bar; its power logs are invalid because of a simulator bug (Section 4). V4 added the grippers.

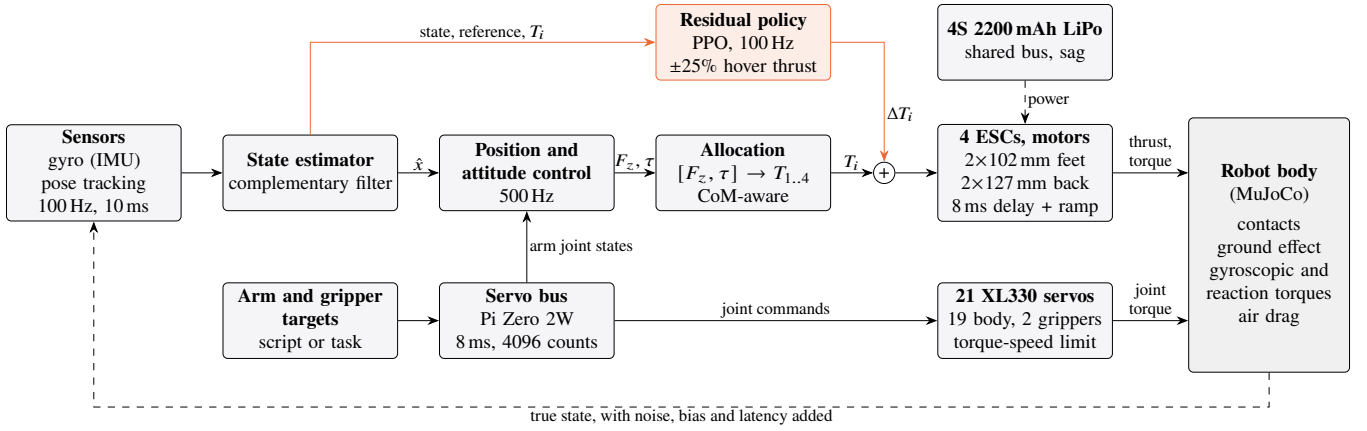


Figure 3: System diagram of the simulated robot. The flight path runs along the top row: sensors, estimator, position and attitude control, thrust allocation, ESCs and motors. The residual policy (orange) adds a per-rotor thrust correction. The servo path along the bottom carries arm, gripper and stance commands over the Dynamixel bus. Parts are those in the parts list; where the controller will run on the real robot is not yet decided.

4 Simulation model

The simulator runs MuJoCo (version 3.14) at a 2 ms step on Rhoban’s MJCF model of the base robot, with the added parts as bodies whose masses come from CAD volume and spec sheets. MuJoCo’s fluid model applies air drag to every body. Table 4 lists the parameters and their sources.

Rotors. Each rotor’s state is its speed ω ($n = \omega/2\pi$ in rev/s). Thrust and shaft torque are

$$T = k_{\text{duct}} C_T \rho n^2 D^4 g_{\text{ge}}(z) g_{\text{climb}}(J), \quad Q = C_Q \rho n^2 D^5, \quad (1)$$

with $C_T = 0.150$ and $C_Q = 0.0131$ derived from EMAX’s thrust table for the ECO II 2306 with a Gemfan 5045 three-blade prop at 16.8 V, and an estimated duct gain $k_{\text{duct}} = 1.15$ at equal shaft power, conservative next to what Pereira measured with tight tip clearance [28]. Ground effect uses the Cheeseman–Bennett form $g_{\text{ge}} = 1/(1 - (R/4z)^2)$ with z clamped at $R/2$, so it raises thrust by at most 1.33 times [5, 31]. Climb reduces thrust linearly to zero at an advance ratio of 0.9, and a bracket deck over each foot fan’s inlet removes 2.5% of its thrust. Each rotor applies its motor reaction torque and the gyroscopic torque $-\Omega \times (sJ_r \omega \hat{n})$ to the body it is mounted on, where Ω is that body’s angular velocity and $s = \pm 1$ the spin direction.

Motors and ESCs. The motor is a DC machine ($K_v = 2400$ rpm/V, 47 m Ω , 1.8 A no-load). With duty d and bus voltage V the winding current is $i = (dV - k_e \omega)/R_m$, clipped to $[-5, 44]$ A, and $J_r \dot{\omega} = k_e(i - I_0 \text{sgn } \omega) - Q$, integrated in four substeps per step. The rotor inertia $J_r = 7.5 \times 10^{-6}$ kg m 2 is a geometric estimate used unchanged for all rotor sizes. The ESC passes the throttle through an 8 ms delay and a 15 ms first-order ramp, tuned so total spin-up matches the 35–40 ms measured on the Agilicious thrust stand [10].

Battery and the implicit bus voltage. The 4S pack has a LiPo open-circuit voltage curve, 10 m Ω per cell and a 3.3 V per cell cutoff. Battery current sums the four motor currents (through 95%-efficient ESCs), the servo load through an 85%-efficient regulator, and the avionics. The motors share the pack resistance R_b , so bus voltage and motor currents depend on each other. Earlier versions computed each step’s motor currents from the previous step’s voltage. That explicit update has a loop gain of $R_b \sum_j d_j^2 / (R_m \eta_{\text{ESC}})$: 0.35 at V1’s hover duty and 0.50 at V2’s, but 1.53 once the 102 mm foot fans hover near 0.87 duty (back rotors near 0.31), where the update oscillates. The V3 logs show it: hover power swung between 5.9 and 972.6 W, and the endurance run hit the voltage cutoff after 0.5 s. The final simulator solves the voltage together with the currents,

$$V = \frac{V_{\text{oc}} - R_b \left(i_{\text{aux}} - \sum_j d_j k_e \omega_j / R_m \right)}{1 + R_b \sum_j d_j^2 / R_m}, \quad (2)$$

which is stable at any duty, and checks the cutoff against a smoothed cell voltage after the first 5 s. In the V4 mission, hover power stays between 322 and 659 W. We exclude all V3 results.

Table 4: Main simulation parameters. “Spec” is a manufacturer value, “derived” is fitted to published data, and “estimate” must be replaced by hardware tests.

Quantity	Value	Status	Source
Motor K_V , R_m , I_0 , $I_{\max}$	2400 rpm/V, 47 m Ω , 1.8 A, 44 A	spec	EMAX ECO II 2306 [8]
C_T , C_Q	0.150, 0.0131	derived	EMAX table, 500 g at 19,200 rpm; C_Q assumes ~80% drive efficiency
Duct gain k_{duct}	1.15	estimate	printed duct, 0.8 mm tip gap; cf. [28]
Rotor inertia J_r	7.5×10^{-6} kg m ²	estimate	geometry (prop plus bell), same for all sizes
ESC delay, ramp, efficiency	8 ms, 15 ms, 0.95	estimate	tuned to 39 ms spin-up [10]
Ground effect	$1/(1 - (R/4z)^2)$, $z \geq R/2$	model	[5, 31]
Foot inlet blockage	0.975	estimate	bracket deck over the inlet
Battery	4S 2200 mAh, 10 m Ω /cell, 240 g	spec, estimate	LiPo OCV curve; cell resistance includes wiring
Servo stall, speed	0.62 N m, 12.9 rad/s	spec (scaled)	XL330-M288 e-manual, 5 V scaled to 6 V [37]
Servo gains, latency	k_p 50, k_d 0.5; 8 ms	model, measured	base robot’s MJCF; Rhoban BAM [4]
Gyro noise, bias	0.005 rad/s, 0.01 rad/s	estimate	IMU class
Pose tracking	100 Hz, 10 ms, 0.5 mm, 0.002 rad	assumption	indoor tracking

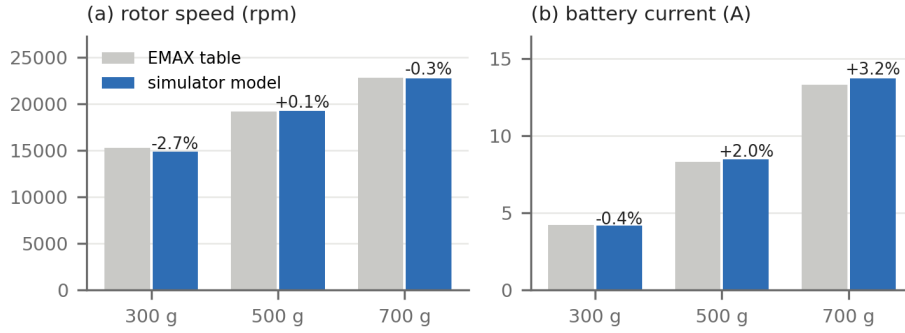


Figure 4: Motor model against EMAX’s published table (ECO II 2306-2400KV, Gemfan 5045 three-blade, 16.8 V, open prop). Labels give the model’s error. C_T and C_Q come from the same table, so this is a consistency check of the electrical model.

Table 5: Training environment (MJX) against the full CPU simulator used for every evaluation.

	MJX training environment	Full CPU simulator
Air drag	none	MuJoCo fluid model on every body
State used by the base controller	true state	estimator on noisy, delayed gyro and pose
Policy observations	true state plus Gaussian noise	estimator output
Servos	PD in torque–speed envelope, no delay	plus 8 ms bus latency and 4096-count quantisation
ESC	ramp, τ randomised 10–30 ms	8 ms delay plus 15 ms ramp
Battery	fixed voltage per episode, 14.4–16.6 V	discharging pack with sag, implicit bus solve
Contacts and solver	foot pads only, 4 iterations, no joint friction or limits	full model

Servos and sensing. Each XL330 is a PD position loop ($k_p = 50$, $k_d = 0.5$, the base robot’s MJCF defaults) inside a torque–speed envelope with 0.62 N m stall torque and 12.9 rad/s no-load speed (the e-manual’s 5 V values scaled to 6 V [37]). Commands are quantised to 4096 counts per revolution and delayed 8 ms, inside the 5.9–10.2 ms range from Rhoban’s identification [4]. The gyro has 0.005 rad/s noise and a random bias of order 0.01 rad/s. Pose tracking, standing in for motion capture or the planned cameras, reports at 100 Hz with 10 ms latency, 0.5 mm and 0.002 rad noise, and a complementary filter fuses it with the gyro.

Check against published motor data. Fig. 4 compares the motor model, with the duct gain set to 1, against EMAX’s table at 300, 500 and 700 g of thrust: rotor speed agrees within 2.7% and battery current within 3.2%. C_T and C_Q were fitted to the same table (the 500 g speed matches by construction), so this shows that the electrical model holds across operating points. It is not an independent test of thrust, and it says nothing about the duct, the 102 mm prop or full throttle. The table’s own full-throttle row does not fit its other rows: at $C_T = 0.150$ its 1380 g point needs 31,925 rpm, where C_Q implies 39.5 A against the 32.7 A listed. A blade-element model of two printed props, written for V2, matched the table’s thrust within 4.4% but over-predicted power by 49 to 58% until profile drag was fitted; it recommends flying commercial props first.

Two simulators. Training needs thousands of parallel copies, so the RL environment reimplements the rotor, motor and servo models in JAX on MuJoCo MJX [23] and trims the physics (Table 5). Every evaluation of a trained policy in this report runs in the full CPU simulator, where the policies never trained.

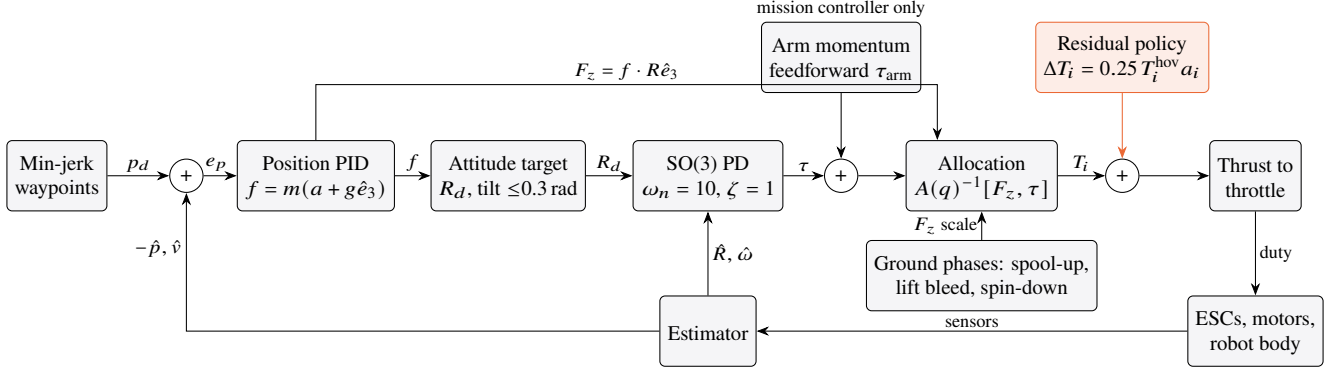


Figure 5: Control loop. The base controller runs at 500 Hz on the estimated state. The arm momentum feedforward is part of the controller used for the scripted mission and fault tests only. The residual policy (orange) runs at 100 Hz, and its action reaches the rotors one policy step (10 ms) late.

5 Control

The base controller turns a reference trajectory into four rotor thrusts at 500 Hz from the estimated state (Fig. 5). It exists in two implementations. The one used for the scripted mission, endurance and fault tests recomputes the inertia from the current arm pose and includes the arm momentum feedforward. The other, written once for NumPy and JAX so that training and evaluation share it, drives the RL environment and the 3D tour; it uses the inertia at the start pose and has no feedforward.

Reference, position and attitude. The tour and the training routes use minimum-jerk segments between timed waypoints, $s(u) = 10u^3 - 15u^4 + 6u^5$ [9], which supply velocity and acceleration feedforward; the mission uses cosine-blended segments. A PID loop on position ($k_p = 6$, $k_d = 4.5$, $k_i = 1.5$, integrator clipped and active only when airborne) gives an acceleration a and force $f = m(a + g\hat{e}_3)$. The desired body z axis is $f/\|f\|$, tilted at most 0.3 rad, and a geometric attitude controller on SO(3) [19] computes

$$e_R = \frac{1}{2} (R_d^T R - R^T R_d)^V, \quad \tau = R \left(J(-\omega_n^2 e_R - 2\zeta\omega_n\omega) + \omega \times J\omega \right) + \tau_{arm}, \quad (3)$$

with $\omega_n = 10$ rad/s, $\zeta = 1$, and collective lift $F_z = f \cdot R\hat{e}_3$. The feedforward τ_{arm} is the low-passed time derivative of the arms’ angular momentum about the whole-body centre of mass, relative to the body’s rotation, with gain 1; it cancels a swinging arm’s reaction torque before it shows up as attitude error.

Allocation. Rotor i at position r_i with thrust axis $\hat{n}_i$ contributes the column $[\hat{n}_i \cdot R\hat{e}_3; (r_i - c) \times \hat{n}_i - s_i(C_Q D_i / C_T k_{duct}) \hat{n}_i]$, where c is the current centre of mass. Positions and axes come from the current pose, so leg and arm motion enter directly. Solving the 4×4 system for $[F_z, \tau]$ gives the thrusts; if any falls outside $[0.3 \text{ N}, T_{max,i}]$, the controller drops the yaw demand, solves again, and clips.

Throttle and ground phases. A steady-state motor model inverts each thrust into a feedforward duty, corrected by $0.3(\omega^* - \omega)/\omega^*$. On the ground the rotors spool to 35% of hover thrust before takeoff. At the end of the descent the reference continues 4 to 6 cm below the floor and the collective is bled at 25% per second, down to half, so ground effect cannot hold the robot just above the floor. At touchdown the thrusts ramp to zero over 0.6 s.

6 Learning

All training used Brax PPO [11, 32] on the MJX environment on one cloud GPU (an NVIDIA L4 for the residual runs), with 4,096 parallel environments where the number was recorded. The policy acts at 100 Hz, every five physics steps, and its action reaches the rotors one step late. Table 6 lists every run.

End-to-end PPO (ppo1). The first policy commands the four rotor thrusts directly. Episodes last 500 steps (5 s) and start in the air with random position (up to 10 cm), tilt (up to 0.3 rad) and velocity; the target jumps once, by up to 30 cm, partway through. The arms and grippers follow random keyframes, and stay still in 20% of episodes. Rotor efficiency (0.9–1.1), battery voltage (14.4–16.6 V) and ESC ramp (10–30 ms) are randomised per episode. The 51 observations cover attitude, noisy gyro, position error, velocity, previous action, rotor speeds, arm and gripper angles, velocities and commands, and battery voltage.

Table 6: Training runs, in order. Steps are policy steps (100 Hz).

Run	Environment	Steps	Wall time	Outcome
ppo1	end-to-end hover, airborne start	41.3M logged	536 s	learned to hover (497 of 500 steps at 20.6M); ended without a final checkpoint; best sim-to-sim result at 27.5M
mission2–6	end-to-end mission, ground start	6.6M–126M	285–1688 s	collapsed: episodes shortened and the policy stopped taking off
residual1	residual, default exploration noise	17.0M	423 s	return fell from 488 to 343; stopped
residual2	residual, small noise, Brax auto-reset	21.3M	473 s	return fell from 538 to 462; stopped
residual3	residual, small noise, fresh reset	102.2M	1533 s	used; best checkpoint at 51.1M

The reward pays for staying alive and near the target and penalises tilt, yaw, body rates, velocity and action changes. Policy and value networks are MLPs with three layers of 128 and 256 units.

Mission runs and the auto-reset leak. Runs mission2 to mission6 trained the same kind of policy on a full mission in 1,000-step episodes: start on the pads with the rotors stopped (60% of episodes; the rest start airborne), take off, hover while the arms move, descend, touch down and spin down. All five collapsed. Episode length fell (mission2: from 346 to 165 steps), landings fell to almost none, and a checkpoint flown in the CPU simulator never left the ground (maximum height 2.1 cm).

The cause we found is in the training wrapper. Brax’s `AutoResetWrapper` [3] restores a finished episode’s physics state and observation but keeps the environment’s info dictionary, where our environments keep the episode clock, the route, the phase flags and the randomised parameters. After its first reset, each environment carried the previous episode’s clock past its landing time. In the mission environment that pays the on-ground reward and switches off the check that fails an episode lagging behind the climb, so staying on the ground paid. The fix is a replacement wrapper that resets every finished environment completely with a fresh random key, keeping only the step counter, truncation flag and metrics. Runs residual2 and residual3 start from identical parameters, and the reset handling is the only recorded change between them; residual2 declined steadily and residual3 improved (Fig. 10a). For the mission runs the attribution rests on that mechanism and on the training script’s history (the fix was written after all of them). We did not rerun the mission environment with the fix.

Residual PPO (residual3). The policy outputs $a \in [-1, 1]^4$, and rotor i ’s thrust becomes $\text{clip}(T_i^{\text{base}} + 0.25 T_i^{\text{hov}} a_i, 0, T_{\text{max},i})$, where T^{hov} is the static hover split. The mean output layer starts near zero (weight scale 10^{-4}) and the exploration noise starts at a standard deviation of 0.15, so training begins at the base controller. Each 12 s episode (1,200 steps) starts on the pads with the rotors stopped: spool-up, takeoff at a random time (0.3–0.8 s), a climb of 0.4–0.6 m, two random waypoints (each up to 0.6 m away in x and y , within ± 1 m, at 0.3–0.8 m height), and a landing below the last one, while the arms and grippers move through random keyframes. Per episode we randomise rotor efficiency (0.92–1.08), battery voltage (14.4–16.6 V) and ESC ramp (10–30 ms); in a quarter of episodes one random rotor loses 10 to 25% of its thrust at a random time between 3 and 8 s. The 68 observations add to the end-to-end set the reference velocity and acceleration, the base controller’s thrusts normalised by hover thrust, three phase flags (airborne, descending, touched down), height, and the position integrator. The per-step reward is 1 for staying alive, plus $2e^{-(d/0.05)^2} + 0.5e^{-(d/0.2)^2}$ for distance d from the reference, plus terms for velocity error, tilt, body rate, action change and action size, plus 1 while landed, minus a penalty for descending faster than 0.4 m/s within 5 cm of the floor. Tilt above 1 rad or a distance above 0.5 m ends the episode with -20 .

Training setup and cost. Learning rate 10^{-4} , entropy cost 10^{-3} , discount 0.99, clipping 0.2, batch 512 in 32 minibatches, unroll 20, four updates per batch, observation normalisation, 4,096 environments. The run took 102M steps in 1,533 s, about 82,000 policy steps (410,000 physics steps) per second after compilation. We picked the checkpoint with the highest training-evaluation return (51.1M steps) before any test on the evaluation route. The logged training time of all nine runs sums to about 1.8 h on one GPU. We did not record the bill; at the on-demand L4 price used in the budget (\$0.80/h), the logged time comes to under \$2, with compilation, evaluation and idle time on top.

7 Results

All results are from simulation, and for the final design (V4) unless a caption says otherwise.

Layout and placement. Before the four-rotor design, a first-principles placement study compared 18 rotor layouts carrying the same 4S 1800 mAh pack (Fig. 6). It uses momentum theory with a figure of merit and drive efficiency calibrated on the EMAX table, checks the hip-roll servo’s holding torque against an estimated 0.22 N m continuous limit, and counts the share of 4,000 random arm poses that put a hand within 1.3 rotor radii of a prop. Fans in the feet flew as long as back-mounted rotors (5.1 min with a bar between the shins at a 32° stance, against 3.9 to 4.8 min on the back), and no arm pose reached a foot fan,

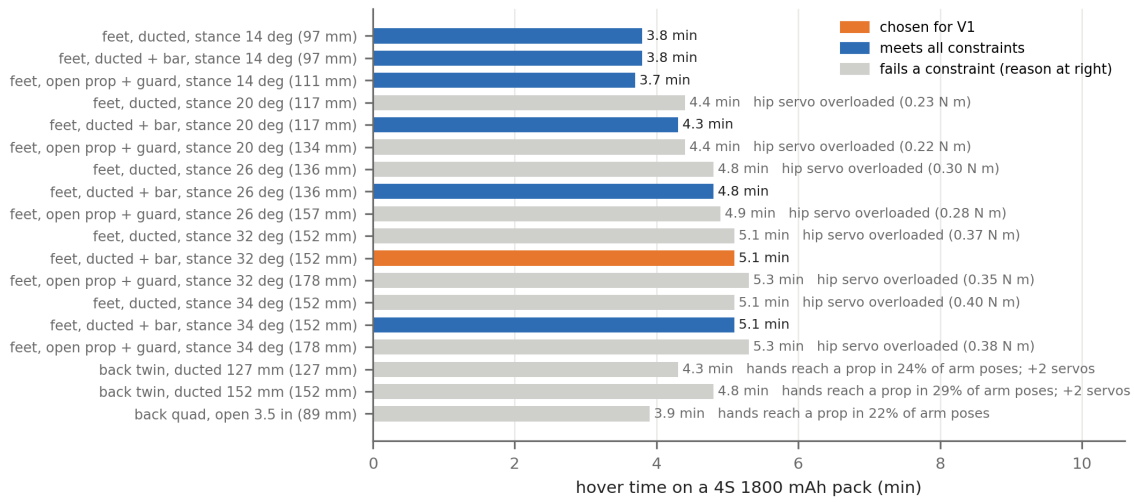


Figure 6: Placement study (first-principles model, earlier two-foot-rotor concept, 4S 1800 mAh pack for every option). Orange: the layout chosen for V1. Grey bars fail the constraint given at right. This study predates the backpack rotors of V2 to V4.

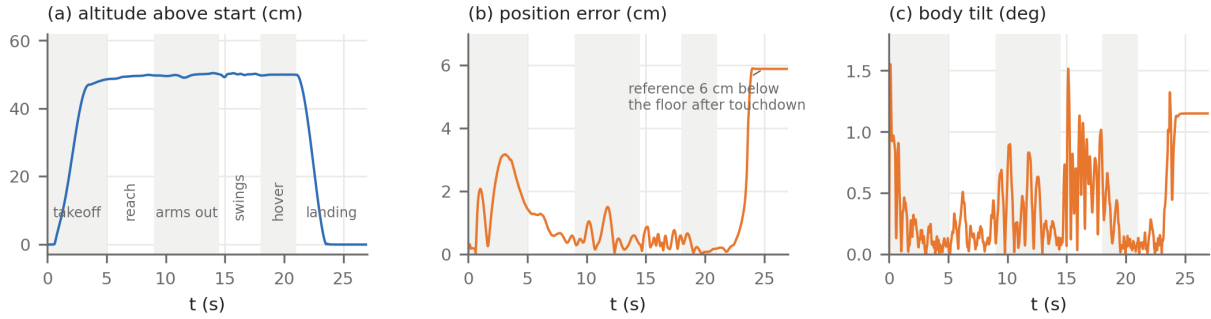


Figure 7: The 27 s mission, hand-written controller, final design: altitude, position error and body tilt, with the mission phases shaded. After touchdown the error settles at 6 cm because the landing reference continues below the floor by design.

against 22 to 29% for back rotors. A wider stance cut hover power from 332 W at 14° to 252 W at 32°; without the bar, the hip-roll servos held 0.37 N m, 170% of their estimated continuous rating. The final design reintroduces back rotors for attitude control, so the hazard result covers only the foot fans; the shoulder limit keeps the hands out of the backpack ducts, and we did not recompute the hazard metric for the final layout. Freeing the legs cost power (Table 3): the 102 mm foot fans need about 30% more than the 152 mm ducts (422 against 323 W), and flight time falls from 5.66 to 4.14 min.

Hands in flight. In the 27 s scripted mission (Fig. 1, Fig. 7) the robot takes off from the pads, hovers at 0.5 m, reaches forward with the right arm, moves both arms forward and then out to the sides, swings them fast in alternation, hovers and lands, while the grippers open and close. The hand-written controller held position within 1.5 cm during the reaches and 0.88 cm during the fast swings, with at most 0.9° and 1.5° of tilt, touched down at 23.53 s and came to rest at 1.15°. Mean hover power was 426.5 W. No servo reached stall in flight: the largest in-flight torque was 0.37 N m at the shoulder pitch during the swings (stall 0.62 N m), and the largest hover holding torque 0.12 N m at the hip roll, below the estimated 0.22 N m continuous limit. Knees, ankles, hip pitch and grippers saw higher torques only in the first 0.11 s on the ground, as the servos pulled to their start pose.

Thrust margin and endurance. The foot fans carry 77% of the thrust. In hover each makes 5.69 N, 77% of its 7.42 N steady-state maximum at 15 V, while each back rotor makes 1.66 of a possible 15.27 N (Fig. 8a). The foot fans peaked at 6.63 N (89% of the maximum) during the fast swings, and the allocator hit a limit on 9 of 13,500 steps. With zero torque demand, total lift can rise only to 1.33 times the weight before a foot fan saturates. At hover the allocator can add 0.48 N m of roll torque, 0.49 N m of pitch and 0.056 N m of yaw (Fig. 8b). On one 4S 2200 mAh pack, with the arm and gripper routine repeating, the robot flew 4.14 min before the smoothed cell voltage reached 3.3 V, at a mean 421.8 W, with 7% charge left; as the voltage fell, the allocator saturated more often (260 steps over the flight).

Fault tolerance of the hand-written controller. To stand in for a damaged prop or a weak ESC, we multiplied one rotor's thrust by 0.7 from $t = 10$ s in an 18 s mission (Fig. 8c). Both earlier designs survived a 30% loss on a foot rotor (V1: 8.7° and

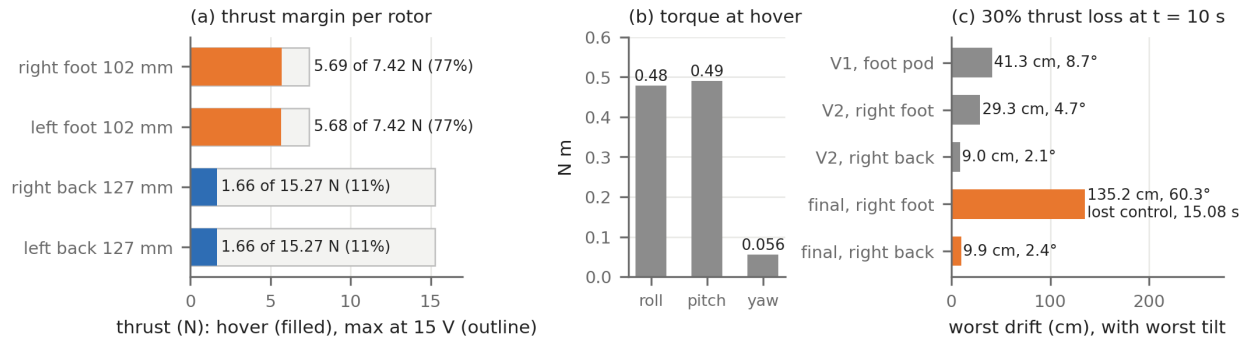


Figure 8: (a) Mean hover thrust per rotor in the mission against its steady-state maximum at 15 V. (b) The largest pure torque the allocator can add at hover with lift fixed and every rotor inside $[0.3 \text{ N}, T_{\max}]$. (c) Worst drift and tilt after one rotor drops to 70% thrust at $t = 10 \text{ s}$, hand-written controller; grey rows are the earlier V1 and V2 designs, orange rows the final design.

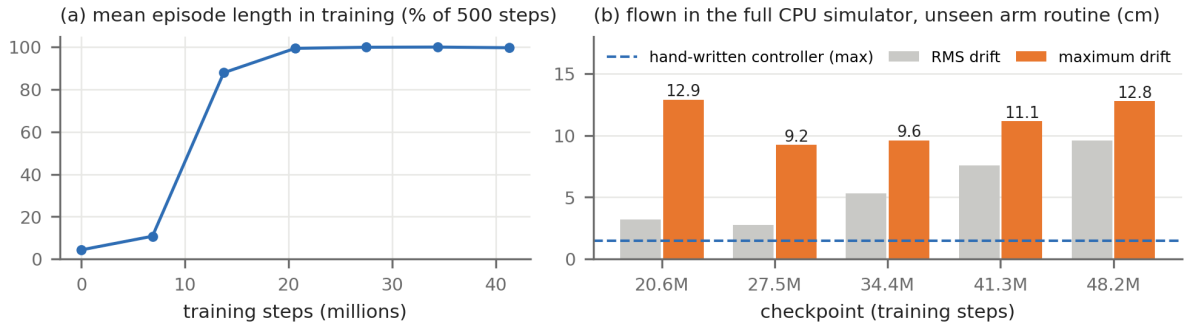


Figure 9: End-to-end PPO (ppo1, hover only, airborne start). (a) Mean episode length during training. (b) The five checkpoints that flew the full 20 s in the full CPU simulator, through an arm and gripper routine the policy never saw, against the hand-written controller's 1.5 cm maximum on the same routine.

41.3 cm worst case; V2: 4.7° and 29.3 cm). The final design survives it on a back rotor (2.4°, 9.9 cm) but loses control 5.1 s after a 30% loss on a foot fan: its foot fans already run at 77% of their maximum, so the healthy one cannot make up the difference.

Servo latency (earlier design only). On V1, which steered with its ankle servos, maximum tilt stayed at or below 5.1° up to 32 ms of servo latency, four times the modelled 8 ms, and reached 22.6° at 48 ms without a crash. We have not swept latency on the final design.

End-to-end PPO. The ppo1 policy survived 497 of 500 steps on average by 20.6M steps and full episodes by 34.4M (Fig. 9). We flew every saved checkpoint once in the full CPU simulator, starting in hover, through a 20 s arm and gripper routine the policy had never seen. The untrained policy lost control after 0.22 s and the 6.9M checkpoint after 0.71 s; the 13.8M checkpoint hovered for 11.6 s, drifting up to 89 cm, before it lost control. From 20.6M steps on, every checkpoint flew the full 20 s. The best, at 27.5M steps, drifted at most 9.2 cm (2.8 cm RMS) with up to 12.8° tilt, against 1.5 cm for the hand-written controller on the same arm routine. RMS drift grew with further training, to 5.3, 7.6 and 9.6 cm at 34.4M, 41.3M and 48.2M steps. The policy only hovers; the mission runs that tried to teach it takeoff and landing collapsed.

Residual PPO training. With the base controller alone the training evaluation return is 538. Residual3 passed it by 17M steps, stayed above it, and peaked at 557 at 51.1M steps (Fig. 10a), 3.5% above the base. With Brax's default exploration noise (about 0.7 standard deviation) the return fell from the first update, and with small noise but the default auto-reset it declined steadily (Fig. 10a). In a 64-episode check in the training environment, the base controller alone survived every episode, including all 14 with a 10 to 25% rotor fault; faults became fatal only in the full CPU simulator. Flown once each through the fault tour below (Fig. 10c), every run-3 checkpoint from 0 to 12.8M steps lost control (between 11.55 and 13.76 s) and every checkpoint from 17.0M on landed, while the training return barely moved across that boundary (536 at 12.8M, 541 at 17.0M). By this test the 55.4M checkpoint is slightly better than the 51.1M one we use (7.83 against 7.97 cm RMS, 3.40 against 5.40 cm landing error).

3D tour: residual policy against the base controller. The evaluation route, which neither controller trained on, starts on the pads, climbs to 0.5 m, visits four waypoints at (0.5, 0, 0.5), (0.5, 0.5, 0.8), (−0.3, 0.5, 0.4) and (−0.3, −0.3, 0.6) m from the start, holds each for 0.6 s, and lands below the last, while the arms and grippers run the mission routine. The waypoints span 0.8 m

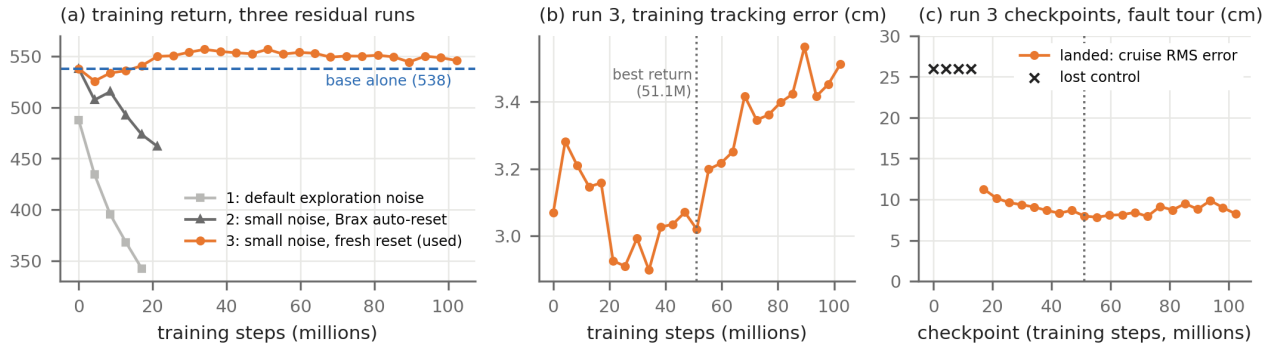


Figure 10: Residual PPO on random 3D routes with ground takeoff, two waypoints and landing (4,096 MJX environments). (a) Evaluation return of the three residual runs: run 1 with Brax’s default exploration noise, run 2 with small noise and Brax’s auto-reset, run 3 with small noise and a complete reset of every finished episode. Runs 2 and 3 start from identical parameters. (b) Mean tracking error per step in run 3 during training; its mean episode length stayed between 1,190 and 1,200 of 1,200 steps. (c) Every run-3 checkpoint flown once through the fault tour in the full simulator with CPU inference: cruise RMS error for checkpoints that landed, a cross for those that lost control. Dotted: the 51.1M checkpoint used in this report.

Table 7: 3D tour in the full CPU simulator, final design. One deterministic run per column, policy inference on the CPU. Fault: right foot fan at 80% thrust from $t = 6$ s. Pad-contact speed is the trunk’s vertical speed in the frame before the first pad-floor contact, from 60 Hz replays of the recorded runs.

	Base, healthy	Base + RL, healthy	Base, fault	Base + RL, fault
Waypoints held within 10 cm (of 4)	4	4	1	3
Waypoint hold error (cm)	3.7, 4.0, 4.6, 3.6	4.3, 2.7, 5.5, 3.6	3.7, 18.2, 29.0	4.3, 14.3, 6.3, 4.5
Cruise RMS / maximum error (cm)	4.37 / 8.73	5.24 / 12.1	21.8 / 76.7	7.97 / 15.5
Cruise maximum tilt (deg)	11.6	12.2	57.5	12.0
Lost control at	n/a	n/a	11.49 s	n/a
Landing spot error (cm)	19.1	1.09	crashed	5.40
Vertical speed at touchdown trigger (m/s)	0.211	0.085	crashed	0.257
Vertical speed at first pad contact (m/s)	0.187	0.109	crashed	0.233
Cruise power (W)	477.5	482.6	498.4	501.8

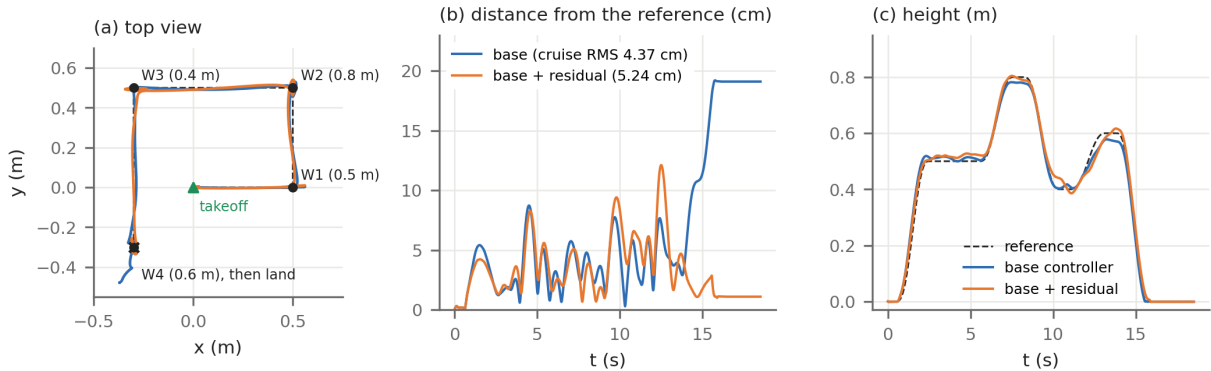


Figure 11: The 3D tour with healthy rotors in the full CPU simulator, base controller (blue) and base plus residual policy (orange). (a) Top view; dashed is the reference, and the robot lands below W4. (b) Distance from the reference; after landing it shows the landing-spot error. (c) Height.

in x and y and reach 0.8 m in height; the base controller’s flown path covered $0.93 \times 1.01 \times 0.78$ m. We flew it four times in the full CPU simulator: base controller alone and with the residual policy, with healthy rotors and with the right foot fan losing 20% of its thrust at $t = 6$ s (the controller is not told). The policy runs deterministically with its inference on the CPU, so these runs repeat exactly (see Reproducibility below). Table 7 and Figs. 11 and 12 give the results; each is one run.

Healthy rotors. Both controllers reached all four waypoints (Fig. 11). The residual policy tracked worse in cruise (5.24 against 4.37 cm RMS, maximum 12.1 against 8.7 cm). It landed better: 1.09 cm from the landing spot, where the base controller drifted 19.1 cm sideways during the descent. Two speeds describe the touchdown. At the touchdown trigger (estimated height first below 1.2 cm, which also starts spin-down) the residual run was descending at 0.085 m/s and the base run at 0.211 m/s. At first pad contact the trunk moved at about 0.109 m/s with the residual and 0.187 m/s without; the residual run slowed to about 0.02 m/s just under 1 cm above its resting height and then settled onto the pads as the motors spun down. In cruise the residual

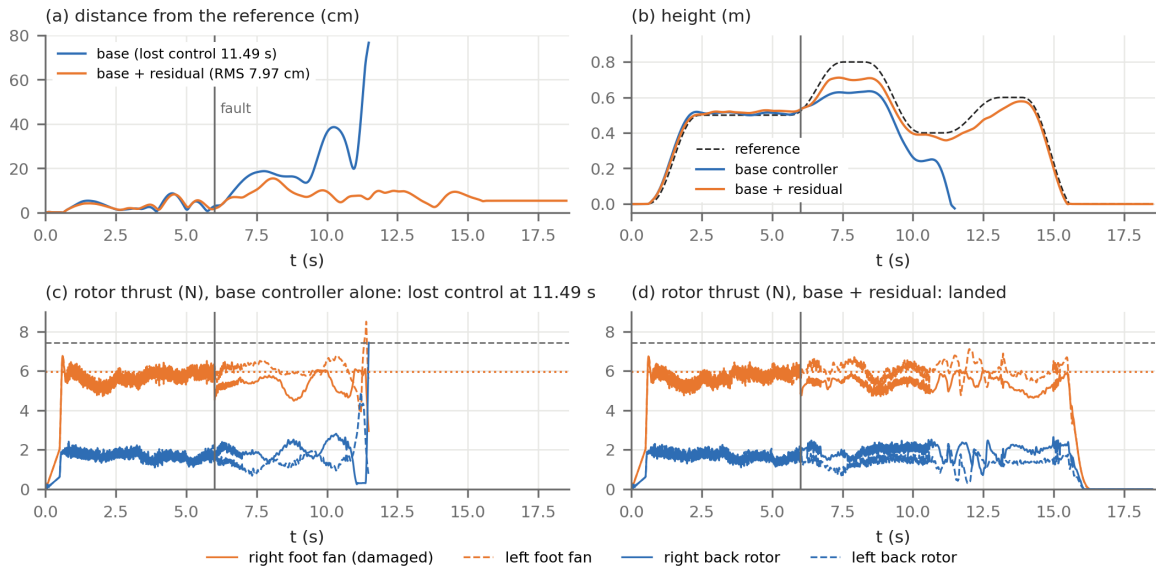


Figure 12: The same tour with the right foot fan at 80% thrust from $t = 6$ s (vertical line). (a) Distance from the reference. (b) Height. (c, d) Rotor thrusts without and with the residual policy; dashed grey is the foot fan’s 7.42 N maximum and dotted orange the damaged fan’s reduced maximum (5.94 N).

correction stayed between 1.2 and 5.5% of hover thrust (25-sample moving mean), and cruise power was 5 W higher (482.6 against 477.5 W).

Foot-fan fault. After the right foot fan lost 20% of its thrust, both controllers shifted load to the left foot fan and the right back rotor (Fig. 12c,d). The base controller alone fell behind the climb to the second waypoint, went into a growing oscillation, and lost control at 11.49 s, 5.5 s after the fault (Fig. 12a,b). With the residual, the robot held three of four waypoints within 10 cm (it reached only about 0.7 m of waypoint 2’s 0.8 m height) and landed upright 5.40 cm from the spot; its touchdown trigger fired at 0.257 m/s and its pads met the floor at about 0.233 m/s. The correction peaked near 8.2% of hover thrust after the fault.

Reproducibility. Closed-loop runs with the policy on the GPU do not repeat. The same healthy-tour command run twice with GPU inference on the L4 gave 4 of 4 waypoints, 4.92 cm RMS and a 0.72 cm landing error the first time, and 3 of 4 waypoints, 5.78 cm and 2.38 cm the second: small floating-point differences in inference grow through the closed loop. With inference on the CPU, repeated runs are bit-identical, so every residual-policy result in this report uses CPU inference and can be reproduced exactly. Base-controller runs use no network and repeat exactly, and the end-to-end hover tests gave the same results with GPU and CPU inference. A checkpoint near the survive-or-crash boundary can flip: the 17.0M checkpoint survived the fault tour with CPU inference but lost control in a GPU run, so single outcomes between about 13M and 21M steps are uncertain.

8 Bill of materials

Table 8 lists every part, tool and service needed to build and test the robot starting with no printer or tools. Prices were checked on 2026-10-01 in US dollars, before tax and shipping. Many retailer pages blocked automated access, so the Chk column records how each price was checked: P, the product page showed it; L, search-listing text for that page; B, Rhoban’s bill of materials for the base robot; E, an estimate with no live listing; C, computed. Prices marked L or E need confirming before ordering. The full build totals \$4,553.61 before tax and shipping. The lean build, \$2,696.92 before tax, keeps every flight part but uses a load-cell thrust stand, a cheaper printer and dryer, a flight controller with a four-in-one ESC, PLA only and basic netting. The lean stack puts its ESC in the trunk, which adds about 30 cm of motor leads to each foot. Motion capture is not in the budget (one OptiTrack camera lists at \$659); the build uses the three cameras with AprilTags, or a borrowed motion-capture lab.

9 Applications and roadmap

At 1.47 kg, with about 4 min of flight and lift that can rise only to 1.33 times its weight, the robot in this report is a research platform. Below we set out where the idea could lead, from published accounts of robots in use today. Every step past the first needs a larger robot or different propulsion, and none of it is demonstrated here.

Table 8: Bill of materials. Unit prices in USD. Lean values differ from full only where noted.

#	Item	Qty	Unit	Full	Lean	Chk	Source
1	Base robot, Rhoban's humanoid (19 XL330-M288-T, Pi Zero 2W, Robot HAT, printed parts, screws); built from source	1	567.00	567.00	567.00	B	Rhoban BOM [22]
2	Servo price update: BOM uses \$23.90 per XL330, listing \$27.49	19	3.59	68.21	68.21	L	ROBOTIS US
3	Dynamixel tools: U2D2 (\$36.92), U2D2 Power Hub (\$19), 5 V supply (\$5)	1	60.92	60.92	60.92	L+B	ROBOTIS US
4	Gripper servos, XL330-M288-T	2	27.49	54.98	54.98	L	ROBOTIS US
5	EMAX ECO II 2306 2400KV motor	4	19.49	77.96	77.96	L	RaceDayQuads
6	ESC, Flycolor X-Cross 2.50 A single, one per rotor (lean: none)	4	26.99	107.96	0.00	L	RaceDayQuads
7a	Flight controller, Holybro Pixhawk 6C Mini (full only)	1	176.99	176.99	0.00	L	Pyrodrone (search listing)
7b	SpeedyBee F405 V4 BLS 55 A stack, FC and 4-in-1 ESC (lean only)	1	86.49	0.00	86.49	P	Wrekd
8	Back rotor props, Gemfan Hurricane 51466 tri-blade, 4-pack	5	3.29	16.45	16.45	L	Wrekd
9	Foot fan props, Gemfan Hurricane 4023 tri-blade 4 in, 4-pack	5	2.99	14.95	14.95	L	RaceDayQuads
10	6 V UBEC, 5 A	1	29.80	29.80	29.80	E	eBay.de listing
11	LiPo 4S 2200 mAh 45C XT60 (Tattu)	3	27.00	81.00	81.00	E	listing, currency unconfirmed
12	Balance charger, ToolkitRC M7AC	1	91.49	91.49	91.49	L	RaceDayQuads
13	Carbon tube 8×6 mm, 1 m	2	9.99	19.98	19.98	L	Motion RC
14	Wire, XT60/XT30 connectors, heat shrink, M2/M3 hardware	1	40.00	40.00	40.00	E	hobby-store basket
15	PLA filament 1 kg	2	22.99	45.98	45.98	E	Bambu store (Asia page)
16	PA-CF filament 1 kg (PAHT-CF), parts near the motors (lean: none)	1	94.99	94.99	0.00	P	GigaParts
17a	Thrust stand, Tyto Robotics Series 1585 (full only)	1	1075.00	1075.00	0.00	L	Tyto Robotics
17b	Load cell 5 kg + HX711 (\$9.99), Arduino Uno (\$37.99), watt meter (\$25) (lean only)	1	72.98	0.00	72.98	L+E	ShillehTek
18a	Safety netting, drone netting panels (full only)	1	212.24	212.24	0.00	L	US Netting (listing)
18b	Nylon netting 25×25 ft, 1 in mesh (lean only)	1	70.11	0.00	70.11	L	Home Depot
19	Net frame, tether line, ceiling anchor, carabiners	1		80.00	40.00	E	lumber, PVC, climbing hardware
20	Arducam OV9281 global-shutter USB camera, with printed AprilTags	3	63.12	189.36	189.36	L	RobotShop
21	Spares, 20% of rows 1, 2, 4–9			216.90	177.21	C	computed
22	3D printer, Bambu Lab P2S (lean: PIS \$399 + hardened hotend \$34.99)	1	549.00	549.00	433.99	P	Bambu Lab
23	Filament dryer, Sunlu S4 (lean: S2)	1	116.99	116.99	44.99	P	Sunlu
24	Soldering: Pinecil V2, 65 W charger, cable, solder, flux pen, tip cleaner, helping hands	1	100.27	100.27	100.27	P	Pine64, others
25	Electrical: multimeter, strippers, crimper, flush cutters, heat-shrink, heat gun	1	86.55	86.55	79.15	P	IWISS, others
26	Mechanical: hex drivers 1.5–3 mm, precision driver kit, digital calipers	1	84.70	84.70	67.70	P+L	iFixit, others
27	Bench power supply 30 V 10 A (lean: none, use the LiPo)	1	54.99	54.99	0.00	P	Walmart
28	Battery and eye safety: LiPo bag, ABC extinguisher, ANSI Z87 glasses, cell checker	1	63.95	63.95	45.95	P+L	RaceDayQuads, others
29	Bench supplies: cutting mat, hobby knife, zip ties, threadlocker, glue gun, tape	1	75.00	75.00	75.00	P	Walmart (ranges)
30	Cloud GPU for retraining on measured data: 100 h AWS g6.xlarge (L4) at \$0.8048/h plus storage (lean: spot or RunPod)	100 h		100.00	45.00	P	AWS price file; RunPod
Robot base (rows 1–4)				751.11	751.11		
Flight hardware (rows 5–14)				656.58	458.12		
Filament and materials (rows 15–16)				140.97	45.98		
Test, safety net, tether, tracking (rows 17–20)				1556.60	372.45		
Spares (row 21)				216.90	177.21		
Shop: printer, dryer, tools, bench supply, safety gear (rows 22–29)				1131.45	847.05		
Cloud GPU (row 30)				100.00	45.00		
Total before tax and shipping				4553.61	2696.92		
Sales tax (8.25%) and shipping, about 10%				455.36	269.69		

Near term: a self-placing camera or relay in disaster sites. After the Surfside building collapse in 2021, drones flew 304 missions; inside the standing part of the structure they held signal repeaters and fetched items, while ground robots were used about twice and were too large to work anywhere but the standing parts [57]. In the DARPA Subterranean Challenge, legged robots dropped WiFi relay nodes as they advanced [55]. A small flying humanoid could fill a narrow role between these: fly onto a ledge or the top of a debris pile in a standing structure, land on its independent legs, switch its fans off, and stay there as a camera or radio relay, or drop a 20 to 30 g beacon. Perching saves energy: a 32 g rotorcraft cut its power by about 40% perched on a ceiling and about 80% on a wall, for up to four times the mission time [46]. Perching hardware usually costs mass; one tether-perching pod added 508 g to a 1,008 g drone, which then had to stay perched at least 48.9% of the time to break even [49]. Here the legs are already the landing gear, so perching adds no mass. Flight near walls brings suction, vibration and collisions [59], none of which the simulator models. Rescue tasks such as the doors, valves, debris and stairs of the DARPA Robotics Challenge [43] need a much larger robot.

Next: power lines and structures. Aerial manipulation already works on power lines. In February 2023 ATC used a drone to install more than 450 bird diverters on 138 kV and 345 kV lines, replacing helicopters [40]. In early 2026 Red Eléctrica installed diverters by drone on a live 220 kV line in the Pyrenees to protect bearded vultures [50], and a Kauai co-op covered 740 spans by drone to protect endangered seabirds [48]. For contact inspection, the Voliro T tilt-rotor drone (3.7 kg, 6.0 kg maximum takeoff, 1.0 kg payload, 10 to 14 min of flight) holds up to 30 N against a structure for ultrasonic thickness and lightning-protection tests, and Voliro reports about 100 inspections a month in 17 countries [58]. This work is done today by drones of about 4 to 6 kg with 10 to 40 min of flight, and it needs tens of newtons of contact force and kilogram payloads. The robot in this report cannot do it. The direction we propose, as a design idea and not a result, is a scaled-up version that flies to the structure, lands on it with its legs, stops its fans, and works with two hands, so that flight time is spent only on getting there and back.

Long term: free-flyers in space. The project began from a question about a robot with thrust in its feet working in space. Free-flying robots already operate inside the ISS: NASA’s Astrobee (about 10 kg, fans feeding nozzles for 0.6 N of thrust, and an arm that grips handrails so it can switch its propulsion off) [39], JAXA’s Int-Ball2 (3 kg, propellers) [47] and DLR’s CIMON (5 kg, fans) [42]. Outside, servicing is moving to robots: DARPA’s RSGS servicer, with two seven-joint arms, launched on 21 July 2026 [52]; GITAI’s dual-arm robot completed its tasks outside the ISS in 2024 [45]; and Motiv’s xLink is to fly on NASA’s Fly Foundational Robots mission in late 2027 to demonstrate tool use and walking across spacecraft structures [44]. Orbital data centers add hardware that will need upkeep: Starcloud-1 launched with an Nvidia H100 in November 2025 [54], Axiom launched two data-center nodes on 11 January 2026 [41], and Google’s Project Suncatcher plans an 81-satellite cluster and lists on-orbit reliability as an open challenge [56].

Two limits follow. Fans do nothing in vacuum, so a space version would use cold-gas thrusters, as NASA’s SAFER jetpack (24 nitrogen thrusters of 3.6 N each) and the AERCam Sprint free-flyer (flown in 1997) did [38, 53]. Today’s servicers use stiff arms anchored to large spacecraft, and the data-center designs plan maintenance by resupply and module replacement [41] or self-assembling tiles [51], not dexterous free-flyers. What carries over from this project is the control problem: allocating thrust across a body whose limbs move, and flying on when one thruster weakens. The first step would be software experiments on existing free-flyer testbeds. We present this as a research direction, not a launch plan or a product.

10 Limitations

1. **Simulation only.** No part of the robot has been built. Every number in this report is a model output.
2. **Thin thrust margin in the feet.** The foot fans hover at 77% of their modelled maximum, total lift can rise only to 1.33 times the weight, and fast arm swings pushed a foot fan to 89%. If the real fan makes less thrust than modelled, the design as drawn may not fly with its arms moving.
3. **Unmeasured prop and duct.** C_T and C_Q come from a 5 in prop with 4.5 in pitch (Gemfan 5045) scaled to 102 mm. The foot prop in the parts list is a 4 in prop with 2.3 in pitch (Gemfan 4023), for which we have no data. The duct gain, rotor inertia, ESC delay and inlet blockage are estimates. The motor check fits the table it is compared with, and that table’s full-throttle point disagrees with its other rows, which matters because the foot fans run near full throttle.
4. **A 30% foot-fan loss is not survived.** The hand-written controller loses control 5.1 s after one. The residual policy trained on 10 to 25% faults and was tested only at 20%.
5. **The residual policy costs accuracy when healthy:** 5.24 against 4.37 cm RMS in cruise, with a 12.1 against 8.7 cm maximum.
6. **Single runs.** Each tour result is one deterministic run on one route with one sensor-noise seed, and the fault result is one fault size on one rotor at one time. We have no statistics over routes, seeds or faults. In a training-environment check the base controller survived all 14 faulted episodes (10 to 25% loss); it failed only in the full simulator, and we have not identified which of the full simulator’s added effects (Table 5) makes the difference.
7. **End-to-end RL is weaker than the hand-written controller.** Its best checkpoint drifted 9.2 against 1.5 cm, its sim-to-sim result got worse with more training, it only hovers, and the end-to-end mission runs were not rerun after the reset fix.
8. **Unmodelled effects.** Downwash interaction between the rotors, with the legs and arms, and with the floor below 0.5 rotor radii (ground effect is clamped there); changes in foot-fan inflow as the legs move; ducts in crossflow and forward flight; servo heating; gear backlash; wiring losses beyond the pack resistance, including the leads to the feet; prop and frame flexibility and vibration; grip forces and any grasped object.
9. **Checks not repeated on the final layout:** the backpack wake clearance (computed for V2), the hand-to-prop hazard metric (computed for a concept without a backpack), and the latency sweep (V1 only). The placement study used an earlier parameter set.
10. **Assumed sensing.** The simulator assumes 100 Hz pose tracking with 10 ms latency; the planned camera and AprilTag system is untested and may be slower or noisier.
11. **Masses.** Printed parts are solid PLA from CAD volume, the avionics mass is an estimate, and the base robot’s own mass comes from its model, not a scale.

Acknowledgements

The base robot is Microban by the Rhoban team [22] (hardware CC BY-NC-SA 4.0, software GPL v3). This project uses its MuJoCo model, meshes, ankle bracket geometry and servo identification [4, 7]; it is non-commercial, credits Rhoban, and will

be shared under the same terms. An AI coding agent wrote most of the code and most of the text of this report, including the simulator, the training environments and the figures, under the author's direction. The author made the design decisions, among them dropping the two-rotor layout, rejecting oversized ducts, and moving to a fan in each foot so the legs could move independently, and reviewed the results.

References

- [1] E. Ackerman, "Bipedal robot uses jet-powered feet to step over large gaps," *IEEE Spectrum*, Mar. 2018. <https://spectrum.ieee.org/bipedal-robot-uses-jetpowered-feet-to-step-over-large-gaps>
- [2] E. Ackerman, "Caltech's LEO flying biped can skateboard and slackline," *IEEE Spectrum*, Oct. 2021. <https://spectrum.ieee.org/bipedal-drone-robot-caltech>
- [3] Brax authors, `brax/envs/wrappers/training.py` (AutoResetWrapper), GitHub source file. <https://github.com/google/brax/blob/57ca9e12849644f5ac09d1313781b81836795720/brax/envs/wrappers/training.py>
- [4] M. Duclausaud and G. Passault, "BAM: Better Actuator Models," GitHub repository, 2024. <https://github.com/Rhoban/bam>
- [5] I. C. Cheeseman and W. E. Bennett, "The effect of the ground on a helicopter rotor in forward flight," *ARC R&M* 3021, 1955. <https://reports.aerade.cranfield.ac.uk/handle/1826.2/3590>
- [6] Chen et al., "Dynamics-informed reinforcement learning for agile and energy-efficient locomotion of a monopedal hopping quadcopter," arXiv:2609.15399, 2026. <https://arxiv.org/abs/2609.15399>
- [7] M. Duclausaud, G. Passault, V. Padois and O. Ly, "Extended friction models for the physics simulation of servo actuators," in *Proc. IEEE ICRA*, 2025. <https://arxiv.org/abs/2410.08650>
- [8] EMAX, "ECO II 2306 brushless motor," product page. <https://shop.emax-usa.com/products/eco-ii-2306-brushless-motor-1700kv-1900kv-2400kv> (The 16.8 V thrust table used here is not on this page; its values are reproduced in Fig. 4.)
- [9] T. Flash and N. Hogan, "The coordination of arm movements: an experimentally confirmed mathematical model," *J. Neuroscience*, vol. 5, no. 7, pp. 1688–1703, 1985. <https://doi.org/10.1523/JNEUROSCI.05-07-01688.1985>
- [10] P. Foehn et al., "Agilicious: open-source and open-hardware agile quadrotor for vision-based flight," *Science Robotics*, vol. 7, no. 67, 2022. <https://doi.org/10.1126/scirobotics.ab16259>
- [11] C. D. Freeman, E. Frey, A. Raichuk, S. Girgin, I. Mordatch and O. Bachem, "Brax: a differentiable physics engine for large scale rigid body simulation," *NeurIPS Datasets and Benchmarks*, 2021. <https://arxiv.org/abs/2106.13281>
- [12] H. Gorbani et al., "iRonCub 3: the jet-powered flying humanoid robot," arXiv:2506.01125, 2025. <https://arxiv.org/abs/2506.01125>
- [13] H. Gorbani, M. Elobaid, G. L'Erario, H. A. O. Mohamed and D. Pucci, "Data-fused MPC with guarantees: application to flying humanoid robots," arXiv:2509.10353, 2025. <https://arxiv.org/abs/2509.10353>
- [14] J. Hwangbo, I. Sa, R. Siegwart and M. Hutter, "Control of a quadrotor with reinforcement learning," *IEEE Robotics and Automation Letters*, vol. 2, no. 4, 2017. <https://arxiv.org/abs/1707.05110>
- [15] Istituto Italiano di Tecnologia, "IIT demonstrates that a humanoid robot can fly," press release, 18 June 2025. <https://opentalk.iit.it/en/iit-demonstrates-that-a-humanoid-robot-can-fly/>
- [16] T. Johannink et al., "Residual reinforcement learning for robot control," in *Proc. IEEE ICRA*, 2019, pp. 6023–6029. <https://arxiv.org/abs/1812.03201>
- [17] E. Kaufmann, L. Bauersfeld, A. Loquercio, M. Müller, V. Koltun and D. Scaramuzza, "Champion-level drone racing using deep reinforcement learning," *Nature*, vol. 620, pp. 982–987, 2023. <https://doi.org/10.1038/s41586-023-06419-4>
- [18] K. Kim, P. Spieler, E.-S. Lupu, A. Ramezani and S.-J. Chung, "A bipedal walking robot that can fly, slackline, and skateboard," *Science Robotics*, vol. 6, no. 59, 2021. <https://doi.org/10.1126/scirobotics.abf8136>
- [19] T. Lee, M. Leok and N. H. McClamroch, "Geometric tracking control of a quadrotor UAV on SE(3)," in *Proc. IEEE CDC*, 2010. <https://doi.org/10.1109/CDC.2010.5717652>
- [20] Y. Li et al., "Jet-HR2: a flying bipedal robot based on thrust vector control," *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 4590–4597, 2022. <https://arxiv.org/abs/2108.11557>
- [21] Li et al., "Learning energy-efficient air-ground actuation for hybrid robots on stair-like terrain," arXiv:2603.26687, 2026. <https://arxiv.org/abs/2603.26687>
- [22] M. Duclausaud (Rhoban team, LaBRI, University of Bordeaux), "Microban: A Compact, Fully 3D-Printable Open-Source Humanoid Robot," GitHub repository and Zenodo software record, 2026. <https://github.com/Rhoban/microban>, <https://doi.org/10.5281/zenodo.20784789>
- [23] MuJoCo contributors, "MuJoCo XLA (MJX)," MuJoCo documentation. <https://mujoco.readthedocs.io/en/stable/mjx.html>
- [24] A. Molchanov, T. Chen, W. Hönig, J. A. Preiss, N. Ayanian and G. S. Sukhatme, "Sim-to-(multi)-real: transfer of low-level robust control policies to multiple quadrotors," in *Proc. IEEE/RISJ IROS*, 2019. <https://arxiv.org/abs/1903.04628>
- [25] M. O'Connell et al., "Neural-Fly enables rapid learning for agile flight in strong winds," *Science Robotics*, vol. 7, no. 66, 2022. <https://doi.org/10.1126/scirobotics.abm6597>
- [26] A. Ollero, M. Tognon, A. Suarez, D. Lee and A. Franchi, "Past, present, and future of aerial robotic manipulators," *IEEE Transactions on Robotics*, vol. 38, no. 1, pp. 626–645, 2022. <https://doi.org/10.1109/RO.2021.3084395>
- [27] A. Paolino et al., "Learning aerodynamics for the control of flying humanoid robots," *Communications Engineering*, vol. 4, 2025. <https://doi.org/10.1038/s44172-025-00447-w>
- [28] J. L. Pereira, "Hover and wind-tunnel testing of shrouded rotors for improved micro air vehicle design," Ph.D. thesis, University of Maryland, 2008. <http://hdl.handle.net/1903/8752>
- [29] S. Pitroda et al., "Capture point control in thruster-assisted bipedal locomotion," arXiv:2406.14799, 2024. <https://arxiv.org/abs/2406.14799>
- [30] F. Ruggiero, V. Lippiello and A. Ollero, "Aerial manipulation: a literature review," *IEEE Robotics and Automation Letters*, vol. 3, no. 3, pp. 1957–1964, 2018. <https://doi.org/10.1109/LRA.2018.2808541>
- [31] P. Sanchez-Cuevas, G. Heredia and A. Ollero, "Characterization of the aerodynamic ground effect and its influence in multirotor control," *International Journal of Aerospace Engineering*, 2017, Art. 1823056. <https://doi.org/10.1155/2017/1823056>
- [32] J. Schulman, F. Wolski, P. Dhariwal, A. Radford and O. Klimov, "Proximal policy optimization algorithms," arXiv:1707.06347, 2017. <https://arxiv.org/abs/1707.06347>
- [33] T. Silver, K. Allen, J. Tenenbaum and L. Kaelbling, "Residual policy learning," arXiv:1812.06298, 2018. <https://arxiv.org/abs/1812.06298>
- [34] K. Sugihara, M. Zhao, T. Nishio, T. Makabe, K. Okada and M. Inaba, "Design and control of a small humanoid equipped with flight unit and wheels for multimodal locomotion," *IEEE Robotics and Automation Letters*, vol. 8, no. 9, 2023. <https://arxiv.org/abs/2303.14718>
- [35] K. Sugihara and K. Okada, "Anti-gravity walking by a flying humanoid robot via thrust-rate input whole-body model predictive control," *IEEE Robotics and Automation Letters*, 2026. <https://arxiv.org/abs/2609.07544>
- [36] J. Tobin, R. Fong, A. Ray, J. Schneider, W. Zaremba and P. Abbeel, "Domain randomization for transferring deep neural networks from simulation to the real world," in *Proc. IEEE/RISJ IROS*, 2017, pp. 23–30. <https://arxiv.org/abs/1703.06907>
- [37] ROBOTIS, "XL330-M288-T," DYNAMIXEL e-manual. <https://emanual.robotis.com/docs/en/dxl/x/xl330-m288/>
- [38] J. Wagenknecht, S. Fredrickson, T. Manning and B. Jones, "Design, development and testing of the Miniature Autonomous Extravehicular

- Robotic Camera (Mini AERCam) guidance, navigation and control system,” NASA technical report, 2003. <https://ntrs.nasa.gov/api/citations/20150017436/downloads/20150017436.pdf>
- [39] O. Alexandrov et al., “Astrobee: free-flying robots for the International Space Station,” NASA technical report, 2026. https://ntrs.nasa.gov/api/citations/20260001396/downloads/astrobee_IEEE_TFR_2026_compressed.pdf
- [40] American Transmission Co., “ATC uses drone to install bird diverters for the first time,” 2023. <http://www.atc11c.com/atc-uses-drone-to-install-bird-diverters-for-the-first-time>
- [41] Axiom Space, “Orbital data centers.” <https://www.axiomspace.com/orbital-data-center>
- [42] DLR, “CIMON: astronaut assistance system.” <https://www.dlr.de/en/research-and-transfer/projects-and-missions/horizons/cimon>
- [43] M. Rosen, “Designing robots to help in a disaster,” *Science News*, Dec. 2014. <https://www.sciencenews.org/article/designing-robots-help-disaster>
- [44] NASA, “Fly Foundational Robots.” <https://www.nasa.gov/mission/fly-foundational-robots/>
- [45] GITAI, “GITAI completes fully successful technology demonstration outside the ISS,” Mar. 2024. <https://gitai.tech/2024/03/19/gitai-completes-fully-successful-technology-demonstration-outside-the-iss/>
- [46] Y.-H. Hsiao et al., “Energy efficient perching and takeoff of a miniature rotorcraft,” *Communications Engineering*, vol. 2, 2023. <https://pmc.ncbi.nlm.nih.gov/articles/PMC10956013/>
- [47] JAXA, “Int-Ball2” (in Japanese). <https://www.kenkai.jaxa.jp/research/automation/intball2.html>
- [48] NRECA, “Hawaii co-op among the first to use drones to install bird diverters,” Jan. 2022. <https://www.electric.coop/hawaii-co-op-among-the-first-to-use-drones-to-install-bird-diverters>
- [49] T. Lan, L. Romanello, M. Kovac, S. F. Armanini et al., “Aerial tensile perching and disentangling mechanism for long-term environmental monitoring,” arXiv:2403.01890, 2024. <https://arxiv.org/abs/2403.01890>
- [50] Red Eléctrica, “Red Eléctrica installs bird-flight diverters using drones on power lines in the Huesca Pyrenees,” press release, Feb. 2026. <https://www.ree.es/en/press-office/press-release/new-s/press-release/2026/02/Red-electrica-installs-bird-flight-diverters-using-drones-on-power-lines-Huesca-Pyrenees>
- [51] Yahoo Tech, “Self-assembling data centers in space are becoming reality as Rendezvous Robotics partners with Starcloud,” Nov. 2025. <https://tech.yahoo.com/science/articles/self-assembling-data-centers-space-155514920.html>
- [52] DARPA, “Robotic Servicing of Geosynchronous Satellites lifts off,” 2026. <https://www.darpa.mil/news/2026/robotic-servicing-of-geosynchronous-satellites-lifts-off>
- [53] T. W. Williams, “Techniques for improving the performance of future EVA maneuvering systems,” NASA technical report, 1995. <https://ntrs.nasa.gov/api/citations/19960022260/downloads/19960022260.pdf>
- [54] Starcloud, “Starcloud-1.” <https://www.starcloud.com/starcloud-1>
- [55] M. Tranzatto et al., “CERBERUS: autonomous legged and aerial robotic exploration in the Tunnel and Urban circuits of the DARPA Subterranean Challenge,” arXiv:2201.07067, 2022. <https://arxiv.org/abs/2201.07067>
- [56] Google Research, “Exploring a space-based, scalable AI infrastructure system design.” <https://research.google/blog/exploring-a-space-based-scalable-ai-infrastructure-system-design/>
- [57] R. R. Murphy, “How robots helped out after the Surfside condo collapse,” *IEEE Spectrum*, Aug. 2021. <https://spectrum.ieee.org/building-collapse-surfside-robots>
- [58] Voliro, “Voliro T specification sheet v3.2,” 2025, and <https://voliro.com/>. <https://cdn.voliro.com/wp-content/uploads/2025/06/Voliro-T-Spec-sheet-v3.2.pdf>
- [59] P. Yang, W. Wen, R. Yang, Y. Chen et al., “Wall Inspector: quadrotor control in wall-proximity through model compensation,” arXiv:2509.21496, 2025. <https://arxiv.org/abs/2509.21496>