Fingers as Legs: Learning Self-Supported Locomotion and Manipulation with an Anthropomorphic Hand

Amirhossein Kazemipour, Hehui Zheng, and Robert Katzschmann

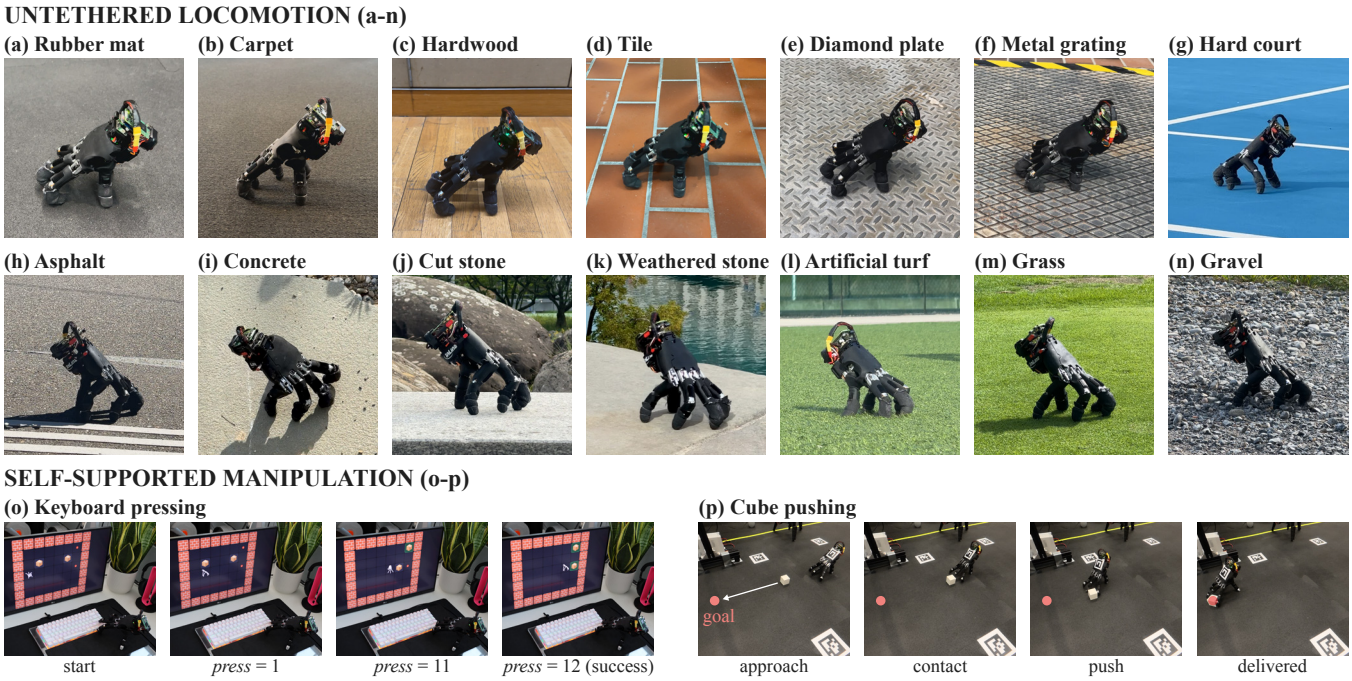

Fig. 1: The hand combines untethered locomotion with self-supported interaction. Top: locomotion demonstrated on 14 surfaces: (a) rubber mat, (b) carpet, (c) hardwood, (d) tile, (e) diamond plate, (f) metal grating, (g) hard court, (h) asphalt, (i) dry concrete, (j) cut stone, (k) weathered stone, (l) artificial turf, (m) grass, and (n) gravel. Bottom: self-supported manipulation: (o) 12 keyboard presses completing a Sokoban level; (p) object pushing from approach through delivery.

***Abstract*—A walking robotic hand must use the same fingers to move its body, support its weight, and interact with the environment. We show how an anthropomorphic hand can learn these skills while retaining its finger design and position controller. Onboard power and computation make the platform self-contained. Our reinforcement learning approach accounts for the hand's unequal fingers, with training in a simulator calibrated from hardware measurements. In simulation, the hand moves faster with our reward formulation than with tuned rewards originally designed for quadrupeds. On hardware, task-specific policies enable untethered crawling, steering, and fall recovery. While supporting its own weight, the hand also executes successive keyboard commands without vision and pushes an object to targets using overhead visual feedback. These results demonstrate a compact mobile manipulator that reuses its fingers for locomotion and interaction, without a separate locomotion mechanism.**

***Index Terms*—Mobile manipulation, loco-manipulation, legged robots, multifingered hands, reinforcement learning.**

I. Introduction

A robotic hand with its own mobility could operate in confined workspaces without requiring the arm that normally carries it to follow. For example, a larger robot could place the hand on a support surface near a restricted opening. The hand could then move toward a control or object, interact with it, and return for retrieval. Using its fingers as legs, like Thing in *The Addams Family*, would avoid a separate locomotion mechanism. We study these capabilities on an off-the-shelf WUJI right hand [1] without modifying its fingers or its built-in position controller.

Realizing this idea requires the same fingers to move the body, support its weight, and interact with the environment. When a finger lifts to step or press a key, the remaining contacts must keep the hand balanced. Pushing an object likewise requires maintaining support during task contact. The opposed thumb and unequal fingers in Fig. 1 complicate this coordination because each fingertip has a different reach. The palm also rests at a tilt, so its body frame is not a level reference for commanded motion.

Our approach has three parts. The locomotion reward is built around the hand's own stance: a penalty pulls each fingertip toward its nominal stance position, commands are expressed in a control frame that removes the nominal palm tilt, and the policy decides where and when each finger steps. Each task has its own policy, trained in a simulator calibrated to measured fingertip friction and to the hand's response to filtered position commands. Onboard power and computation enable untethered

All authors are with the Soft Robotics Lab, ETH Zurich, Zurich, Switzerland. {akazemi, zhengh, rkk}@ethz.ch

operation. We also evaluate self-supported keyboard pressing without vision and vision-guided object pushing.

Our contributions are threefold:

- **System:** An 818 g mobile manipulator built from a commercial anthropomorphic hand, retaining its finger kinematics and position controller. It carries its own battery and computer, runs untethered, and uses the same fingers to walk, to support itself, and to interact.
- **Method:** A locomotion reward built around the hand's own stance. A footprint objective pulls each of the unequal fingers toward its own nominal stance position, while the policy learns where and when each finger steps.
- **Evaluation:** In simulation, the hand moves faster with our reward than with quadruped rewards adapted to it. On hardware, task-specific policies crawl untethered on 14 surfaces, steer, recover from falls, press successive keyboard keys without vision, and push an object to targets with overhead visual feedback.

II. Related Work

a) Mobile hands and finger-limbed robots: Prior work has explored finger symmetry, modularity, and body reconfiguration to enable mobility. DLR-Crawler uses six identical hand-finger modules as hexapod legs [2]. Gao *et al.* combine a purpose-built symmetric, reversible hand with cyclic gaits optimized by a genetic algorithm and vision-guided manipulation [3]. Hand-shaped avatars have been studied for learned locomotion [4] and detachment from humanoids [5]. Other designs reconfigure hands into humanoids [6] or use origami digits for grasping and crawling [7]. Reinforcement learning also coordinates locomotion and manipulation with reconfigurable limbs [8]. We instead keep the unequal finger geometry of a commercial anthropomorphic hand and add untethered locomotion and self-supported manipulation.

b) Manipulation with legs: A leg used for manipulation is no longer available for support. Robots use legs to press buttons, open doors, and move objects [9], [10], [11], [12]. Whole-body policies coordinate locomotion and manipulation [13]. Hierarchical navigation systems separate learned skills, perception, and planning [14]. Our hand must likewise balance interaction and support, with its fingers providing all body support.

c) Morphology and reward design: Locomotion learning often exploits a robot's left–right symmetry through mirror losses, data augmentation, or equivariant policies [15], [16], [17]. These methods need a mirroring of states and actions that leaves the reward unchanged [17]. A hand with an opposed thumb and four unequal fingers has no such mirror.

Gait rewards for legged robots often prescribe timing. Periodic reward composition assigns each leg swing and stance intervals and relative phases [18], and Margolis and Agrawal [19] (Walk These Ways) reward a parameterized contact schedule and Raibert foot-position targets. Our footprint objective anchors place rather than time: each fingertip is pulled toward its own nominal position, captured from the settled stance and expressed in a body frame that is level at that stance, with a weaker pull along the direction of travel so that steps are cheap. The lift objective only encourages a stepping rate that grows with the commanded speed; which finger steps when, and how far, is left to the policy.

d) Sim-to-real transfer: Sim-to-real locomotion relies on modeling actuator dynamics, observation noise, latency, and contact variation [20], [21], [22]. For our position-controlled hand, hardware measurements inform the actuation and contact models.

III. System and Common Learning Framework

A. Self-contained hand platform

We equip a 738 g off-the-shelf WUJI right hand for untethered operation while retaining its finger kinematics, factory controller gains, and vendor-provided position-control interface. The hand has 20 actuated joints, four per finger. The joints are non-backdrivable. We configure the firmware's position-command low-pass filter to a 3 Hz cutoff. Each joint is limited to 1.0 A during the experiments.

The dorsal module in Fig. 2(a) provides onboard power, sensing, and computation, bringing the robot's mass to 818 g. A ROS 2 stack runs a 500 Hz serial driver and policy inference at a nominal 50 Hz. The actor and observation normalizer execute on the Pi as a 32-bit floating-point ONNX model. Crawl commands arrive over a dedicated 2.4 GHz gamepad link, with Wi-Fi outside the locomotion control loop. A safety supervisor monitors communication, electrical limits, and attitude.

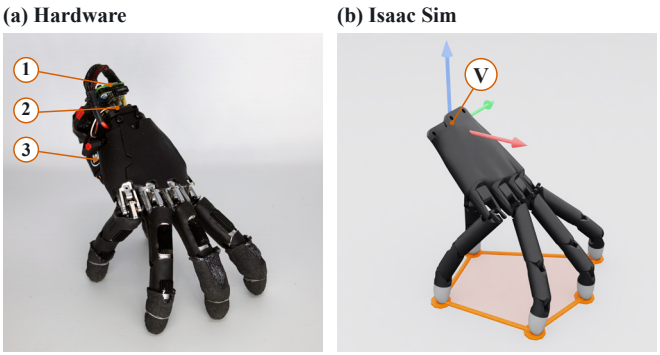

Fig. 2: Onboard power, sensing, and computation make the commercial hand self-contained. (a) Hardware with (1) a Raspberry Pi Zero 2 W, (2) a BNO085 IMU, and (3) a four-cell lithium-polymer battery. The dorsal module adds 80 g to the 738 g hand. (b) Corresponding NVIDIA Isaac Sim model, showing the stance-calibrated control frame V (colored axes) and the support polygon formed by the five fingertip contacts.

B. Policy interfaces

Each policy is a feedforward network, trained with PPO in simulation (Section IV) and run onboard at 50 Hz. It maps a short history of proprioceptive inputs, plus the task inputs in Table I, to an increment of the joint position targets that the hand's built-in position controller then tracks.

All policies receive 46 common inputs at each control step: each joint's angle relative to its fixed reference angle (20, rad), a unit vector indicating the direction of gravity (3, dimensionless),

TABLE I: Task-specific information available to each actor. Dimensions are in parentheses. V is each task's stance-calibrated frame (Section IV-B). Positions are in metres and velocities in m/s.

Policy	Task-specific information
Crawl / recovery	Planar command in V (2), yaw-rate command in rad/s (1)
Keyboard	Requested-key indicator (4), pressing fingertip position in V (3)
Pushing	Root-relative object position (3), world object velocity (3), object-to-target vector (3), all expressed in V

angular velocity (3, rad/s), and the previous policy output (20, dimensionless). The gravity vector and angular velocity are expressed in the palm frame. The previous output is the clipped policy output a_t defined below. Table I summarizes the additional task-specific inputs. Each observation term retains eight samples, oldest to newest, before the term histories are concatenated and normalized using fixed training statistics. Motor current serves only the safety supervisor.

At each control step, the controller adds the joint-target increment $\Delta q_t^{\mathrm{tar}} = \alpha_q \tanh(a_t)$ to the previous target and enforces joint limits. Here, a_t is the clipped policy output and α_q is the task-specific angle scale: 0.060, 0.028, and 0.040 rad for crawl/recovery, keyboard, and pushing, respectively. Simulation includes a one-step command delay and a 3 Hz low-pass filter. Deployment uses the same observation and action transforms, with additional hardware safety limits.

C. Hardware-calibrated simulation

Policies act through the hand's retained position controller. We therefore calibrate the simulator against the measured response to its filtered position targets. Frequency sweeps of the proximal and distal interphalangeal joints, loaded fingertip pulls, and timed command responses identify stiffness, fingertip friction, delay, filtered joint speed, and filter cutoff (Table II).

IV. Morphology-Adapted Locomotion

A. Learning problem and optimization

The crawl policy learns to follow planar-velocity and yaw-rate commands while supporting the hand on its fingertips. Commands are sampled with $v_x \in [0, 0.16]$ and $v_y \in [-0.16, 0.16]$ m/s and $\omega_z \in [-0.45, 0.45]$ rad/s. Episodes terminate upon palm contact, roll beyond 35°, pitch beyond 30°, or timeout. Contact by any other non-fingertip link incurs a penalty.

We train separate actor and critic networks using proximal policy optimization (PPO) [23] in NVIDIA Isaac Lab [24]. Both are multilayer perceptrons with exponential linear unit (ELU) activations. Table III lists the actor architecture and training settings. The actor receives the observations available on hardware (Section III). During training, the critic also receives the base state, joint torque, and fingertip contact force. Physics runs at 200 Hz, with one policy action every four physics steps. Each PPO update uses 24 steps per environment, five epochs, and four minibatches, with a clip ratio of 0.2.

TABLE II: Hardware measurements used to parameterize the simulator. Joint stiffness is expressed relative to the nominal, uncalibrated model.

Quantity	Estimate	Procedure
Joint stiffness scale	15–22× (18.7±2.5×)	joint frequency sweeps
Kinetic fingertip friction μ_k	0.88 (range 0.80–0.97)	three loaded fingertip pulls
Closed-loop delay	≈19 ms	timed command response
Filtered joint speed	2.8 / 2.5 rad/s	flexion / abduction ramps
Command filter cutoff	3 Hz	trajectory comparison

TABLE III: Training and objective settings for the deployed crawl policy (SIM). Here and in later captions, SIM marks simulation results and HW marks hardware results. The yaw coefficient is introduced by curriculum. CoM denotes center of mass.

Setting	Value
Parallel environments / episode	4096 / 20 s
Actor hidden layers	512, 256, 128 (ELU)
Actor input / output	392 / 20
Control rate	50 Hz
PPO iterations	8,000 + 5,000 payload/contact adaptation
Planar tracking / yaw λ_v/λ_ω	$+1.0$ / 0, then $+0.5$
Objective weights $\lambda_{\mathrm{fp}}/\lambda_{\mathrm{lift}}(k)/\lambda_{\mathrm{dir}}$	-30 / $+6.25 \rightarrow +0.625$ / -0.5
Undesired contact	-1.0
v_z / ω_{xy} / action rate	-2.0 / -0.05 / -0.01
Torque / joint acceleration	-2.5×10^{-5} / -2.5×10^{-7}
Tip friction / gain scale	$[0.30, 1.30]$ / $[0.78, 1.22]$
Payload CoM / effort scale	±10/15 mm at 80 g / $[1.0, 1.5]$

An adaptive learning rate starts at 10^{-3}. We use $\gamma = 0.99$, generalized advantage estimation with $\lambda = 0.95$, and an entropy coefficient of 0.005.

Further training adapts the crawl policy to the dorsal module's added load and contact conditions. Randomization covers fingertip friction, effort scale, and the payload's horizontal and vertical center of mass offsets (Table III), as well as palm mass, palm center of mass, and IMU bias. The actuator gains vary around the calibrated stiffness multipliers (Table II).

B. Stance-calibrated fingertip objectives

We express fingertip motion in a frame aligned to the nominal stance and assign each unequal finger its own target. The footprint objective supplies this geometric reference. Lift and direction objectives add frequency and swing-direction shaping during training. The policy learns each finger's timing (Fig. 3).

a) Stance-calibrated control frame: We obtain the reference stance by holding fixed joint targets in simulation for 4 s with the 80 g payload, then saving the settled root pose and joint positions. Let $q_B(t)$ be the root orientation and q_{off} the fixed rotation that levels this stance and aligns forward with the horizontal vector from the palm to the centroid of the index–little fingers' distal-link origins. We define $q_V(t) = q_B(t) \otimes q_{\mathrm{off}}$. Frame V, illustrated in Fig. 2(b), is level and heading-aligned in the nominal stance. It then follows the root rotation rather than remaining gravity-leveled. Expressing commands and base-relative fingertip kinematics in V removes the nominal palm tilt without suppressing body motion.

b) Footprint objective: Each fingertip is referenced to its own nominal position to accommodate the hand's unequal stance geometry. For fingertip j, p_j is its base-relative position

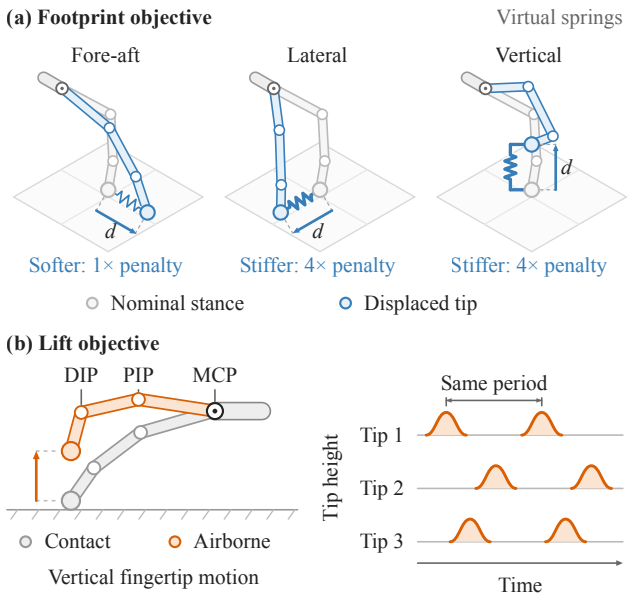

Fig. 3: The footprint objective penalizes lateral and vertical departures four times as strongly as fore–aft motion. (a) Virtual springs around nominal fingertip targets illustrate this ratio for equal displacements. (b) Airborne lifting and illustrative height traces with the same stepping frequency but different fingertip timing. MCP, PIP, and DIP denote the metacarpophalangeal, proximal interphalangeal, and distal interphalangeal joints.

in V. Each environment captures p_j^* at its first reward evaluation after initialization and holds it fixed across subsequent resets. With $W = \mathrm{diag}(0.25, 1, 1)$, the reward feature is

$$\bar{r}_{\mathrm{fp}} = \sum_{j=1}^{5} (p_j - p_j^*)^\top W (p_j - p_j^*). \tag{1}$$

It contributes $\lambda_{\mathrm{fp}} \bar{r}_{\mathrm{fp}}$ with $\lambda_{\mathrm{fp}} = -30$. The penalty has the form of the energy stored in virtual springs around the stance targets (Fig. 3(a)): softer fore–aft springs leave room for stepping, while stiffer lateral and vertical springs discourage deviations. The stiffnesses are fixed reward weights, not controller gains; Section VI-A sweeps λ_{fp}. The targets move with the body without prescribing ground locations or a footfall sequence.

c) Auxiliary lift objective: For commanded planar velocity v_{cmd} and vertical fingertip velocity u_j relative to the rotating base in V, the lift objective favors a command-dependent stepping frequency (Fig. 3(b)):

$$\begin{aligned} \phi_t &= \phi_{t-1} + 2\pi f(\|v_{\mathrm{cmd}}\|)\Delta t, \\ z_{j,t} &= (1-\alpha_\tau) z_{j,t-1} + \alpha_\tau u_{j,t} e^{-\mathrm{i}\phi_t}, \\ \bar{r}_{\mathrm{lift},t} &= \sum_{j=1}^{5} g_{j,t} z_{\max} S(z_{\min}, z_{\max}, |z_{j,t}|). \end{aligned} \tag{2}$$

The complex exponential moving average uses $\alpha_\tau = \Delta t/(\Delta t + \tau)$, with control interval $\Delta t = 20\,\mathrm{ms}$ and $\tau = 0.3\,\mathrm{s}$. The phase ϕ is kept in $[0, 2\pi)$; each wrap marks the start of a new stepping cycle. Phase, moving average, and contact history reset each episode and at command resampling. The actor does not observe ϕ. Frequency f linearly maps 0.02 m/s to 0.16 m/s to 1 Hz to 3.5 Hz, with endpoint clamping. $S(z_{\min}, z_{\max}, \cdot)$ is a cubic smoothstep that rises from 0 at $z_{\min} = 0.005$ to 1 at $z_{\max} = 0.015\,\mathrm{m/s}$.

Gate $g_{j,t}$ requires an airborne tip (force below $F_{\mathrm{th}} = 0.15\,\mathrm{N}$), contact at or above F_{th} since the latest phase wrap, and commanded planar speed at least $v_{\mathrm{dead}} = 0.02\,\mathrm{m/s}$. The lift weight $\lambda_{\mathrm{lift}}(k)$ is 6.25 for the first 18,000 per-environment steps (750 of the 8,000 PPO iterations), then decays geometrically to 0.625 over the next 36,000 steps and stays there.

d) Auxiliary direction objective: For fingertip velocity relative to the rotating base, we penalize its planar component opposite to the command direction, only when tip force is below F_{th} and commanded planar speed is at least v_{dead}. This leaves planted fingertips free to move backward relative to the base while propelling it forward.

The full reward formulation also tracks planar velocity and, once a forward gait has formed, adds yaw-rate tracking at the same 18,000-step point, so that the policy does not learn stepping and turning at once. It penalizes undesired contacts, vertical body motion, roll and pitch rates, joint effort and acceleration, and changes in action. The planar- and yaw-tracking kernels use widths of 0.10 m/s and 0.225 rad/s, respectively. Table III lists all coefficients.

C. Fall recovery

Fall recovery lets the hand resume crawling without a person setting it upright. A separate recovery policy is trained with the same PPO setup; each 20 s episode starts with the hand lying on its side at a random heading, and its reward penalizes palm tilt away from level and rewards reaching the crawl-stance height and joint pose, with no early termination. Once upright, the policy keeps the hand balanced through continuous joint motion. A smooth joint-target ramp then brings the hand to rest in the static crawl stance over 1.5 s. On hardware, an upright-state detector starts this ramp when stance-relative tilt stays below 10° and angular speed below 4 rad/s for 0.5 s. In simulation, the complete procedure rights the hand in 28 of 32 simulated falls, with a median time of 6.1 s and a mean absolute joint error of 0.004 rad relative to the crawl stance; in the remaining four episodes the upright detector did not fire within the 20 s.

V. Self-Supported Manipulation Policies

We test whether the hand can operate a control at a known location and guide an object using visual feedback. Separate keyboard-pressing and object-pushing policies retain the calibrated simulator, proprioceptive history, support requirements, and PPO implementation. Both use normalized observations and ELU activations, with an actor of three hidden layers (512, 256, and 128 units) and a critic of three hidden layers (256, 128, and 64 units) that also receives privileged simulation state during training.

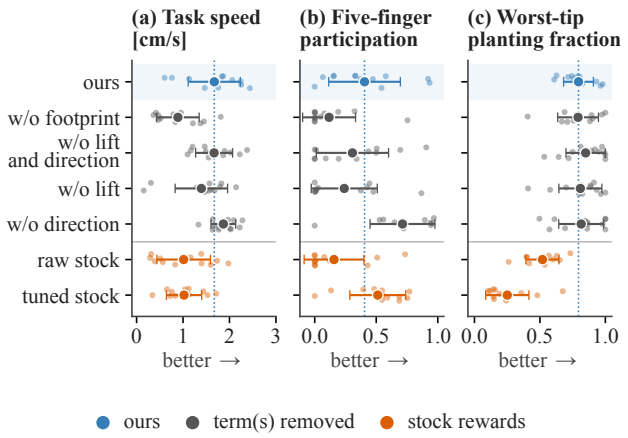

Fig. 4: Reward-term ablation and stock-reward comparison: the footprint objective improves task speed and five-finger participation (SIM). Small dots show seed means. Large markers and bars show mean ± std. over twelve training seeds, each evaluated on 256 domain-randomized episodes. Dotted lines mark our formulation.

A. Keyboard pressing without vision

A key identifier, one-hot during commands and zero while idle, selects among four learned nominal press locations. Before each evaluation block, an operator aligns the keyboard to these locations using trial presses. Encoder-based forward kinematics gives the pressing fingertip position in palm-attached frame V but does not track hand translation relative to the keyboard. Successive commands rely on maintained alignment. Misalignment requires manual realignment. Rewards encourage approach, increasing press depth, requested-key actuation, and support from other fingertips. Penalties discourage wrong-key presses, keycap contact by other hand parts, and loss of stance. Physical keyboard USB events score presses but are not actor inputs. Latency runs from command issue to recorded outcome, including event transport and processing.

B. Vision-guided object pushing

An overhead camera tracks a dorsal marker and the object to construct the pushing observations in Table I. The target remains fixed in the world after each trial begins. Training uses a 60 Hz sample-and-hold camera model with a latency of one to three control steps, 3% dropout, and 2 mm position noise. The policy learns approach, contact, and pushing together.

The task rewards approaching the object, moving it toward the target, and keeping it there, while retaining the support and regularization terms.

VI. Experiments

We first test the reward formulation in simulation, then evaluate how the untethered hand moves, recovers its stance, and interacts with its surroundings.

A. Reward-term ablation in simulation

We evaluate task speed, five-finger participation, and contact posture. Participation can include pad-side or nail-side contact,

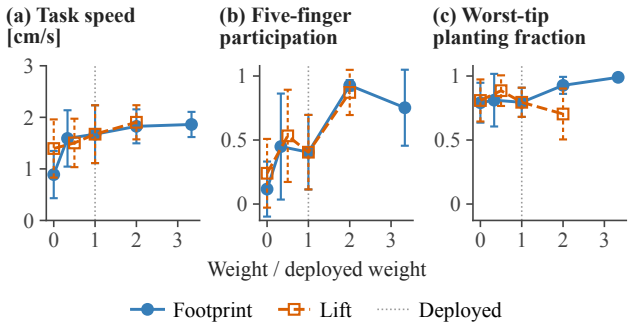

Fig. 5: Footprint and lift weights change finger participation and contact posture differently (SIM). Markers and bars show mean ± std. across twelve training seeds. Weights are shown as multiples of their deployed values. The dotted line marks the deployed weight.

so we interpret it alongside posture. We compare our formulation with quadruped rewards adapted to the hand, then remove terms and vary their weights to identify their contributions.

a) Design: Only rewards differ across configurations. The hand model, initial conditions, action interface, observations, curriculum, and domain randomization are fixed. We train each configuration on the same twelve seeds with 4096 environments for 8,000 crawl iterations. Each policy is then evaluated in 256 flat-ground episodes with one evaluation seed. Only our formulation was tested on hardware, after additional adaptation (Table III).

The seven configurations are our formulation (Section IV), four variants removing the footprint, lift, direction, or both lift and direction objectives, and two stock baselines. *Raw stock* adapts the rewards from Isaac Lab's ANYmal-D flat-ground velocity-tracking task [24] to the hand, retaining the published weights (feet-air-time $+0.5$, flat-orientation -5). Both baselines compute the flat-orientation penalty as the sum of squared horizontal components of unit gravity in the stance-calibrated frame V. It is zero at the nominal crawl stance and independent of heading. For *tuned stock*, we screen 24 random reward-weight settings on two seeds for task speed, then evaluate the top three on six fresh seeds. The selected configuration sets feet-air-time to 2.0 and halves the flat-orientation weight, rescales the remaining regularization penalties, and removes the foot-slide and undesired-contact penalties. Six of its twelve reported seeds were also used to select this configuration. Our weights were fixed by the hardware campaign before the ablation and were not retuned.

b) Measures: Task speed is progress in the commanded direction, averaged over each episode and the training command distribution under domain randomization. *Five-finger participation* is the fraction of episodes in which every fingertip touches down at least three times and spends at least 5% of the episode in contact.

We distinguish tip-side from nail-side contact using the pad axis's tilt above horizontal toward the nail. Lower tilt indicates more pad-side contact. We average tilt over time steps in contact and report the worst fingertip and the mean across

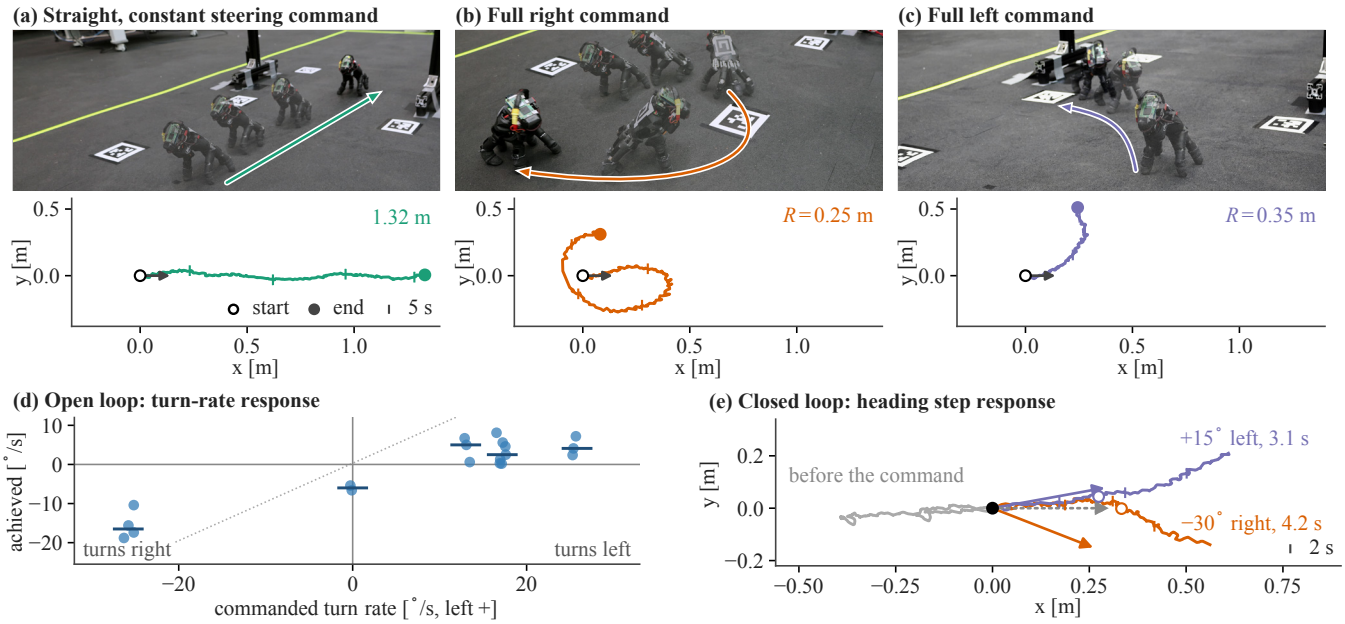

Fig. 6: The hand turns in both directions with asymmetric rates. IMU feedback enables commanded heading changes (HW). (a–c) Open-loop snapshots and paths, aligned to a common origin: (a) corrected straight motion, (b) the full right-turn command, and (c) the full left-turn command. (d) Achieved versus commanded turn rate: horizontal bars show medians and the dotted diagonal marks achieved = commanded. (e) Heading changes measured by overhead tracking. The black dot marks the first command. Grey paths precede it. Dotted and solid arrows show the previous and commanded headings. Circles mark the start of a 1 s interval with the stride-averaged heading within $\pm 5^\circ$ of the target. Labeled times are measured from the final command increment.

fingertips. The *worst-tip planting fraction* is the fraction of contact time below 64° for the fingertip with the largest fraction of nail-side contact. Contacts are reported per fingertip, not per side, so tilt serves as the classifier. The *non-nail participation* variant counts only contacts below this threshold toward the 5% contact-time requirement.

c) Comparison with stock rewards: In Fig. 4(a), the hand moves faster with our formulation than with either tested stock baseline. Compared with tuned stock, our formulation is 0.65 cm/s faster on average (95% confidence interval [+0.26, +1.02] across twelve seeds) and faster in 10 of 12 seeds.

Speed alone would hide how the fingers touch the ground (Fig. 4(b) and (c)). Tuned stock reaches a higher mean five-finger participation, although that difference is uncertain, but it plants fingers on their nails more in every seed. Our formulation has higher non-nail participation (+0.35 [+0.18, +0.53]), showing why participation and contact posture must be considered together.

d) Contribution of individual objectives: Among the tested reward components, the footprint objective provides the clearest supported gains in task speed and five-finger participation. Removing it reduces task speed by 0.79 cm/s [+0.37, +1.19] and lowers five-finger participation (Fig. 4(a) and (b)). At the deployed weight, this objective shifts mean contact tilt across fingertips 8.8° toward the nail side ([+1.4, +15.9]). The footprint objective therefore improves five-finger participation without improving every aspect of contact posture.

The ablations do not establish an independent speed benefit from the auxiliary lift or direction objectives, whether removed separately or together. Removing the direction objective alone increases five-finger participation by 0.31 [+0.07, +0.52], while its contact-posture effect remains uncertain. Six additional seed-matched comparisons at doubled footprint weight also leave the speed and contact effects of direction removal uncertain. These tests do not establish a benefit from the direction objective. We retain it in the reported reward formulation because the hardware policies were trained with it.

e) Contact posture and reward weights: Doubling footprint or lift weight raises five-finger participation but affects contact posture differently (Fig. 5(b) and (c)). Doubling footprint weight gives a higher worst-tip planting fraction than doubling lift weight (paired difference +0.22 [+0.11, +0.33], 10 of 12 seeds). Their speed difference in Fig. 5(a) is uncertain (−0.08 cm/s [−0.39, +0.21], footprint minus lift).

Above the deployed footprint weight, the worst-tip planting fraction keeps improving while task speed changes remain uncertain; mean five-finger participation peaks at twice the deployed weight and decreases again by 3.3 times. Higher weights were tested only in simulation. Taken together, the measured gains in speed and five-finger participation come from the footprint objective, while the lift and direction objectives shape stepping rhythm and swing direction during training.

B. Untethered crawling and steering

The hand crawled untethered under gamepad control across the 14 indoor and outdoor surfaces in Fig. 1(a)–(n), from

(a) $t = 0$ s **(b)** $t = 1$ s **(c)** $t = 3$ s **(d)** $t = 4$ s **(e)** Crawl

Fig. 7: The hand returns to its crawl stance after a fall (HW). (a–d) Learned righting. (e) Controller settling.

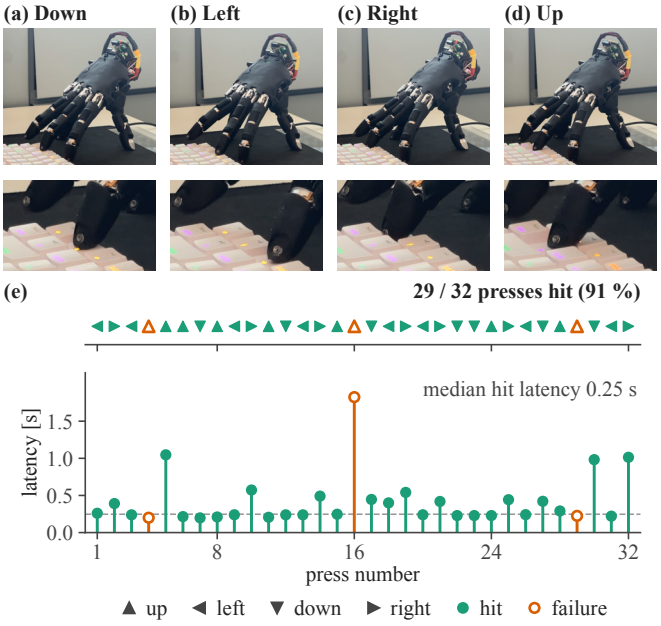

Fig. 8: The hand presses four arrow keys while supporting itself (HW). (a–d) Whole-hand views and fingertip close-ups of Down, Left, Right, and Up presses. (e) Outcomes (top) and command-to-keystroke latency (bottom) in the 32-command sequence. Marker shapes identify commands. Filled green markers show correct presses (hits), and open orange markers show failures. The dashed line marks median latency for correct presses.

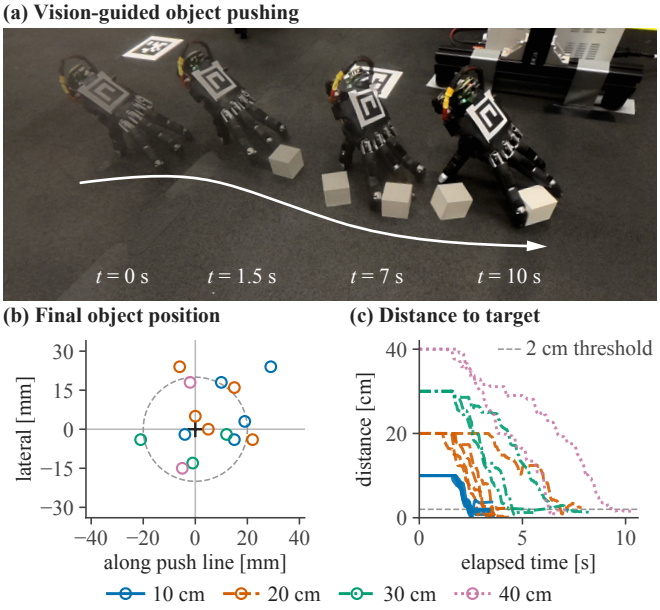

Fig. 9: One policy approaches and pushes a cube to targets using overhead visual feedback (HW). (a) A delivery sequence. (b–c) Final target-relative positions and distance-to-target traces for 15 deliveries, colored by target distance. Trace time starts at the first recorded control sample; final positions are the last sample, after the 1 s dwell, during which the hand keeps pushing. Delivery requires entering the 2 cm radius (dashed guides), then remaining within 5 cm for 1 s.

smooth flooring to metal grating, grass, and gravel; these runs are a qualitative demonstration, shown in the supplementary video. We evaluated steering with one policy on a 1.3 m mat, using an overhead camera for measurement only.

a) Open loop: Steering changes the hand's path, but the response differs between right and left turns. Without a steering command, the hand drifts right by about $6^\circ\,\mathrm{s}^{-1}$ regardless of its initial heading. A constant correction produces nearly straight travel in Fig. 6(a). In Fig. 6(b)–(d), right-turn rate scales with the command, while left-turn rate plateaus. Across 21 trajectories, mean path speed was 0.093 m/s.

b) Closed loop: To close the loop, a proportional controller on the onboard IMU heading sets the crawl policy's yaw-rate command, adding a constant offset that cancels the drift and keeping the command within the training range. Fig. 6(e) shows the resulting heading changes. In five heading-step trials, the hand reached the target heading for both $\pm 15^\circ$ steps and for the 30° right turn. The 30° left turns fell short because the offset that cancels the drift also adds to left-turn commands, pushing them beyond the limit. These trials demonstrate commanded heading changes in both directions using onboard sensing.

C. Fall recovery

The hand recovered in 21 of 25 hardware trials (84%), including 11 of 14 thumb-side falls and 10 of 11 wrist-side falls. Trials started from fallen poses with fingers curled and motors initially disabled. The learned recovery policy raised the hand without assistance; in the four failures the fingers caught on each other and the hand stalled. Fig. 7(a)–(d) shows a representative sequence. The upright-state detector then triggered a smooth transition to the crawl stance shown in Fig. 7(e), ready for locomotion.

D. Keyboard pressing

Keyboard pressing tests self-supported interaction with human controls, as shown in Fig. 8(a)–(d). With the policy running onboard, an operator issued the four arrow-key commands in mixed order on a fixed, manually aligned keyboard. We counted presses of the commanded key as correct and presses of wrong keys or timeouts as failures. Fig. 8(e) shows a sequence containing 29 correct presses from 32 consecutive keyboard commands over 72.5 s, without keyboard realignment. The hand maintained its support stance with a maximum tilt of 7.7°. All three failures were Up-key commands that activated the adjacent Right Shift key, consistent with a press-location offset. Median command-to-keystroke latency among correct presses was 0.25 s.

a) Keyboard sequences: The same interface controlled Sokoban through physical keystrokes. With operator-issued commands, the hand successfully executed the optimal 9- and 12-move solutions for one- and two-box levels, respectively (Fig. 1(o)).

E. Vision-guided object pushing

Object pushing tests whether the hand can coordinate body motion and object contact while supporting itself. We used a PLA cube measuring $40 \times 40 \times 40$ mm and weighing 41.4 g. With live overhead tracking of the hand and cube, one policy approached and pushed the cube, as shown in Fig. 9(a). A delivery required entry within 2 cm of the target, followed by 1 s within 5 cm. Fig. 9(b) and (c) show 15 deliveries at target distances of 10 cm to 40 cm, presented as a demonstration of the capability. Across these deliveries, the final target error averaged 17 mm (range 5 mm to 37 mm), under half the cube's side length.

VII. Conclusion

We show that a commercial anthropomorphic hand can serve as a self-contained mobile manipulator. Its fingers move the body, support its weight, and interact with the environment, avoiding a separate locomotion mechanism. A stance-calibrated formulation and a hardware-calibrated simulator support transfer through the existing position-control interface. Simulation ablations show that the footprint objective improves task speed and five-finger participation within the tested formulation. Untethered locomotion, recovery, keyboard pressing, and vision-guided object pushing were demonstrated on hardware.

Extending the range of commanded heading changes and integrating onboard perception would broaden where the hand can operate. Visual registration could automate keyboard alignment, while onboard hand and object tracking could support pushing beyond the overhead camera's workspace.

With these improvements, a larger robot could deploy the hand into a confined workspace, let it operate controls and move objects, and retrieve it. Giving robotic hands their own mobility could make future robots more versatile in the spaces and interfaces built for people.

References

[1] WUJI Technology, "Wuji Hand: Product introduction," archived product documentation. Accessed: Sep. 12, 2026. [Online]. Available: https://docs.wuji.tech/docs/en/wuji-hand/v1/overview/

[2] M. Görner, T. Wimböck, A. Baumann, M. Fuchs, T. Bahls, M. Grebenstein, C. Borst, J. Butterfass, and G. Hirzinger, "The DLR-Crawler: A testbed for actively compliant hexapod walking based on the fingers of DLR-Hand II," in *Proc. IEEE/RSJ IROS*, 2008, pp. 1525–1531.

[3] X. Gao, K. Yao, K. Junge, J. Hughes, and A. Billard, "A detachable crawling robotic hand," *Nature Communications*, vol. 17, no. 1, p. 428, 2026.

[4] T. Sasaki, H. Shimobayashi, M. Hamze, M. Inami, and E. Yoshida, "Locomotion generation of hand-shaped robotic avatar," in *Proc. Augmented Humans Int. Conf. (AHs)*, 2025, pp. 152–159.

[5] H. Shimobayashi, T. Sasaki, M. Hirose, E. Yoshida, and M. Inami, "Handoid: Inter-morphologic robotic hand avatar for multi presence," in *ACM SIGGRAPH Emerging Technologies*, 2025, pp. 1–2.

[6] R. Li, C. Ma, S. Li, Z. Wei, Y. Yao, H. Shi, C. K. Liu, S. Song, and M. Ding, "Handroid: Bridging dexterous hand and humanoid," *arXiv preprint arXiv:2607.16187*, 2026.

[7] E. Lerner and J. Zhao, "Stiffness-programmable origami robots with tendon-driven actuation via multimaterial 3D printing," *Advanced Robotics Research*, 2026.

[8] H. Sun, L. Yang, Y. Gu, J. Pan, F. Wan, and C. Song, "Bridging locomotion and manipulation using reconfigurable robotic limbs via reinforcement learning," *Biomimetics*, vol. 8, no. 4, p. 364, 2023.

[9] X. Cheng, A. Kumar, and D. Pathak, "Legs as manipulator: Pushing quadrupedal agility beyond locomotion," in *Proc. IEEE ICRA*, 2023, pp. 5106–5112.

[10] P. Arm, M. Mittal, H. Kolvenbach, and M. Hutter, "Pedipulate: Enabling manipulation skills using a quadruped robot's leg," in *Proc. IEEE ICRA*, 2024, pp. 5717–5723.

[11] X. He, C. Yuan, W. Zhou, R. Yang, D. Held, and X. Wang, "Visual manipulation with legs," in *Proc. CoRL*, ser. Proceedings of Machine Learning Research, vol. 270, 2025, pp. 4218–4234.

[12] Y. Ji, G. B. Margolis, and P. Agrawal, "DribbleBot: Dynamic legged manipulation in the wild," in *Proc. IEEE ICRA*, 2023.

[13] Z. Fu, X. Cheng, and D. Pathak, "Deep whole-body control: Learning a unified policy for manipulation and locomotion," in *Proc. CoRL*, ser. Proceedings of Machine Learning Research, vol. 205, 2023, pp. 138–149.

[14] D. Hoeller, N. Rudin, D. Sako, and M. Hutter, "ANYmal parkour: Learning agile navigation for quadrupedal robots," *Science Robotics*, vol. 9, no. 88, p. eadi7566, 2024.

[15] W. Yu, G. Turk, and C. K. Liu, "Learning symmetric and low-energy locomotion," *ACM Trans. Graphics*, vol. 37, no. 4, 2018.

[16] F. Abdolhosseini, H. Y. Ling, Z. Xie, X. B. Peng, and M. van de Panne, "On learning symmetric locomotion," in *ACM SIGGRAPH Conf. Motion, Interaction and Games (MIG)*, 2019.

[17] M. Mittal, N. Rudin, V. Klemm, A. Allshire, and M. Hutter, "Symmetry considerations for learning task-symmetric robot policies," in *Proc. IEEE ICRA*, 2024, pp. 7433–7439.

[18] J. Siekmann, Y. Godse, A. Fern, and J. Hurst, "Sim-to-real learning of all common bipedal gaits via periodic reward composition," in *Proc. IEEE ICRA*, 2021, pp. 7309–7315.

[19] G. B. Margolis and P. Agrawal, "Walk these ways: Tuning robot control for generalization with multiplicity of behavior," in *Proc. CoRL*, ser. Proceedings of Machine Learning Research, vol. 205, 2023, pp. 22–31.

[20] J. Tan, T. Zhang, E. Coumans, A. Iscen, Y. Bai, D. Hafner, S. Bohez, and V. Vanhoucke, "Sim-to-real: Learning agile locomotion for quadruped robots," in *Robotics: Science and Systems (RSS)*, 2018.

[21] J. Hwangbo, J. Lee, A. Dosovitskiy, D. Bellicoso, V. Tsounis, V. Koltun, and M. Hutter, "Learning agile and dynamic motor skills for legged robots," *Science Robotics*, vol. 4, no. 26, p. eaau5872, 2019.

[22] F. Bjelonic, F. Tischhauser, and M. Hutter, "Towards bridging the gap: Systematic sim-to-real transfer for diverse legged robots," *arXiv preprint arXiv:2509.06342*, 2025.

[23] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.

[24] M. Mittal *et al.*, "Isaac Lab: A GPU-accelerated simulation framework for multi-modal robot learning," *arXiv preprint arXiv:2511.04831*, 2025.