



Reflex: Visually Grounded Reactive Humanoid Control

Taoyang Jia^{*1,2}, Weikai Huang^{*1}, Linxin Song^{*3}, Jared Darlington¹,
Jieyu Zhao³, Yue Wang³, Jiafei Duan⁴, Zhongzheng Ren^{5,6}, and Ranjay Krishna¹

¹University of Washington ²Stanford University ³University of Southern California

⁴National University of Singapore ⁵University of North Carolina at Chapel Hill ⁶Allen Institute for AI

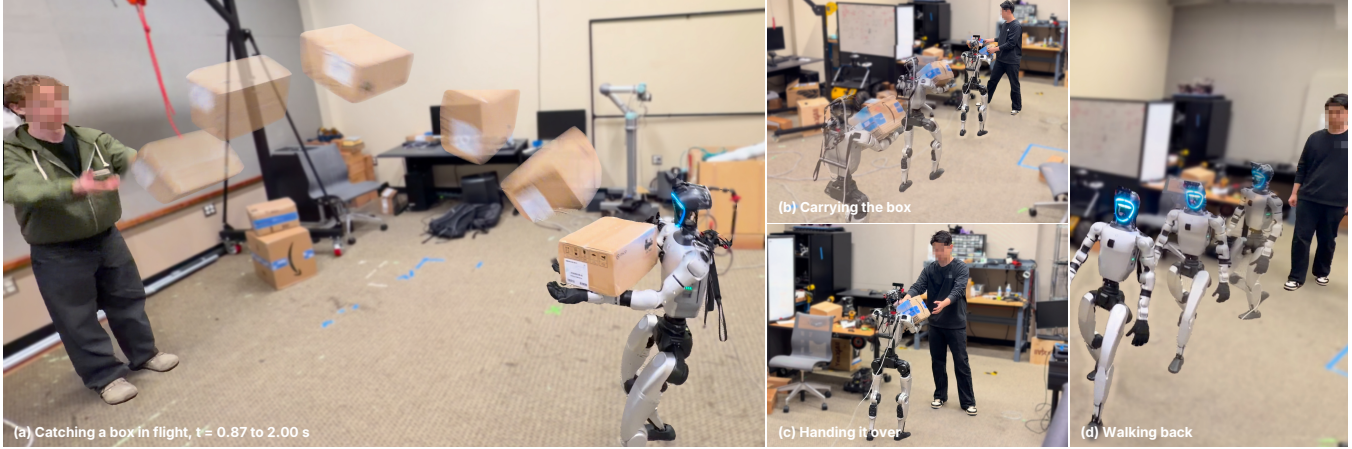


Fig. 1. **Visual reactive whole-body catching on a humanoid.** (a) Catching a thrown carton from onboard RGB-D input. (b)–(d) Downstream application: carrying the box 7.8m, handing it to a person, and walking 7.3m back. The retained box is passed directly to the downstream walking controller for carrying and handover.

Abstract—Humans effortlessly react to the world; they adeptly return a tennis shot, catch a falling cup, step aside from a passing cart, or pass a box tossed during truck unloading. These everyday actions require anticipating a moving target and reflexively responding with the whole body within about a second. Such visual reactive whole-body control remains challenging for humanoids, which must (1) coordinate whole-body control, (2) react to object dynamics in real time, and (3) ground perception in egocentric vision. We study catching thrown boxes as a representative task that demands all three capabilities; existing systems cover only part of it, relying on end-effectors, motion capture, or marked objects. We present **Reflex**, a three-stage framework that learns these capabilities stage by stage while connecting them through a compact representation of object dynamics. First, a state-based policy learns whole-body catching with reinforcement learning. Second, the policy learns to infer the dynamics of a moving box from delayed and incomplete observations. Third, an egocentric RGB-D encoder learns to recover this dynamics representation directly from visual observations while keeping the whole-body controller fixed. This decomposition provides each capability with a direct training signal and makes visual learning substantially more scalable. In simulation, **Reflex** achieves 85.7% catch success from egocentric RGB-D, compared with 87.9% when given privileged box observations. Temporal visual history is critical: removing it reduces success by 35.6 and 39.8 points on centered and laterally offset throws. On a real Unitree G1, **Reflex** catches human-thrown boxes with 65% and 40% success at 2.5 and 5m, reaching 76% and 59% of human catchers’ success at the same distances.

I. INTRODUCTION

Humans routinely react to fast-moving objects with their entire body. A goalkeeper shifts, dives, and braces for an incoming shot; a worker catches a box tossed from a truck; a person reaches for a falling cup while stepping to keep balance; another twists aside as a cart passes unexpectedly close. These behaviors require more than tracking an object in motion. They require predicting where it will go, deciding how the body should meet or avoid it, and coordinating a whole-body response before contact, often within about a second. Humanoids have the morphology to perform such behaviors, but robust *visually reactive whole-body control* remains largely out of reach.

This capability couples three challenges. First, the robot must control the whole body: it must place its feet and torso, move its arms into position, absorb impact, and remain balanced. Second, it must reflexively react to fast object dynamics under latency. In our real-robot trials, a thrown box reaches the robot roughly one second after release, while camera-to-control latency alone has a median of 69 ms in our detector-based pipeline. The robot therefore cannot simply react to the latest observation; it must infer where the object is going and commit to an interception before contact. Third, it must do so from egocentric vision. Because the camera moves with the robot, observed image motion combines object motion with self-motion, while the robot’s own arms increasingly occlude the object near contact.

*Equal contribution (co-first authors).

We study catching thrown boxes as a representative test of all three challenges. Catching a box is more demanding than intercepting a ball with the hands: the robot must reposition its body, use the forearms and torso to form a support, absorb the box’s momentum, and retain the load without losing balance. At the same time, the box is an ordinary uninstrumented object observed only through onboard RGB-D cameras. The task is difficult even for people: across our trials, three human catchers succeed on only 28.3% of throws from 7 m.

Existing systems address only parts of this problem. Robotic manipulators can catch moving objects from fixed or wheeled bases, avoiding the challenge of whole-body balance and impact absorption [1], [2], [3]. Recent humanoid systems demonstrate dynamic interactions such as punching away a ball [4] or catching a basketball with the hands [5], but rely on motion capture or retroreflective objects rather than grounding fast object motion from onboard vision. As a result, whole-body control, fast prediction of object dynamics, and egocentric visual grounding have rarely been combined on a real humanoid.

We present *Reflex*, a three-stage framework for visually reactive whole-body control (see Figure 2). **Stage 1** learns whole-body catching from privileged simulator state using reinforcement learning and human motion priors. **Stage 2** learns to infer the dynamics of a moving box from delayed (20–60 ms latency) and incomplete observations of its position, velocity, orientation, size, and observation age, producing a latent representation that conditions the controller to react. **Stage 3** learns to recover this same dynamics representation from egocentric RGB-D history and camera ego-motion while keeping the whole-body controller fixed. Our decomposition gives each capability a direct training signal and isolates expensive visual learning from reinforcement learning.

In simulation, *Reflex* achieves 85.7% catch success from egocentric RGB-D, compared with 87.9% when given privileged box observations. Temporal visual history is vital for good performance; removing it reduces success by 35.6 and 39.8 points on centered and 0.6m-offset throws. On a real Unitree G1, *Reflex* catches human-thrown boxes on 65% and 40% of throws at 2.5 and 5 m, reaching 76% and 59% of the human success rate at the same distances (85.0% and 68.3%). Ablations further show that camera ego-motion matters for visual grounding, that regressing the learned latent works better than regressing the full privileged box state, and that action-matching initialization outperforms weight mapping by 5.2 points on average.

More broadly, our results suggest a path toward humanoids that can react to dynamic events directly from onboard vision, rather than relying on instrumented environments or task-specific perception systems. Catching is one instance of a broader class of visually reactive whole-body skills, including interception, avoidance, handoffs, and physical assistance. These everyday tasks require predicting how the world will change and coordinating the body before contact.

II. RELATED WORK

We review prior work on catching thrown objects, on humanoid whole-body control and loco-manipulation, and on privileged learning for visual sim-to-real transfer.

A. Catching thrown objects

Catching a thrown object requires predicting its trajectory and moving into position within a short time window. Early systems addressed this with fixed-base arms guided by vision [6], with later work learning object dynamics and catching motions from demonstration [1] or planning impact-aware bimanual motions with fixed-base arms [7]. Learning-based approaches have since extended catching to whole-body model predictive control combined with policy learning [2], dexterous and bimanual throw-and-catch [8], [9], and mobile manipulators [10], [3]. More recently, bipedal humanoids have been trained to intercept balls as goalkeepers [4] and to catch basketball passes [5]. However, most of these systems observe the object with motion capture or fixed external cameras, with fewer relying on onboard sensing: humanoids have kicked and struck moving balls from onboard vision [11], [12], a legged manipulator plays badminton with a learned perception-error model [13], and onboard catching has been shown on a wheeled humanoid [14] and a quadruped [15], and concurrent work catches a thrown box on a humanoid from egocentric RGB without reporting real-robot success rates [16]. To address this, we train a humanoid to catch and retain a thrown box using onboard egocentric RGB-D.

B. Humanoid whole-body control and loco-manipulation

Human motion data has become a common foundation for humanoid whole-body control. This approach began in physics-based character animation with motion imitation [17] and adversarial motion priors [18]. Extending this to object interaction, Catch & Carry [19] paired a motor-primitive module learned from human demonstrations with an egocentric-vision policy trained by reinforcement learning to catch balls and carry boxes in simulation. These methods have since moved from simulation to hardware, where learned motion trackers provide general whole-body control on real humanoids [20], [21], [22]. Recent loco-manipulation systems go further, lifting and carrying boxes and bulky objects through whole-body contact [23], [24], [25], [26], [27], [28]. In these tasks, though, the object is at rest when interaction begins, so the robot can approach and make contact at its own pace. Catching removes this freedom, since the body must already be in place when the box arrives. Like these methods, we build on human motion, but train a dedicated catching controller that reacts to the incoming box and coordinates interception and retention.

C. Privileged Learning and Visual Sim-to-Real

Policies trained in simulation can use information that is unavailable at deployment, such as the exact state of an object. Privileged learning exploits this by training a teacher with such information to supervise a student

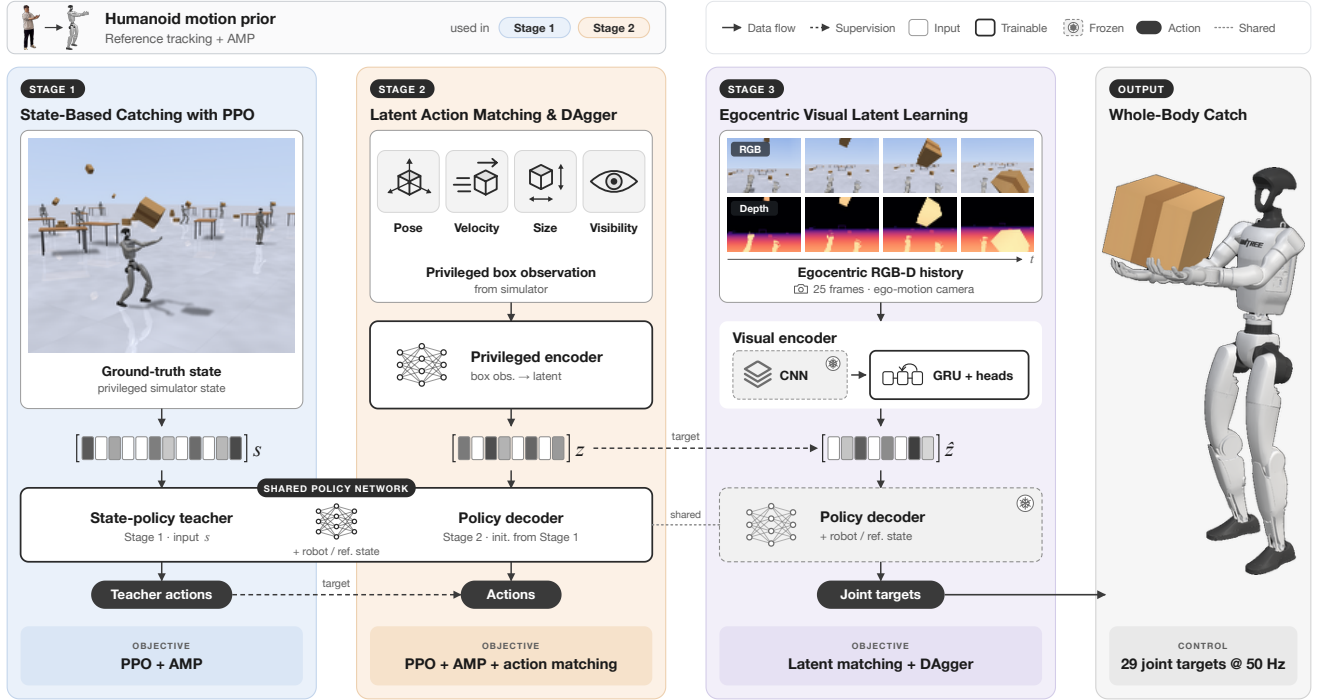


Fig. 2. **Training pipeline and policy architecture.** Stage 1 learns whole-body control by training the state-policy teacher with PPO. Stage 2 learns the object dynamics: it initializes the privileged encoder and decoder latent-input weights by matching teacher actions, then jointly refines the encoder and decoder with PPO. Human motion examples supply reference tracking and an AMP style reward. Stage 3 learns visual grounding by predicting the latent from RGB-D histories and camera ego-motion, using box-state supervision and DAgger rollouts. The pretrained CNN and heatmap head stay fixed during GRU training; the privileged encoder and decoder remain fixed throughout Stage 3. The decoder also receives robot/reference observations. Because Stage 3 learns only the latent predictor, the visual controller reuses the PPO-trained decoder without modification.

that uses onboard sensors [29], typically alongside asymmetric critics [30], domain randomization [31], and on-policy data aggregation [32]. Legged robots adopted this approach early, with students that reproduce a teacher’s latent from observation history [33], [34] or walk from egocentric depth [35]. Humanoids now use it to perform loco-manipulation, door opening, parkour, and ladder climbing from depth or RGB [36], [37], [38], [39]. However, these students perceive static terrain or slow-moving objects. A thrown box instead moves quickly through the view of a moving camera. Following RMA [34], we keep the action decoder fixed and train a visual model by latent regression to predict the privileged latent from RGB-D history and camera ego-motion.

III. TASK FORMULATION

We first define the catching task and its sensing and control interface, then the criteria used to evaluate it.

A. Visually Grounded Reactive Catching

We study the task of catching a human-thrown box with a 29-degree-of-freedom (DoF) Unitree G1. The robot must move into the path of the incoming box, receive it on its forearms against the torso, dissipate its momentum, and retain it without falling. Unlike quasi-static loco-manipulation, the robot cannot choose when contact occurs: it must infer the box trajectory and coordinate its whole body before the box arrives.

At deployment, the visual input is a stream of 192×144 RGB-D observations $\mathbf{I}_t$ from a torso-mounted camera. The controller additionally receives the robot state required for whole-body stabilization, but no box pose, velocity, or external-camera measurement. The camera moves with the robot and the box becomes increasingly occluded near contact, making the task a closed-loop coupling of visual estimation and whole-body response.

At each 20ms control step, the model outputs a 29-dimensional whole-body joint command, corresponding to one command for each actuated DoF. These commands are executed as joint-position targets. Thus, the task-level interface maps egocentric RGB-D observations and robot state directly to 29-DoF control; the internal policy decomposition is introduced only in Sec. IV. Simulation uses Isaac Lab [40], with physics at 200 Hz and control at 50 Hz.

B. Success Criteria

We evaluate three distinct outcomes. *Interception* indicates that the robot has arrested the incoming motion of the box: the box slows below 1 m/s near the torso, regardless of whether it is subsequently retained. The primary metric, *catch success*, additionally requires the box to be supported in the chest region, with the wrists on opposite sides of the box, without relying on the head or shoulders, and at low speed for 0.6 s. This criterion distinguishes sustained retention from incidental contact or a momentary trap.

We assess post-catch stability separately because retaining the box does not necessarily leave the robot in a settled pos-

ture. *Torso excursion* records whether the torso tilt reaches 38° between launch and the end of the episode, while falls are reported independently. For the box-size sweep, where a fixed chest-region test is not comparable across geometries, we use the geometry-agnostic sustained-hold criterion specified in Sec. V-A. These metrics separate perception and interception failures from retention and whole-body stabilization failures.

IV. LEARNING EGOCENTRIC WHOLE-BODY CATCHING

a) Perception-control decomposition: We decompose the deployed policy into a visual perception function V_ψ and a whole-body control function π_θ . Given a history of K egocentric RGB-D observations and camera ego-motion, V_ψ predicts an eight-dimensional object-dynamics latent. The controller combines this latent with the robot/reference observation $\mathbf{r}_t \in \mathbb{R}^{160}$ and produces a normalized 29-DoF action:

$$\hat{\mathbf{z}}_t = V_\psi(\mathbf{I}_{t-K+1:t}, \mathbf{e}_{t-K+1:t}), \quad \mathbf{a}_t = \pi_\theta(\mathbf{r}_t, \hat{\mathbf{z}}_t). \quad (1)$$

The action is converted to nominal joint-position targets as

$$\mathbf{q}_t^{\text{nom}} = \mathbf{q}_0 + \mathbf{s} \odot \mathbf{a}_t, \quad (2)$$

where $\mathbf{q}_0 \in \mathbb{R}^{29}$ is the nominal joint configuration, $\mathbf{s} \in \mathbb{R}^{29}$ contains per-joint action scales, and $\odot$ denotes elementwise multiplication. This latent interface separates the question of what object information can be inferred from vision from how that information should drive the body.

Training V_ψ and π_θ jointly from pixels with reinforcement learning is both computationally expensive and difficult to optimize. Every on-policy rollout would require high-throughput RGB-D rendering, while the policy must simultaneously explore a large visual space and a 29-dimensional action space. Catch success is delayed until contact and retention, providing weak supervision for learning object motion from images; camera ego-motion, intermittent occlusion, and policy-induced viewpoint changes further couple perception errors to the state distribution visited by the controller. A monolithic end-to-end policy would therefore entangle visual representation learning with motor-skill discovery and make perception adaptation require retraining the controller.

We instead organize learning into **three stages**, one per capability (Fig. 2): **Stage 1** learns whole-body control from box state with PPO, **Stage 2** learns the object dynamics as a compact control latent (its PPO also refines the controller), and **Stage 3** grounds this latent in vision while the controller remains fixed. A downstream carrying and handover module follows the catching controller (Sec. IV-D).

A. Stage 1: Learning Whole-Body Control

Stage 1 isolates the motor skill: given box-state channels, PPO learns body placement, receiving posture, impact absorption, and retention in rendering-free simulation.

a) State observations and policy: We train a state-based catching policy π_{state} with proximal policy optimization (PPO) [41] over successive warm-started runs. Its 170-dimensional observation $\mathbf{o}_t^{\text{state}}$ combines the robot/reference input $\mathbf{r}_t$ with ten box-state channels: pelvis-frame position and linear velocity (three each), observation age (one), and indicators of measurement jitter, freshness, and frozen observations (three). Simulation supplies these channels by corrupting box detections with randomized noise, missed observations, and delay and passing them through the Kalman filter used on hardware (gravity and linear drag). The actor is a multilayer perceptron with hidden widths 512–256–128, exponential linear unit activations, and the 29 outputs of Eq. (2); an asymmetric critic additionally observes uncorrupted full-body poses and ground-truth box position and velocity.

b) Catching objective and randomized training: Approach shaping encourages interception, hold shaping rewards forearm support near the chest, and a sustained-catch bonus rewards retention. Tracking retargeted human reference motion and an adversarial motion prior (AMP) [18] guide coordinated movement; penalties discourage excessive torso motion, abrupt commands, and falls. PPO updates the actor and critic, while the AMP discriminator learns from simulated motion and retargeted human examples. Throws, contact properties, and robot dynamics are randomized; held-box initializations and perturbations train retention and recovery.

c) Teacher for latent transfer: After state-based PPO, we freeze the teacher. For Stage 2 action supervision, we query it with ground-truth position and velocity channels. Its decoder weights initialize the latent controller, and its actions label the initialization dataset. The ground-truth state-policy reference in the experiments instead feeds ideal (complete, noise-free, zero-delay) box measurements through the policy’s Kalman filter.

B. Stage 2: Learning Object Dynamics

Stage 2 learns what the controller needs to know about the moving box: a privileged encoder compresses a delayed, partially observed box observation into a latent that PPO shapes to account for the box’s motion, defining the object dynamics that Stage 3 must perceive.

a) Privileged latent interface: The privileged encoder E_ϕ maps a box observation $\mathbf{p}_t \in \mathbb{R}^{26}$ to the latent used by the control function π_θ :

$$\mathbf{z}_t = E_\phi(\mathbf{p}_t) \in [-1, 1]^8. \quad (3)$$

The 26 channels comprise camera-frame position (3), relative linear velocity (3), box half extents (3), orientation (12), angular velocity (3), visibility (1), and normalized observation age (1). Orientation uses the six unique entries of each axis outer product for the box’s long and short axes, making the representation invariant to axis-sign flips.

We select the freshest visible sample in a delayed 25-frame window, with field-of-view masking and observation dropout. When no sample is visible, position, linear/angular

velocity, and orientation terms are zeroed, visibility is zero, and normalized age is one; box half extents remain available. A linear projection and residual MLP, followed by tanh, produce the bounded latent, whose coordinates are learned through action matching and task optimization. The encoder thus sees one delayed sample, including velocity, at a time; inferring motion from observation history is left to Stage 3.

b) *Action-matching initialization*: Let $\mathbf{a}_t^T = \pi_{\text{state}}(\mathbf{o}_t^{\text{state}})$ denote the teacher mean action. The action-supervision dataset $\mathcal{D}_A$ stores tuples $(\mathbf{r}_t, \mathbf{p}_t, \mathbf{a}_t^T)$. We retain the teacher’s decoder body, replace its object-input weights with randomly initialized latent-input weights, and initialize the new path-way by behavior cloning (BC):

$$\mathcal{L}_{\text{BC}} = \mathbb{E}_{\mathcal{D}_A} \left[w_t^{\text{vis}} \left\| \pi_{\theta}(\mathbf{r}_t, E_{\phi}(\mathbf{p}_t)) - \mathbf{a}_t^T \right\|_2^2 \right]. \quad (4)$$

The expectation is over dataset samples, and w_t^{vis} is a visibility-based sample weight: 5 for visible-box samples and 1 otherwise. Only the privileged encoder and the first decoder layer’s latent-input weights are updated; the remaining decoder parameters stay fixed.

c) *Action-labeled data aggregation*: We first collect teacher-driven trajectories, then run the latent controller and query the fixed state-policy teacher for action labels on its visited states. This dataset aggregation (Dagger)-style procedure [32] expands $\mathcal{D}_A$ with states encountered under latent control, and action matching is repeated on the aggregated data. These teacher action labels are distinct from the retargeted human motion examples used for reference tracking and the motion prior.

d) *Task-specific PPO refinement*: After action matching and aggregation, PPO jointly optimizes the privileged encoder and the full decoder, using the catching objective and motion priors introduced in Stage 1. The asymmetric critic retains access to uncorrupted body poses and ground-truth box state while the actor receives $\mathbf{r}_t$ and the dynamics latent computed from delayed box observations. Teacher action matching is used for initialization and aggregation; PPO refinement optimizes task return and, because the decoder is also updated, refines whole-body control as well (Sec. V-C).

Latent-controller training extends the box-size distribution while retaining throw, dynamics, contact, and recovery randomization. Stage 2 samples object-observation latency once per episode from one to three control steps (20–60 ms), training the controller to act on delayed perception; the camera-to-control latency of our detector-based hardware pipeline (69 ms median; Tab. IV) exceeds this range, and Stage 3 targets add a further 40 ms lag. In a single-seed comparison, a controller trained without it is 5–11 points lower at every tested delay (Sec. V-D). Figure 3 illustrates the variations; numerical settings are given in Sec. V-A. The resulting encoder and decoder are frozen for Stage 3.

C. Stage 3: Learning Visual Grounding

Stage 3 grounds the dynamics latent in egocentric vision by supervised regression from RGB-D and ego-motion his-

tories; rendering enters only this stage, and the controller remains unchanged.

a) *Temporal visual architecture*: The visual encoder V_{ψ} implements the perception function in Eq. (1) using histories of length $K = 25$. Visual control retains the robot/reference input, decoder, and joint-command interface; only the source of the dynamics latent changes.

A convolutional neural network (CNN) predicts a box-localization heatmap. A fixed geometric layer forms $\mathbf{g}_t \in \mathbb{R}^8$ from normalized image coordinates (2), local minimum depths in two neighborhoods (2), a heatmap peak score (1), and a camera-frame surface point obtained by pinhole backprojection (3). A gated recurrent unit (GRU) processes these features and nine-dimensional ego-motion vectors to infer object motion under viewpoint changes and missed detections. Appearance thus reaches the GRU only through the localization heatmap; in simulation, ego-motion comes from the simulator. Two output heads predict the dynamics latent and auxiliary position/velocity; the latter supplies training supervision, while only the latent drives the decoder.

b) *Two-step visual training*: The Stage 2 privileged encoder E_{ϕ} and action decoder π_{θ} stay fixed throughout visual training. We train the visual encoder in two steps. First, we train the RGB-D CNN and box-localization heatmap using Gaussian targets at projected box centers with binary cross-entropy supervision. Second, we freeze the CNN and heatmap head and train the GRU to predict the dynamics latent and auxiliary box position and velocity from geometric-feature and ego-motion histories, using the objective below.

The visual-training dataset $\mathcal{D}_V$ pairs histories ending at t with latent targets $\mathbf{z}_t = E_{\phi}(\mathbf{p}_t)$ from the fixed privileged encoder, where $\mathbf{p}_t$ uses a fixed two-step (40 ms) latency and no dropout, and auxiliary targets $\mathbf{u}_t \in \mathbb{R}^6$, consisting of standardized current camera-frame box position and velocity. Let $\hat{\mathbf{u}}_t \in \mathbb{R}^6$ denote the auxiliary prediction. The temporal-training objective is

$$\begin{aligned} \mathcal{L}_V &= \mathbb{E}_{\mathcal{D}_V} \left[\omega_t (\ell_z(t) + \lambda_{\text{aux}} \ell_{\text{state}}(t)) \right], \\ \ell_z(t) &= \frac{1}{8} \sum_{j=1}^8 \alpha_j \left(\frac{\hat{z}_{t,j} - z_{t,j}}{\sigma_j + \varepsilon} \right)^2, \\ \ell_{\text{state}}(t) &= \frac{1}{6} \|\hat{\mathbf{u}}_t - \mathbf{u}_t\|_2^2. \end{aligned} \quad (5)$$

Here σ_j is the training-set standard deviation of latent coordinate j , and ε ensures numerical stability. The nonnegative weights α_j , normalized to average one, reflect the fixed decoder’s action sensitivity to one-standard-deviation latent errors. The coefficient λ_{aux} balances auxiliary supervision, and ω_t emphasizes close-range observations (Sec. V-A).

c) *Visual-policy data aggregation*: Initial data come from privileged-controller rollouts. We then run the visual controller, label its visited states with the fixed privileged encoder, aggregate the data, and refit the GRU and output heads with the localizer and controller fixed. This second Dagger-style procedure [32] provides latent targets for the views encountered under visual control. Unlike the action-labeled aggregation in Stage 2, it supervises the predicted latent rather than the controller’s action.

Cached geometric features, ego-motion, and privileged observations provide aligned sequences and latent labels. Windows stay within episodes; validation uses separate episodes, and aggregation-round selection and final evaluation use separate throws.

d) Appearance adaptation: We augment box colors during localizer pretraining and rollout collection; localizer pretraining also applies random RGB gain and offset, pixel noise, small image shifts, and about 2.4 cm depth noise, and the temporal model is fit on noise-perturbed features with the color-augmented localizer frozen. Motion blur, auto-exposure dynamics, and structured depth artifacts are not modeled.

D. Downstream Carrying and Handover

The downstream demonstration in Fig. 1 connects the catching policy to the pretrained SONIC motion controller [22] for walking with the retained box and handover. Visual human detection identifies the recipient for the handover stage.

A supervisor transfers command authority after persistent near-chest box evidence and low leg velocity, torso tilt, and base speed establish a settled hold. Hold-state persistence accommodates close-range occlusion. At transfer, the supervisor establishes the navigation frame and switches gains and observations to SONIC. The reference holds the current posture for 2 s before transitioning to a standing cradle for walking and handover. A MuJoCo deployment twin checks perception, communication, joint commands, and controller transfer.

In the 20-trial downstream task evaluation, cumulative success rates are 65% after catching, 35% after carrying, 30% after handover, and 25% after return.

V. EXPERIMENTS

Our experiments address four questions:

- **Q1:** How well does `Reflex` catch in simulation and on the real robot, compared with privileged input and human catchers?
- **Q2:** How do the catching objective and motion priors shape whole-body control, and how does catching vary with box size and payload?
- **Q3:** How should the object-dynamics latent be learned, and how does it handle lateral offsets and perception delay?
- **Q4:** Which visual inputs and supervision ground the latent in egocentric vision, and how do appearance, distance, and detector-based perception affect it?

In the tables, pink identifies proposed configurations, gray denotes references or auxiliary diagnostic rates, and bold marks the highest catch rate in each comparison, excluding gray references. The abbreviation w/o means without, and GT denotes ground truth.

A. Experimental Setup

a) Training settings: Stage 1 uses randomized state observations; its teacher is queried with ground-truth box position and velocity, and the ground-truth reference feeds

ideal box measurements through its Kalman filter. Stage 2 BC initialization uses five data-aggregation rounds and gives visible-box samples fivefold weight. Stage 2 joint PPO uses 4096 environments, 600 iterations, and an initial learning rate of 10^{-4} , without additional latent noise or whole-vector dropout. Release distances span 2.5–5.0 m; box size is randomized among three sizes: $25 \times 19 \times 13$, $34 \times 26 \times 22$, and $42 \times 32 \times 27$ cm.

The privileged encoder has two 64-unit residual hidden layers. The visual model combines eight geometric features and nine ego-motion features using a two-layer, 128-unit GRU. The color-augmented localizer is pretrained for 20 epochs with batch size 32 and learning rate 3×10^{-4} . Temporal training uses AdamW for 60 epochs with batch size 512 and maximum learning rate 2×10^{-3} . Sequence training uses $\lambda_{\text{aux}} = 0.5$ and an eightfold sample weight at box ranges of 0.5–2.5 m, normalized within each minibatch. Visual training uses two or three DAGger rounds per controller, selected on held-out evaluation throws. On one RTX 3090, rendering-free Stage 2 PPO runs at about 28k environment steps/s (4096 environments), whereas rendering 192×144 RGB-D in the loop runs at about 1.5k steps/s (320 environments); the 59M steps of Stage 2 PPO take about 35 min instead of roughly 11 h, while Stage 3 renders about 1.6M steps.

b) Evaluation protocol and comparisons: Each environment contributes one throw from 3.0–3.5 m with a $34 \times 26 \times 22$ cm box. Evaluation starts without a held box, scores from the first step, and ends after 640 steps before reset. Rethrows, deliberately unreachable targets, robot pushes, and held-box kicks are disabled. Reference start frames are uniformly sampled from the standing segment, independently of the policy.

Throw attitude, spin, aim scatter, physical properties, and reset-state noise remain randomized. A catch requires chest-region support, wrists flanking the box, clearance from head/shoulder support, and low box speed for 0.6 s. Torso excursion records whether tilt reaches 38° between launch and evaluation end, separately from catch success and falls.

The initialization comparison trains six controllers per latent configuration and evaluates each on 640 held-out throws (3840 throws per configuration); the controllers share the predecessor policy and, for our configuration, BC initialization. Tables report mean catch rates. The state-policy reference uses one checkpoint.

We compare the ground-truth state-policy reference and two latent configurations, each with privileged and visual inputs. Both inherit the decoder body and use the same architecture and control interface. Ours uses BC initialization; the state-initialized alternative aligns encoder coordinates with position/velocity and maps the predecessor’s input weights without BC. Both continue joint encoder/decoder PPO training (Tab. VII). The privileged observation mask uses the hardware camera’s field of view ($\pm 53^\circ \times \pm 40^\circ$), whereas the rendered 4:3 pinhole image spans $\pm 45^\circ$ vertically, so privileged-input results are a reference for visual substitution. Tab. I, Tab. IV, and Fig. 4 use the first-trained of the six controllers (81.6–91.7% at the reference): its default visual

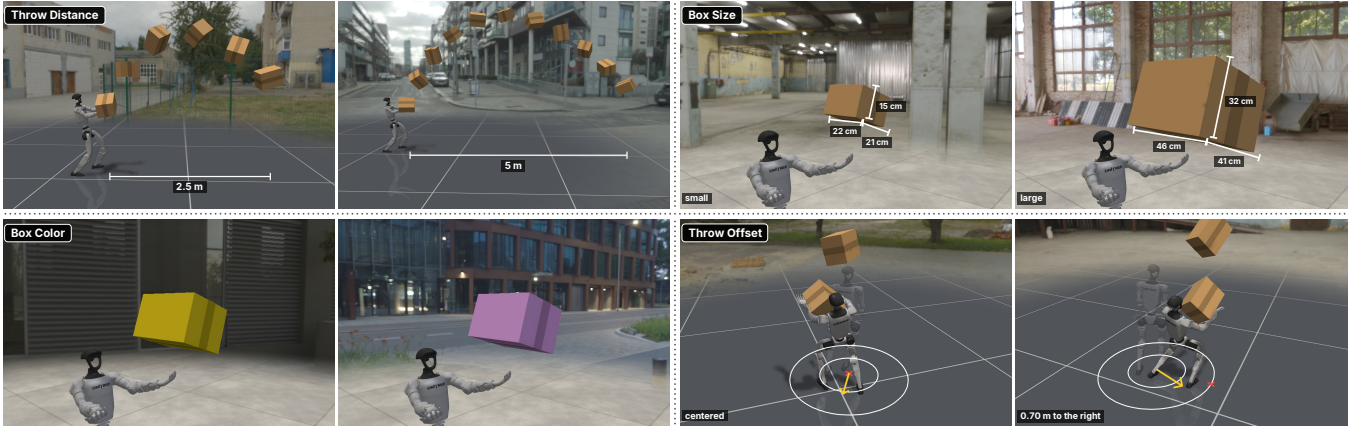


Fig. 3. **Throw and appearance variations.** Simulation examples show changes in release distance, box size, color, and landing location. The box-size panel illustrates measured real boxes; simulated weight and distance tests use the $34 \times 26 \times 22$ cm training box. Color augmentation is applied when training the visual component. These variations define the operating conditions evaluated in Tab. I.

component in the figure (85.3%) and its color-augmented component in the tables (88.4–88.8%). In tables and figures, the state policy is the Stage 1 controller, privileged $\mathbf{z}$ is the Stage 2 latent controller with privileged box observations, and visual $\hat{\mathbf{z}}$ (visual latent) is the full Stage 3 system, Reflex.

c) Condition benchmark: Tab. I varies one factor at a time from the setting of Sec. V-A: a 0.18 kg box thrown from 3.0–3.5 m with at most 5° aim error. Small and large sizes are measured real boxes; the regular size and the weight and distance tests use the $34 \times 26 \times 22$ cm training box, and distance tests raise the launch-speed cap from at most 8.5 m/s in training to about 12 m/s, so 99% of 5 m and all 7 m throws are faster than any training throw. The large box exceeds every training size, the small box is near the smallest ($25 \times 19 \times 13$ cm), 0.5–2 kg payloads and 7 m throws lie outside the training conditions, and Stage 3 data use the 3.0–3.5 m throw setting. Weight, distance, and color columns use the chest-region criterion of Sec. V-A. Because box geometry changes, the size columns instead require the box to settle within 0.65 m of the pelvis below 1.2 m/s for 0.6 s, stay above 0.35 m, and remain at rest at 12.8 s; early resets fail. Interception requires the box to slow below 1 m/s near the torso, irrespective of retention. The visual-latent rows use the color-augmented visual component.

B. Overall Performance

For Q1 in simulation, the visual controller catches 3290/3840 throws (85.7%), compared with 87.9% using privileged latent input to the same decoder (Tab. VII), so visual prediction preserves most of the catch performance.

On the real robot, Tab. II reports 20 trials per condition. Catch success reaches 65%, 40%, and 15% for the regular box at 2.5, 5, and 7 m and ranges from 30% to 60% across the tested payloads; with 20 trials, 95% Wilson intervals span up to about ± 20 points. As a human reference, three people each caught 20 regular-box throws per distance, standing with arms relaxed until release. Pooled human success is 85.0%, 68.3%, and 28.3% at 2.5, 5, and 7 m; the real G1 reaches 76% and 59% of it at 2.5 and 5 m (95% intervals 54–107%

and 33–103%), and both are low at 7 m, which approaches the practical range of human throwing and catching.

C. Whole-Body Control

To answer Q2, we ablate the Stage 1 objective, reused in latent PPO, and examine box size and payload.

a) Reward objective: Rewards are removed only during Stage 2 PPO from the shared initialization, so Tab. III tests what retains and extends the skill. PPO raises success from 75.3% to 88.6% for centered throws and from 44.2% to 76.1% at 0.6 m offset. Removing hold shaping reduces offset success to 50.8%; removing all three task groups gives 2.7%. Removing tracking or AMP alone costs 4–5 points, not significant over three seeds; removing both collapses training (a foot-height termination ends most episodes), and no episode reaches the throw. Without Stage 1, the same latent controller trained by PPO from random initialization with the Stage 2 recipe and a standard learning rate catches only 6.7% and 0.2% after 1000 iterations (6.7% and 0.5% after 600; one seed); Stage 1 itself was trained for longer, so this shows that it is needed within the Stage 2 budget.

b) Retention and torso motion: Torso excursion ranges from 1% to 72% across the six visual controllers (mean 21%, versus 14% for the state-initialized group). One checkpoint catches 91.7% but has 72% excursion; another catches 86.2% with 1% excursion. These post-launch metrics distinguish successful box retention from a settled whole-body posture. For the six visual controllers, the pelvis never drops below 0.52 m, and 3 of 3840 episodes end early (counted as failures).

c) Box size and payload: With visual input, 0.5–2 kg payloads are caught at 77.7–91.2%, on par with the state policy (73.1–88.8%; Tab. I). Performance drops for boxes outside the reference size: visual catch success is 25.5% for small boxes and 39.7% for large boxes, compared with interception rates of 58.1% and 86.1%. The gap to interception indicates that many of these boxes are reached but not retained. The controller is effectively size-blind: Stage 1 trains on the regular box without a size input, $\mathbf{z}$ does not encode size, and small boxes fail with privileged input as

TABLE I

SIMULATION ROBUSTNESS. CATCH AND INTERCEPTION RATES (%); ONE CHECKPOINT PER CONTROLLER, 640 THROWS PER CONDITION (320 FOR THE DETECTOR). WITH THE SAME DETECTOR, VISUAL LATENT CONTROL EXCEEDS DETECTION PLUS FILTERING WITHIN ITS VISUAL TRAINING RANGE.

Controller	Rate (%)	Box size (cm)			Avg.	Box weight (kg)				Avg.	Throw distance (m)				Avg.	Box color								Avg.
		Small 22×21×15	Regular 34×26×22	Large 46×41×32		0.5	1.0	2.0	2.5		5.0	7.0	White	Grey		Red	Green	Blue	Black					
<i>Privileged box-state input</i>																								
Stage 1 / GT state	Catch	19.5	89.4	43.1	50.7	88.8	85.2	73.1	82.4	88.1	79.7	58.3	75.4	88.9	88.9	88.9	88.9	88.9	88.9	88.9	88.9			
	Intercept	54.2	93.8	79.4	75.8	95.5	95.5	93.0	94.7	93.3	88.0	65.3	82.2	93.8	93.8	93.8	93.8	93.8	93.8	93.8	93.8			
Stage 2 / privileged z	Catch	21.6	88.4	47.8	52.6	93.4	92.5	84.7	90.2	90.9	79.7	41.6	70.7	88.3	88.3	88.3	88.3	88.3	88.3	88.3	88.3			
	Intercept	46.6	96.2	86.7	76.5	97.0	96.7	93.9	95.9	97.0	90.6	49.5	79.0	96.2	96.2	96.2	96.2	96.2	96.2	96.2	96.2			
<i>Egocentric RGB-D input</i>																								
Detector (YOLO) + state-based	Catch	11.2	68.1	17.5	32.3	67.2	59.7	48.4	58.4	71.9	60.0	32.8	54.9	65.3	10.9	65.6	62.5	56.9	41.6	50.5				
	Intercept	45.6	78.4	70.9	65.0	80.9	78.1	72.8	77.3	80.9	74.7	41.2	65.6	77.2	15.0	80.9	75.6	68.1	55.6	62.1				
Detector (our CNN) + state-based	Catch	20.5	73.3	24.4	39.4	75.6	76.2	65.5	72.4	85.0	66.9	38.3	63.4	77.3	74.7	74.5	73.8	77.2	77.7	75.9				
Detector (our CNN) + latent decoder	Catch	18.0	85.2	29.5	44.2	90.9	91.2	83.3	88.5	88.8	67.7	33.9	63.5	85.2	87.0	87.0	86.4	88.8	87.0	86.9				
Stage 3 / visual z (ours)	Catch	25.5	88.8	39.7	51.3	91.2	89.7	77.7	86.2	90.2	43.0	13.3	48.8	87.5	87.7	86.9	87.2	86.6	87.8	87.3				
	Intercept	58.1	95.5	86.1	79.9	97.2	97.2	91.9	95.4	97.3	53.3	18.6	56.4	95.5	95.6	94.2	95.8	95.5	95.8	95.4				

Avg.: unweighted mean within each condition group. Detector (YOLO): real-photo fine-tuned YOLOv8s on rendered 640×480 RGB-D; box depth feeds the state policy's Kalman filter. Simulation rows are single checkpoints, so differences below about 5 points are not resolved; the Regular column uses the $34 \times 26 \times 22$ cm training box. Intercept: box slowed below 1 m/s near the torso, without requiring retention. Binomial standard error: at most 2.0 points for 640 throws, 2.8 for 320. Detector (our CNN): our color-augmented heatmap CNN (same rendered training data and camera as ours) with a Kalman filter tuned on held-out throws and a ground-truth arm trigger; its state drives the state policy or, via E_ϕ , our decoder.

TABLE II

REAL-ROBOT CATCHING. CATCH RATES (%) OVER 20 G1 TRIALS PER CONDITION, VARYING ONE FACTOR AT A TIME WITH THE OTHERS FIXED AT ABOUT 2.5 M AND THE REGULAR BOX; 95% WILSON INTERVALS SPAN UP TO ABOUT ± 20 POINTS (E.G., 0.5 VS. 1 KG: 9/20 VS. 12/20, FISHER $p = 0.53$). THE G1 REACHES 76% AND 59% OF HUMAN SUCCESS (THREE PEOPLE, 60 TRIALS PER CONDITION) AT 2.5 AND 5 M.

Catcher	Box size (cm)				Box weight (kg)				Throw distance (m)			
	Small 22×21×15	Regular 31×26×21	Large 46×41×32	Avg.	0.5	1.0	2.0	Avg.	2.5	5.0	7.0	Avg.
Real G1 (ours)	20.0	65.0	45.0	43.3	45.0	60.0	30.0	45.0	65.0	40.0	15.0	40.0
Human reference	75.0	85.0	78.3	79.4	85.0	86.7	76.7	82.8	85.0	68.3	28.3	60.6

TABLE III

REWARD ABLATION. CATCH RATES (%), MEAN $\pm$ S.D. OVER THREE TRAINING SEEDS) AFTER PPO. REMOVING ALL THREE TASK GROUPS DROPS SUCCESS TO 33.7% AND 2.7%. *FOOT-HEIGHT TERMINATION; NO EPISODE REACHES THE THROW. [‡]STAGE 2 RECIPE TRAINED BY PPO FROM RANDOM INITIALIZATION, ONE SEED, 1000 ITERATIONS.

Training objective	Catch rate (%), lateral offset (m)	
	0.0	0.6
Init. only (no PPO)	75.3	44.2
No Stage 1 (scratch) [‡]	6.7	0.2
Full objective	88.6$\pm$4.8	76.1$\pm$4.8
<i>Task rewards removed</i>		
w/o success bonus	87.8 $\pm$ 1.6	71.8 $\pm$ 2.6
w/o approach shaping	82.8 $\pm$ 5.0	68.2 $\pm$ 5.6
w/o hold shaping	83.9 $\pm$ 3.0	50.8 $\pm$ 3.8
w/o all three task groups	33.7 $\pm$ 5.0	2.7 $\pm$ 1.0
<i>Motion priors removed</i>		
w/o tracking rewards	84.3 $\pm$ 3.3	71.5 $\pm$ 0.2
w/o AMP reward	83.8 $\pm$ 2.0	71.8 $\pm$ 3.1
w/o tracking and AMP	training collapses*	

well (19.5–21.6%).

D. Object Dynamics

To answer Q3, we compare latent initializations and test lateral offsets and added delay.

a) Controller initialization: The BC-initialized latent-learning pipeline improves visual catch success by 5.2 percentage points over position/velocity weight mapping (Tab. VII), with a similar gain under privileged input; the visual gain holds in five of six seed pairs. Action matching establishes the latent control interface before joint encoder/decoder PPO refinement. Ridge probes with quadratic features on held-out flight ticks decode the delayed box position and velocity from **z** (R^2 0.94–0.99 and 0.90–0.99 across the six controllers) but not box size, orientation, or angular velocity ($R^2 \leq 0.13$), whereas the weight-mapped latent keeps position and velocity linearly ($R^2 \geq 0.98$) and retains some orientation (R^2 up to 0.58). A probe of the box's current, undelayed state (first controller of each configuration) shows that **z** partly compensates for observation age: for samples at least 80 ms old, it predicts the current position and velocity with RMSE 0.33 m and 1.5 m/s, against 0.6–0.7 m and 3.4 m/s for the delayed sample itself and 0.40 m and 2.1 m/s for the weight-mapped latent, which stays closer to the delayed observation.

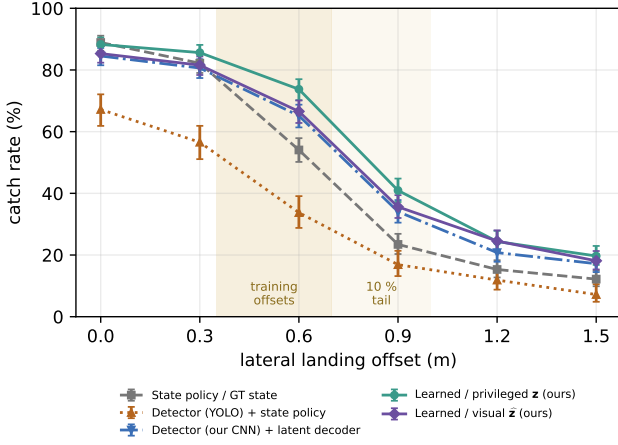


Fig. 4. **Catching across lateral landing offsets.** Catch rates for the ground-truth state-policy reference, the YOLO detector with the state policy, our CNN detector with the latent decoder, privileged latent input, and visual latent input. Shading indicates the training-offset region and its tail. Each point pools 640 throws for one policy checkpoint (320 throws for the YOLO curve). Visual latent input exceeds the ground-truth state policy by about 12 points at 0.6 and 0.9 m, whereas YOLO detection lowers the state policy’s catch rate at every reported offset.

b) *Lateral landing offset:* Figure 4 varies lateral offset for one checkpoint (640 throws per point). Visual success falls from 85.3% at 0 m to 66.6% at 0.6 m and 35.6% at 0.9 m; privileged-input rates are 88.3%, 73.8%, and 40.9%. The state policy with ground-truth state falls faster, from 88.9% to 54.1% and 23.4%, so the visual controller exceeds it by about 12 points at 0.6 and 0.9 m (five further controllers: $+12.5 \pm 8.9$ points at 0.6 m); with the YOLO detector the state policy reaches 67.2%, 33.8%, and 16.9%. The margin over the state policy arises in Stage 2 PPO (44.2% at 0.6 m before PPO; Tab. III), so Stage 1 is not an upper bound.

TABLE IV
LATENCY. HARDWARE TIMINGS (TOP) AND DELAY SENSITIVITY (BOTTOM). LATENCY RANDOMIZATION HELPS AT EVERY DELAY.

Hardware measurement	Latency (ms)		
	p50	p95	p99
Camera exposure $\rightarrow$ YOLO input	50.1	69.0	72.0
YOLO inference	8.2	12.9	14.9
Camera exposure $\rightarrow$ control tick	69.1	90.2	94.1
Control period (nominal 20 ms)	20.0	20.4	20.9

Catch rate (%), simulation	Added perception delay (ms)						
	0	20	40	60	80	120	160
Stage 3 / visual $\hat{z}$ (ours)	88.4	87.2	83.0	72.7	60.9	32.3	19.7
w/o latency randomization	80.8	79.1	71.7	65.8	50.8	27.0	11.1

p50/p95/p99: 50th/95th/99th percentiles. Hardware: detector-based pipeline (YOLO, Kalman filter, Stage 1 policy), excluding actuation. Simulation delays encoder inputs beyond zero-latency rendering (latent targets already lag 40 ms); each row: one training seed, 640 throws per point; reruns of a cell differ by up to 6 points.

c) *Added perception delay:* Tab. IV delays the visual encoder input in simulation to test the Stage 2 latency randomization. With visual input, catch success is 88.4%, 87.2%, and 83.0% with 0, 20, and 40 ms of added delay and 72.7% at 60 ms, versus 65.8% at 60 ms for the controller

trained without latency randomization (one training seed each), which is 5–11 points lower at every delay; beyond the training range, both decline (32.3% and 27.0% at 120 ms). The 60 and 80 ms columns bracket the 69 ms median of our detector-based hardware pipeline, so the zero-delay rates elsewhere are likely optimistic for hardware. With privileged input at matched total delay (added delay plus the 40 ms label lag), success falls from 88.3% at 40 ms to 76.4% at 120 ms, against 88.4% to 60.9% with visual input, so most of the visual decline arises in the encoder, whose training inputs are not delayed. The latency-randomized controller is most successful at 60 ms of delay (91.9% vs. 82.7% without delay; privileged input), whereas the one trained without delay peaks at zero (84.5%), consistent with the controller compensating for the latency it was trained with.

E. Visual Grounding

To answer Q4, we ablate the visual encoder and test appearance, distance, and a detector-based alternative.

TABLE V
VISUAL-ENCODER ABLATION. CATCH RATES (%), MEAN $\pm$ S.D. OVER FIVE POLICIES). REMOVING TEMPORAL HISTORY COSTS THE MOST.

Visual encoder	Catch rate (%), lateral offset (m)	
	0.0	0.6
Privileged latent $\hat{z}$ (reference)	87.9 $\pm$ 3.5	74.3 $\pm$ 4.5
Full visual encoder	85.2 $\pm$ 4.2	66.6 $\pm$ 8.9
<i>Inputs removed</i>		
w/o camera ego-motion	74.8 $\pm$ 3.8	44.0 $\pm$ 3.2
w/o temporal history (single frame)	49.6 $\pm$ 4.9	26.8 $\pm$ 2.4
<i>Training components removed</i>		
w/o DAGger		
(teacher rollouts only)	86.2 $\pm$ 2.8	67.6 $\pm$ 4.4
w/o box-state loss	85.0 $\pm$ 3.8	70.3 $\pm$ 4.1
w/o heatmap supervision	77.8 $\pm$ 5.5	55.7 $\pm$ 7.0
w/o latent regression		
(state regression instead)	66.5 $\pm$ 7.1	47.1 $\pm$ 2.8

a) *Visual inputs and supervision:* Tab. V evaluates visual inputs and training objectives over five trained policies per variant. Variants use one aggregation round, except teacher-rollout training without DAGger. A single-frame encoder, which cannot observe motion, loses 35.6 and 39.8 percentage points on the centered and 0.6 m cells, and regressing the 26-D privileged state for the frozen encoder instead of the latent loses 18.7 and 19.5. Removing camera ego-motion costs 10.4 and 22.6 points, and removing heatmap supervision 7.4 and 10.9. Teacher-rollout training alone achieves 86.2% and 67.6% at the two offsets, while removing the auxiliary box-state loss gives 85.0% and 70.3%. Both stay within seed variation, and zero or one aggregation round gives 86.2 and 85.2% on the centered cell, against 85.7% for the main encoders (two or three rounds) of the same five seeds, so neither component measurably helps in simulation. The state-regression variant must also predict orientation, angular velocity, and extents that $\mathbf{g}_t$ barely observes, so its gap does not isolate the latent target.

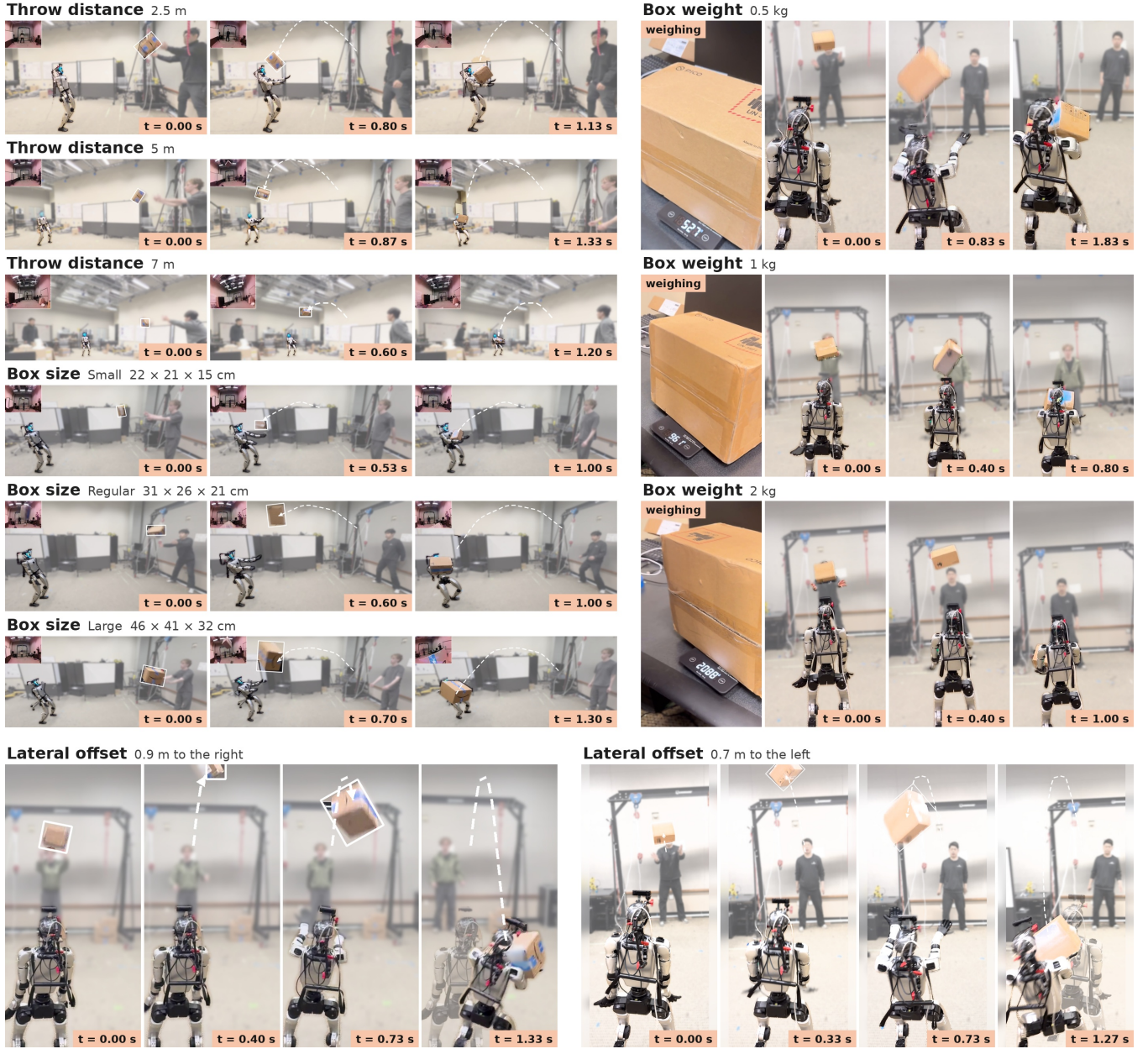


Fig. 5. **Real-robot catching across throw distances, box sizes, payloads, and lateral offsets.** Successful trials at 2.5, 5, and 7 m and with small, regular, and large boxes (left, each at release, during approach, and after the catch), with 0.5, 1, and 2 kg payloads (right; weighing views show the payload mass), and with lateral offsets of 0.9 m right and 0.7 m left (bottom). Insets show onboard camera views; timestamps indicate elapsed time within each sequence. White outlines mark box positions and dashed curves indicate flight direction. The same visual latent controller succeeds at every tested distance, box size, payload, and lateral offset; Tab. II reports catch rates for distance, size, and payload, and Sec. V-F the lateral-offset rate.

TABLE VI

COLOR AUGMENTATION. CATCH RATES (%), DECODER FIXED.
AUGMENTATION RAISES BLUE AND BLACK BOXES TO 88.0 AND 88.8%.

Box appearance	Catch rate (%)	
	Without color augmentation	With color augmentation
Cardboard	87.0	87.8
White	78.0	87.0
Grey	78.8	88.9
Red	87.3	88.8
Green	83.3	87.8
Blue	60.2	88.0
Black	61.2	88.8

TABLE VII

STAGE 2 INITIALIZATION. CATCH RATES (%), MEAN $\pm$ S.D. OVER SIX CONTROLLERS). ACTION MATCHING AVERAGES 5 POINTS HIGHER.

Latent initialization	Catch rate (%), latent input	
	Stage 2 / privileged $\mathbf{z}$	Stage 3 / visual $\hat{\mathbf{z}}$
Action matching (ours)	87.9 ± 3.2	85.7 ± 3.4
Weight mapping	82.8 ± 3.2	80.5 ± 2.9

b) *Box appearance:* With the decoder fixed, color augmentation raises success on blue/black boxes from 60.2/61.2% to 88.0/88.8%, while cardboard changes from 87.0% to 87.8% (Tab. VI). Across seven tested colors,

success is 87.0–88.9% in this evaluation run (86.6–87.8% on six colors in the separate run of Tab. I), demonstrating reduced appearance sensitivity.

c) *Learned latent versus detection plus filtering*: The visual gap grows with throw distance: at 5 m, privileged latent input matches the state policy (79.7%), whereas visual input reaches 43.0%, although the encoder still detects the box on every 5 and 7 m throw. For a like-for-like comparison with detection plus filtering, two baselines use our own color-augmented heatmap CNN (same rendered training data, camera, and resolution) and pass its per-frame 3D box point to a Kalman filter with gravity and drag whose speed bound, noise, gating, and output lag are selected on held-out throws; both receive a ground-truth arm trigger, which raises the arms earlier than ours. The filtered state drives either the state policy (“Detector (our CNN) + state-based” in Tab. I) or, encoded by E_ϕ , our decoder (“Detector (our CNN) + latent decoder”), which isolates the learned temporal model. Excluding the 5 and 7 m throws, which lie beyond the Stage 3 data, the visual latent outperforms the state-based pipeline in all 13 conditions of Tab. I (by 5.0–15.6 points) and the latent-decoder pipeline in 14 of 18 conditions of Tab. I and Fig. 4 (mean +1.5 points, largest for box size; with the same trigger, 88.3 vs. 84.5% at the reference and 69.8 vs. 65.2% at 0.6 m; across five further controllers at 0 and 0.6 m the margin is $+0.3 \pm 5.2$ points, ahead in 7 of 10 cells), so its advantage over detection plus filtering comes from the Stage 2 latent interface, while the learned filter is on par with a tuned Kalman filter. At 5 and 7 m the filtered pipelines catch more (67.7 and 33.9% for the latent decoder), but this reflects the Stage 3 data range: adding 2.5–7 m throws to Stage 3 raises visual success to 75.8 and 36.7%, with 88.8% at the reference. The “Detector (YOLO)” row reproduces our previously deployed perception node (real-photo weights, hardware depth and size checks, untuned filter) and serves only as a reference.

F. Real-Robot Evaluation

To test the conditions of Q2–Q4 on hardware, we evaluate catches of human-thrown boxes with different sizes, payloads, throw distances, and lateral offsets. Unless the size is varied, real-robot trials use the $31 \times 26 \times 21$ cm regular box.

Figure 5 shows qualitative catches across these conditions. Human throwers cannot place the box at a precise lateral offset; of 20 throws aimed to either side (e.g., 0.9 m right and 0.7 m left in Fig. 5), the G1 caught 8.

a) *Hardware runtime*: Tab. IV reports measurements from two hardware sessions of our detector-based pipeline (1274 box measurements and approximately 33,000 control ticks). The 95th-percentile camera-to-control latency is 90.2 ms, and the 50 Hz control period exceeds 25 ms on 0.015% of ticks.

VI. DISCUSSION AND LIMITATIONS

The decomposition lets visual grounding be learned with the controller fixed and keeps rendering out of reinforcement learning, so only about 1.6M rendered steps are needed.

During latent PPO, retaining the catching skill requires the task rewards, chiefly hold shaping, and at least one motion prior; payloads up to 2 kg remain catchable, whereas boxes outside the reference size are often intercepted but not retained, and torso excursion separates retention from a settled posture. For object dynamics, BC initialization improves visual catching by 5.2 points on average, and in a single-seed comparison, latency randomization adds 5–11 points across simulated delays. For visual grounding, temporal history and ego-motion contribute most, regressing the learned latent gives higher success than the evaluated full-state regression baseline, and, with the same detector, the learned latent outperforms detection plus Kalman filtering within its visual training range and, once Stage 3 data cover them, also on 5 and 7 m throws. The main limitation is long-range visual grounding: on 5 m throws, faster than any training throw and beyond the 3.0–3.5 m Stage 3 data, visual success falls to 43.0% while privileged input reaches 79.7%, and real-robot success falls to 40% and 15% at 5 and 7 m; adding far throws to the Stage 3 data raises simulated 5 m success to 75.8%, not yet tested on hardware. Further limitations: a single task with human throws and 20 real trials per condition (intervals up to about ± 20 points); hardware latency was measured only on our detector-based pipeline, where it exceeds the randomized range; the controller is effectively size-blind; and rendering omits motion blur and exposure dynamics.

VII. CONCLUSION

Reflex learns visual reactive whole-body catching on a humanoid by decoupling whole-body control, object dynamics, and visual grounding into three stages linked by an object-dynamics latent. In simulation, visual control reaches 85.7% catch success versus 87.9% with privileged input, and on a real Unitree G1 it catches human-thrown boxes at 76% and 59% of human success at 2.5 and 5 m. Extending this decomposition to other visual reactive whole-body tasks is a natural next step.

REFERENCES

- [1] S. Kim, A. Shukla, and A. Billard, “Catching objects in flight,” *IEEE Trans. Robot.*, vol. 30, no. 5, pp. 1049–1065, 2014.
- [2] S. Abeyruwan *et al.*, “Agile catching with whole-body MPC and blackbox policy learning,” in *Proc. 5th Annu. Learn. Dyn. Control Conf. (LADC)*, ser. PMLR, vol. 211, 2023, pp. 851–863.
- [3] Y. Zhang, T. Liang, Z. Chen, Y. Ze, and H. Xu, “Catch It! learning to catch in flight with mobile dexterous hands,” in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, 2025, pp. 14 385–14 391.
- [4] J. Ren *et al.*, “Humanoid goalkeeper: Learning from position conditioned task-motion constraints,” *arXiv:2510.18002*, 2025.
- [5] Y. Wang *et al.*, “HumanX: Toward agile and generalizable humanoid interaction skills from human videos,” *arXiv:2602.02473*, 2026.
- [6] B. Hove and J.-J. E. Slotine, “Experiments in robotic catching,” in *Proc. Amer. Control Conf. (ACC)*, 1991, pp. 380–386.
- [7] L. Yan, T. Stouraitis, J. Moura, W. Xu, M. Gienger, and S. Vijayakumar, “Impact-aware bimanual catching of large-momentum objects,” *IEEE Trans. Robot.*, vol. 40, pp. 2543–2563, 2024.
- [8] F. Lan *et al.*, “DexCatch: Learning to catch arbitrary objects with dexterous hands,” *arXiv:2310.08809*, 2023.
- [9] B. Huang *et al.*, “Dynamic handover: Throw and catch with bimanual hands,” in *Proc. Conf. Robot Learn. (CoRL)*, ser. PMLR, vol. 229, 2023, pp. 1887–1902.
- [10] K. Dong, K. Pereida, F. Shkurti, and A. P. Schoellig, “Catch the ball: Accurate high-speed motions for mobile manipulators via inverse dynamics learning,” in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, 2020, pp. 6718–6725.
- [11] Y. Wang *et al.*, “Learning vision-driven reactive soccer skills for humanoid robots,” *Sci. Robot.*, vol. 11, p. ead1152, 2026.
- [12] J. Ren *et al.*, “SMASH: Mastering scalable whole-body skills for humanoid ping-pong with egocentric vision,” *arXiv:2604.01158*, 2026.
- [13] Y. Ma, A. Cramariuc, F. Farshidian, and M. Hutter, “Learning coordinated badminton skills for legged manipulators,” *Sci. Robot.*, vol. 10, no. 102, p. eadu3922, 2025.
- [14] B. Büuml, O. Birbach, T. Wimböck, U. Frese, A. Dietrich, and G. Hirzinger, “Catching flying balls with a mobile humanoid: System overview and design considerations,” in *Proc. IEEE-RAS Int. Conf. Humanoid Robots (Humanoids)*, 2011, pp. 513–520.
- [15] B. Forrai, T. Miki, D. Gehrig, M. Hutter, and D. Scaramuzza, “Event-based agile object catching with a quadrupedal robot,” in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, 2023, pp. 12 177–12 183.
- [16] S. Hu, Q. Liao, Y. Chi, K. Zakka, Y. S. Shao, P. Abbeel, and K. Sreenath, “Workhorse: Learning robust whole-body humanoid loco-manipulation from human data,” *arXiv:2610.09117*, 2026.
- [17] X. B. Peng, P. Abbeel, S. Levine, and M. van de Panne, “DeepMimic: Example-guided deep reinforcement learning of physics-based character skills,” *ACM Trans. Graph.*, vol. 37, no. 4, pp. 143:1–143:14, 2018.
- [18] X. B. Peng, Z. Ma, P. Abbeel, S. Levine, and A. Kanazawa, “AMP: Adversarial motion priors for stylized physics-based character control,” *ACM Trans. Graph.*, vol. 40, no. 4, 2021.
- [19] J. Merel *et al.*, “Catch & Carry: Reusable neural controllers for vision-guided whole-body tasks,” *ACM Trans. Graph.*, vol. 39, no. 4, pp. 39:1–39:14, 2020.
- [20] T. He *et al.*, “Learning human-to-humanoid real-time whole-body teleoperation,” in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, 2024, pp. 8944–8951.
- [21] Q. Liao *et al.*, “BeyondMimic: From motion tracking to versatile humanoid control via guided diffusion,” *Sci. Robot.*, vol. 11, no. 117, p. eadx8924, 2026.
- [22] Z. Luo *et al.*, “SONIC: Supersizing motion tracking for natural humanoid whole-body control,” *Sci. Robot.*, vol. 11, no. 117, p. ead4592, 2026.
- [23] J. Dao, H. Duan, and A. Fern, “Sim-to-real learning for humanoid box loco-manipulation,” in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, 2024, pp. 16 930–16 936.
- [24] L. Yang *et al.*, “OmniRetarget: Interaction-preserving data generation for humanoid whole-body loco-manipulation and scene interaction,” in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, 2026, *arXiv:2509.26633*.
- [25] H. Weng *et al.*, “HDMI: Learning interactive humanoid whole-body control from human videos,” *arXiv:2509.16757*, 2025.
- [26] H. Wang *et al.*, “PhysHSI: Towards a real-world generalizable and natural humanoid-scene interaction system,” *arXiv:2510.11072*, 2025.
- [27] S. Zhao *et al.*, “ResMimic: From general motion tracking to humanoid whole-body loco-manipulation via residual learning,” *arXiv:2510.05070*, 2025.
- [28] C. Zheng *et al.*, “Embracing bulky objects with humanoid robots: Whole-body manipulation with reinforcement learning,” *arXiv:2509.13534*, 2025.
- [29] D. Chen, B. Zhou, V. Koltun, and P. Krähenbühl, “Learning by cheating,” in *Proc. Conf. Robot Learn. (CoRL)*, ser. PMLR, vol. 100, 2020, pp. 66–75.
- [30] L. Pinto, M. Andrychowicz, P. Welinder, W. Zaremba, and P. Abbeel, “Asymmetric actor critic for image-based robot learning,” in *Proc. Robot.: Sci. Syst. (RSS)*, 2018.
- [31] 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/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, 2017, pp. 23–30.
- [32] S. Ross, G. Gordon, and D. Bagnell, “A reduction of imitation learning and structured prediction to no-regret online learning,” in *Proc. 14th Int. Conf. Artif. Intell. Statist. (AISTATS)*, ser. PMLR, vol. 15, 2011, pp. 627–635.
- [33] J. Lee, J. Hwangbo, L. Wellhausen, V. Koltun, and M. Hutter, “Learning quadrupedal locomotion over challenging terrain,” *Sci. Robot.*, vol. 5, no. 47, p. eabc5986, 2020.
- [34] A. Kumar, Z. Fu, D. Pathak, and J. Malik, “RMA: Rapid motor adaptation for legged robots,” in *Proc. Robot.: Sci. Syst. (RSS)*, 2021.
- [35] A. Agarwal, A. Kumar, J. Malik, and D. Pathak, “Legged locomotion in challenging terrains using egocentric vision,” in *Proc. Conf. Robot Learn. (CoRL)*, ser. PMLR, vol. 205, 2023, pp. 403–415.
- [36] T. He *et al.*, “VIRAL: Visual sim-to-real at scale for humanoid loco-manipulation,” in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2026.
- [37] H. Xue *et al.*, “Opening the sim-to-real door for humanoid pixel-to-action policy transfer,” *arXiv:2512.01061*, 2025.
- [38] Z. Wu *et al.*, “Perceptive humanoid parkour: Chaining dynamic human skills via motion matching,” *arXiv:2602.15827*, 2026.
- [39] S. Zhao *et al.*, “LadderMan: Learning humanoid perceptive ladder climbing,” *arXiv:2606.05873*, 2026.
- [40] NVIDIA, M. Mittal, P. Roth, J. Tigue, A. Richard, *et al.*, “Isaac Lab: A GPU-accelerated simulation framework for multi-modal robot learning,” *arXiv:2511.04831*, 2025.
- [41] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, “Proximal policy optimization algorithms,” *arXiv:1707.06347*, 2017.