METATRON: A Promptable Whole-Body Execution Layer for Humanoids

Chen Tessler[1,*], Yifeng Jiang[1,*], Alec Otto[2], Yi Shi[1,†],
Umar Iqbal[1], Gal Chechik[1], Xue Bin Peng[1]

Metatron bridges divine will and the physical world—
an intermediary through whom intent becomes action.

Inspired by Orlov (2005) and Scholem (1941) [1], [2].

Interactive demo: `https://nvlabs.github.io/ProtoMotions/demos/metatron/`

***Abstract*— Humanoid controllers often require task-specific training despite sharing the need for balance and whole-body coordination. We present METATRON, a promptable whole-body execution layer combining a generative base policy with a programmable skill layer. The base policy receives partial goals, collectively called a prompt, that can combine timed body-part targets, velocity commands, and style cues. It produces diverse whole-body motions consistent with these goals. Training combines motion capture, synthetic recovery trajectories, and retargeted scene-interaction motions. After training, the policy is frozen. Stateful programs called skills express task intent as prompts at each control step, drawing rough geometric sketches that the policy turns into coordinated movement. Skills are ordinary code, refined by coding agents that analyze motion data and fit compact rules without task-specific policy training. On the physical Unitree G1, one frozen checkpoint supports walking, recovery, teleoperated object pickup, and sitting on furniture. Comparative evaluations show higher reaching success and push robustness than the tested reference-tracking pipelines. The same recipe produces climbing and acrobatics on a simulated character. METATRON enables coding agents to turn learned motor capabilities into new skills through code, without further policy training.**

I. INTRODUCTION

Consider a humanoid that must walk to a table in a particular style, grasp a cup at a designated moment, recover after a shove, present it to a human, and shake hands. Today this typically requires separate learned controllers for locomotion [3], manipulation [4], recovery [5], compliant control [6], and motion tracking [7]. These controllers duplicate capabilities such as balance and coordination, and transitions between them demand additional engineering.

We argue that a more scalable recipe is to separate *what* a motion should accomplish from *how* to coordinate the body: train one general-purpose whole-body policy with a flexible goal-conditioning interface, and let task-specific programs specify the goals. The policy becomes an execution layer that provides physical coordination; the programs become the controllers that supply intent. Each program specifies a partial motion goal that we call a *prompt*: a sitting command specifies the shape and location of a nearby object, while a style cue biases the generated motion toward example poses.

We present METATRON, a promptable whole-body execution layer for humanoids (Figure 1). It combines a capable base policy with a programmable skill layer. The base policy builds on MaskedMimic [8], which trains a dense motion tracker and distills it into a generative policy that accepts sparse body-part targets at specified future times.

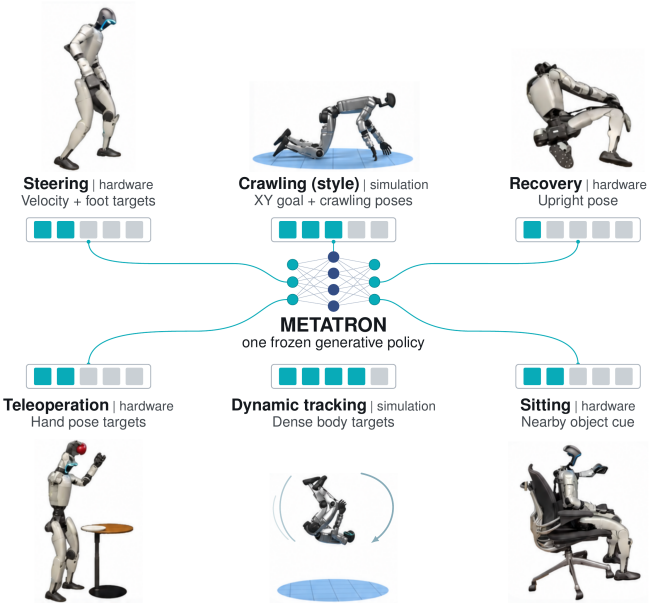


Fig. 1: **One frozen controller per embodiment, diverse behaviors.** Partial prompts specify goals; each controller provides whole-body coordination across tasks without task-specific training. Teal and gray blocks indicate specified and unspecified goal components, respectively.

Once trained, the policy is frozen and a runtime layer runs programs called *skills* that convert task intent into prompts at each control step (Section IV). Skills are ordinary code, authored by humans or coding agents. We show how LLM agents analyze motion clips and fit compact rules that capture shared motion characteristics, turning task design into a software problem. Reinforcement learning trains the motion experts; on-policy distillation teaches the base policy to imitate them under partial goals, including at states reached by the student itself. This training supports approximate goal sketches rather than requiring exact reference trajectories.

METATRON also extends MaskedMimic in several directions. Forward kinematics reconstructs local body poses from joint and inertial measurements, without external motion

[1]NVIDIA. [2]Dollhouse Robotics. *Equal contribution. †Work done while interning at NVIDIA.

capture; world-position goals additionally require localization. Scene elements are represented as geometric primitives specified by measured dimensions and poses. Style conditioning allows operators and skills to select the generated motion's character. We augment the training data with synthetic push-recovery and get-up trajectories, extending the repertoire beyond what motion capture alone provides. On the modeling side, spherical projection bounds the stochastic latent, while predictive and temporal auxiliary losses regularize its fused representation (Section III-C).

Our contributions are a whole-body policy with a flexible prompt interface, training data augmented with scene interactions and synthetic recoveries (Section V), and a skill layer whose code can be refined by coding agents without policy retraining (Section IV). One frozen checkpoint per embodiment supports the demonstrated tasks on physical G1 and simulated SOMA. Separate G1 comparisons show higher whole-body reaching success and push robustness than the tested reference-tracking pipelines (Section VI).

II. Related Work

Physics-based motion imitation: Reference tracking has progressed from motion-specific policies [9] to controllers that follow large and diverse motion collections [10], [11], [12]. Dense tracking provides a strong training signal for coordinated motion, but a tracking interface still requires a reference trajectory. METATRON uses this supervision to distill a policy that accepts partial goals.

Partial-goal and generalist motion control: Latent generative controllers such as ASE, CALM, and PULSE learn reusable motion representations [13], [14], [15]. GPC [16] learns a generative controller over discrete motion codes. Such representations can support downstream tasks through task-specific policy training. MaskedMimic [8] instead trains a controller directly on partial motion descriptions, supporting multiple command types and transitions between them in simulation. METATRON extends this approach with a larger training set including synthetic recovery motions and retargeted scene interactions, a latent design that separates intent from solution, a skill layer that makes task logic explicit as ordinary code refined by coding agents, and body-pose reconstruction from robot sensors for physical deployment (Sections III and IV). Other distillation-based controllers [17], [18], [19] likewise distill one policy for many command types, and concurrent generalist trackers scale a similar partial-goal interface [20], [21]. Our emphasis is on combining optional goal streams with explicit skill programs: humans and coding agents refine the prompts without task-specific policy training.

From animation to robotics: Large-scale motion tracking and whole-body teleoperation demonstrate that human motion can supervise versatile robot control [11], [12], [22]. BeyondMimic [7] further combines tracking with latent diffusion and test-time guidance for new objectives. Guidance adjusts generated motion to meet an objective at runtime. Our policy instead learns to respond directly to partial goals: skills modify the prompt, and one prior-and-decoder pass produces joint targets at each control step, without task-specific optimization of the generated motion.

III. Metatron: Base Policy

Training proceeds in two phases. Dense motion trackers learn whole-body coordination from complete reference motions using privileged simulator state (Section III-A). Masked on-policy distillation then trains one generative policy per embodiment, conditioned on partial prompts through a flexible interface (Sections III-B and III-C). Distillation can combine multiple specialist trackers, with each frozen expert labeling states from its own training group. The deployable policy $\pi_\theta(a_t \mid o_t, a_{t-1}, g_t)$ maps a prompt g_t, current observations and sensor history o_t, and the previous action a_{t-1} to joint actions using only deployment-available inputs. Once frozen, stateful programs called *skills* construct prompts to express diverse behaviors without retraining (Section IV).

A. Dense Motion Tracker

An MLP policy learns whole-body coordination from complete reference motions:

$$a_t^{\mathrm{trk}} = \pi^{\mathrm{trk}}\big(s_t^{\mathrm{trk}}, y_t, a_{t-1}\big), \tag{1}$$

where s_t^{trk} includes privileged simulator state and y_t encodes complete reference states at a configured set of future offsets. We optimize the tracker with PPO [23]. Its reward combines tracking terms [9] and action-rate regularization [7]:

$$r_t = \sum_k w_k \exp(-\lambda_k e_{k,t}) - w_a \mathcal{R}_{\mathrm{act},t} + r_t^{\mathrm{extra}}. \tag{2}$$

The errors $e_{k,t}$ measure body position, orientation, and velocity tracking; w_k and λ_k set their weights and sensitivities. The action regularizer $\mathcal{R}_{\mathrm{act},t}$ penalizes changes between consecutive commands, and r_t^{extra} collects selected-body precision terms and embodiment-specific penalties. These rewards train the trackers, not the distilled student. The supplementary material details the G1 BONES and SOMA experts' rewards, architectures, and optimization settings.

B. Prompt Interface

To support partial goals at runtime, we distill the frozen trackers into METATRON. At timestep t, a prompt combines four optional goal streams and scene context,

$$g_t = \left(\mathcal{C}_t^w, \mathcal{C}_t^\ell, u_t, \xi_t, b_t\right). \tag{3}$$

Masks identify which bodies, properties, and timesteps are constrained, so an omitted goal can remain unspecified.

Spatio-temporal goals: $\mathcal{C}_t^w$ specifies body positions or orientations in world coordinates, each with a time-to-arrival. Position and orientation constraints are independently optional: a prompt can request a planar location while leaving height free, specify a timed hand target, or supply dense whole-body poses.

Canonicalized pose goals: $\mathcal{C}_t^\ell$ describes the desired pose independently of its world placement. Horizontal positions are relative to the target torso and aligned with its heading; heights

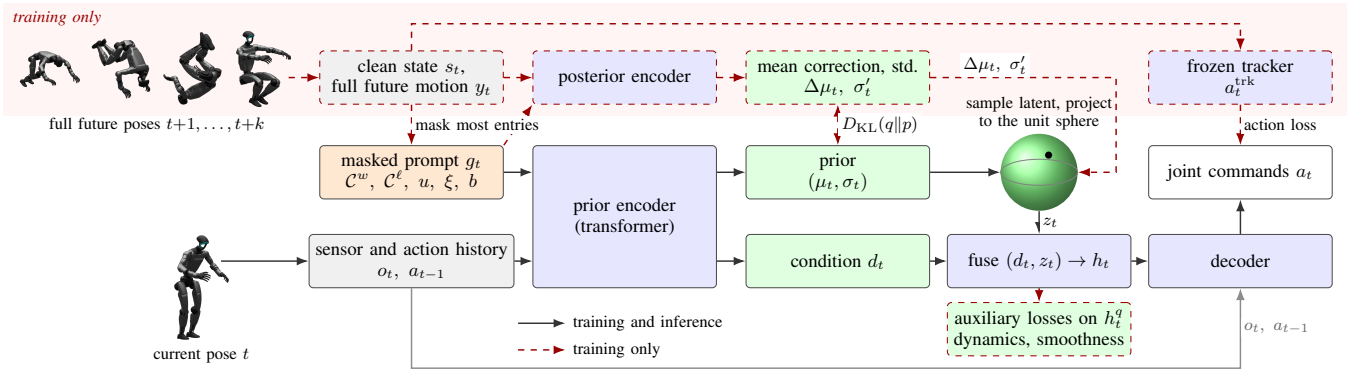


Fig. 2: **Model and training.** The prior maps partial prompts and observation history to a deterministic condition and a latent distribution. The decoder combines the condition with a sphere-projected latent to produce joint targets. Training adds a posterior conditioned on the full reference and supervision from a frozen tracker. Only the solid path runs at deployment. Pose renders illustrate state, not camera inputs.

are measured above the target pose's lowest body point. A raised-hand pose therefore retains its meaning regardless of world position or heading. These goals are not offsets from the robot's current pose.

Velocity: u_t specifies planar travel velocity in the agent's heading frame, without specifying an endpoint.

Style cue: ξ_t specifies the character of the motion through an example clip. A small set of frames provides canonicalized body poses and optional heading-aligned velocities. Pooling these frames into one token discards temporal order: the cue describes example motion states without assigning arrival times. A location goal can therefore be paired with running or crouched-walking examples without prescribing their trajectory.

Scenes: For object interaction, each scene element is decomposed offline into bounding boxes b_t (e.g. a sofa yields seat and backrest boxes). The policy observes the eight corner positions of each box in the agent's heading frame, from which both position and orientation are recoverable. This compact, fixed-width representation needs only pre-measured dimensions and an object-pose estimate at deployment.

C. Generative Policy

A partial prompt admits many valid motions. Regressing a single action directly from that prompt can average incompatible motion completions [8]. We instead use a conditional variational autoencoder (VAE): a privileged posterior uses the full reference during training, while a prior uses only deployment-available observations and the prompt. A deterministic condition d_t encodes the observed context; a stochastic latent selects a completion. We project this latent onto the unit sphere [18] before combining it with d_t. With $x_t = (o_t, a_{t-1})$, privileged simulator state s_t, and complete reference y_t, the distributions before projection are

$$\begin{aligned} p_\theta(\tilde{z}_t \mid x_t, g_t) &= \mathcal{N}(\mu_t, \mathrm{diag}(\sigma_t^2)), \\ q_\phi(\tilde{z}_t \mid x_t, g_t, s_t, y_t) &= \mathcal{N}(\mu_t + \Delta\mu_t, \mathrm{diag}(\sigma_t'^2)). \end{aligned} \tag{4}$$

The prior transformer combines sensor and prompt tokens to predict d_t, μ_t, and σ_t; masks exclude unavailable goals (Figure 2). A separate training-only posterior encoder predicts the mean correction $\Delta\mu_t$ and its own variance from privileged state, reference, and prompt features. For either distribution, a sample is projected as $z_t = \tilde{z}_t / \|\tilde{z}_t\|_2$ and fused with d_t as $h_t = f_\theta(d_t, z_t)$. The decoder maps the current observations, previous action, and h_t to an action. Superscripts p and q denote quantities computed with prior and posterior samples, respectively. A training-only predictor F estimates the next condition from h_t^q, and a temporal consistency loss smooths consecutive fused representations [24], [25].

D. Training

Distillation collects states under a mixture of posterior-guided student and frozen-tracker rollouts. The tracker labels each visited state using its complete reference, while the student receives randomly masked prompts. The student minimizes

$$\begin{aligned} \mathcal{L} = \mathbb{E}\big[\ell(a_t^q, a_t^{\text{trk}}) + \beta_e D_{\text{KL}}(q_\phi \| p_\theta)\big] + \lambda_\mu \mathbb{E}(\|\mu_t\|_2 - 1)^2 \\ + \lambda_{\text{dyn}} \mathbb{E}\big[\ell(F(h_t^q), \mathrm{sg}(d_{t+1}))\big] + \lambda_{\text{tc}} \mathbb{E}\big[\ell(h_t^q, \mathrm{sg}(h_{t-1}^q))\big] \\ + \lambda_{\text{act}} \mathbb{E}\big[\ell(a_t^q, \mathrm{sg}(a_{t-1}^q))\big], \end{aligned} \tag{5}$$

where a_t^q is the posterior-decoded student action, ℓ averages squared error over coordinates, and sg treats the target as a constant during optimization. The first term imitates the frozen tracker; KL aligns the prior and posterior Gaussians before projection. The remaining terms encourage a unit-norm prior mean, predict the next condition, smooth consecutive representations, and smooth consecutive student actions. Temporal terms use only within-episode pairs, excluding resets. The schedule β_e balances reconstruction and prior–posterior alignment. For G1, cyclical annealing [26] repeatedly relaxes then strengthens this constraint; SOMA uses a single ramp followed by a constant weight. The supplementary material gives the schedules, reductions, coefficients, and SOMA's additional diversity loss. At deployment, only the prior, fusion module, and decoder are needed.

IV. Metatron: Skill Layer

An underspecified prompt can produce several valid motions, some less stable or desirable than others. A velocity

command alone may yield foot stutter or hesitant steps. A *skill* is a stateful program that constructs and updates prompts at each control step, supplementing high-level intent with additional goals that guide the policy toward the desired behavior. The *harness* runs the active skill and manages transitions between skills.

Because the policy tolerates approximate goals (Section III), a skill does not need to plan precise trajectories. Instead, it draws a **rough sketch of the desired motion**: a few geometric targets such as static points, straight lines, or simple periodic curves. The policy renders this sketch into coordinated whole-body movement, handling balance, weight transfer, and joint actions. A sketch with only a handful of parameters suffices; a separately trained motion planner is not required.

Authoring a skill follows a common pattern (Figure 3). A human writes a first-pass rule describing goals required by the task. For simple cases such as compliance or recovery, this rule is already sufficient. When visible artifacts remain, such as inefficient stepping during locomotion or hesitant weight shifts when rising from a chair, the human describes these issues and a coding agent takes over refinement. The agent constructs an evaluation playground that spans the relevant variation: different furniture geometries and approach angles for sitting, or randomized velocity sequences for steering. It then analyzes motion-capture clips and fits compact rules that capture shared characteristics across the corpus. The agent scores each candidate on the playground, identifies failing cases, and iterates.

We showcase example skills constructed by the agent below:

Steering: A steering command specifies velocity and heading but leaves stepping unspecified. With only these constraints, the policy sometimes produces small, high-frequency steps that waste energy and transfer unsafely to hardware. We used coding agents to fit a single periodic foot-trajectory rule from locomotion clips in a benchmark of randomized velocity sequences. The rule is governed by two quantities: travel speed v and the angle α between travel direction and the robot's facing. At future time τ, the foot target in the planned torso's heading frame is

$$\mathbf{p}_i(\tau) = \mathbf{b}_i + \begin{bmatrix} A(v)\cos\alpha\cos\phi_i \\ W\left(s_i|\sin\alpha| + |\cos\alpha|\sin\alpha\cos\phi_i\right) \\ H\max(0, -\sin\phi_i) \end{bmatrix}, \quad (6)$$

where $\mathbf{b}_i$ is the calibrated stance offset, $s_R = -1$, $s_L = 1$, and phase $\phi_i = \phi_t + 2\pi f(v)\tau + \delta_i$ alternates between feet ($\delta_R = 0$, $\delta_L = \pi$). The fitted functions $A(v)$ and $f(v)$ set swing amplitude and cadence; W and H control lateral widening and foot lift. Each foot swings with the travel command, lifts at mid-swing, and returns to a stance that widens as travel turns lateral. Cadence and step length both grow with speed, producing longer, quicker strides rather than faster shuffling. Forward walking, side-stepping, and turning emerge from this rule at different values of α. Fitted coefficients and deployment details are provided in the supplementary material.

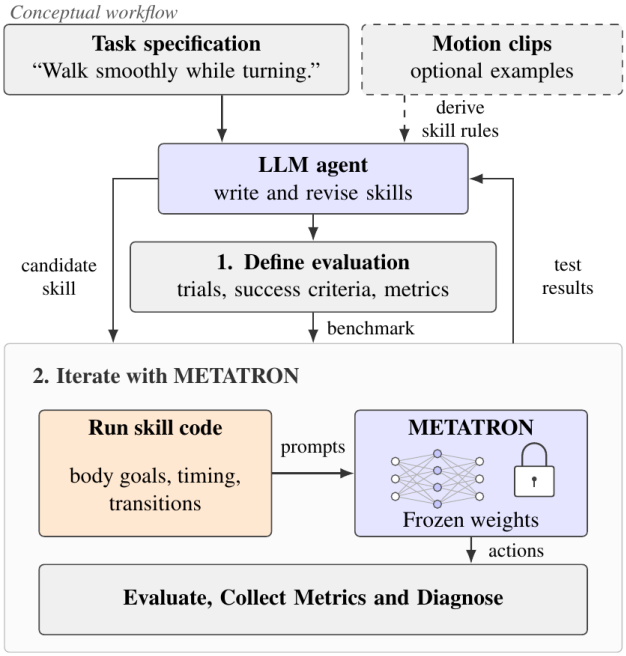


Fig. 3: **Agent-assisted skill development.** Given a task specification and optional motion examples, a human and LLM agent define evaluation scenarios and criteria. The LLM agent then iteratively refines the skill code using simulation feedback. Skills specify goals for the frozen METATRON policy, enabling the model to perform new tasks without additional training.

Sitting: Sitting on furniture requires only coarse goals derived from the seat's bounding-box geometry. A coding agent fit the goal rule from a benchmark of seated interactions. The result is three static target points: a torso target at a fixed height above the seat surface, slightly behind the seat center, and both feet just past the front edge at stance width. All prompt slots receive these three targets, without intermediate waypoints. The policy coordinates turning and stepping before sitting as needed. Standing reverses the pattern: the skill raises the torso target along a straight line from seated to standing height while marching it continuously forward. No stepping pattern is prescribed, yet the policy produces coordinated steps to rise and walk away.

Recovery: Recovery returns the robot from a fallen configuration to an upright pose. A hand-authored rule is sufficient: the simulation skill specifies a canonicalized upright pose, while the G1 deployment skill constrains only torso height. Arrival time scales with the remaining height deficit. A lower entry threshold and higher exit threshold prevent repeated switching near the boundary. Automatic recovery is disabled during dense tracking and unconditional generation; the supplementary material gives the thresholds and target construction.

Interactive compliance: Physical human-robot interaction requires maintaining stance while accommodating human-guided arm motion. A hand-authored rule is again sufficient. The skill commands zero travel velocity and constrains torso

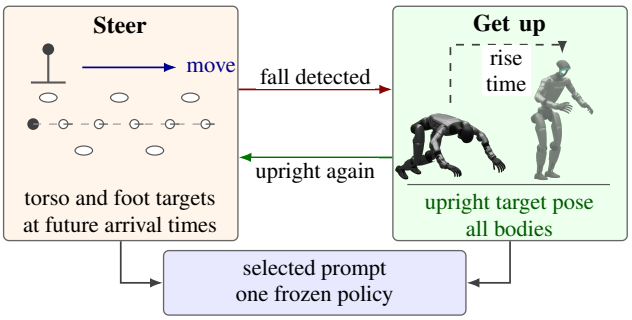


Fig. 4: **Change the prompt, retain the controller.** In this simulation skill, a detected fall switches steering goals to a whole-body upright pose; steering resumes once upright. G1 deployment instead requests torso height alone.

height and heading, leaving arm poses unspecified. Unlike full-pose tracking, these prompts do not require the arms to return to a prescribed pose, so the policy treats externally induced motion as consistent with the prompt. In our demonstrations, this produces handshakes and guided manipulation without a separate compliance controller.

These skills re-purpose existing behaviors into new capabilities without requiring any skill-specific training.

V. Training Data

For each embodiment, motion-capture data, scene interactions, and synthetic recoveries provide supervision for the distilled policy (section III-C). G1 partitions these sources among four specialist trackers (section III-A): BONES, AMASS+LAFAN, recovery, and SAMP. Kinematic retargeting adapts motions to the G1 morphology; scene interactions additionally require collision handling, while recoveries are generated directly for each embodiment [27].

Motion-Capture Retargeting: Our base dataset is a large-scale motion-capture library [28] spanning locomotion, dance, acrobatics, sports, and everyday activities.

Scene-Interaction Retargeting: Scene interactions require preserving body–object contacts as well as the motion. We use SAMP [29], which provides human motions such as sitting on furniture and lying on beds, represented with the SMPL body model. Kinematic retargeting alone can introduce furniture collisions or distort the posture to avoid them.

We use two-phase optimization. The first phase optimizes keypoint alignment, joint limits, smoothness, and foot–ground contact without scene geometry. The second adds a scene-collision cost and a joint-anchor penalty, favoring root translation over changes to the joint configuration. We represent furniture with an oriented surface point cloud, filtering interior surfaces per motion. The cost penalizes penetration depth but not contact, allowing contact without hovering. We will release the retargeted SAMP dataset upon publication.

Synthetic Recovery Motions: Recovery motions must accommodate the embodiment's dynamics and ground contacts. We therefore generate them directly using each simulated embodiment. We extend Hybrid Imitation Learning (HIL) [30] with MaskedMimic-style masking [8] of body targets and arrival times. We use this policy to record recoveries from randomized fallen poses. Collection and filtering settings are given in the supplementary material.

VI. Experiments

A. Experimental Setup

Our implementation builds on ProtoMotions [31]. We train separate METATRON policies for SOMA and G1. The G1 policy distills four specialist trackers covering BONES, AMASS+LAFAN, recovery, and SAMP; the SOMA policy distills a scene-aware tracker.

Training: Trackers are trained before distillation, and skills use the resulting frozen METATRON policy without retraining. The G1 demonstrations use the four-expert policy. The reference-pipeline comparisons use a separate policy trained without synthetic recovery data; the ablation models are trained separately under a matched budget and parameter counts. The supplementary material lists hardware, training budgets, and hyperparameters.

Deployment: Our C++ deployment stack compiles the base policy with TensorRT and CUDA graphs to run on a laptop with an NVIDIA GPU, which the robot connects to via Ethernet cable. An Xbox controller and a PICO headset provide mode selection, steering, and hand-pose commands. All hardware demonstrations use deterministic prior-mean inference. Automatic recovery uses FK-derived torso height and the state machine described in Section IV.

B. Hardware Demonstrations

The physical G1 uses one frozen METATRON policy for all demonstrated control modes (Figures 5 and 7). Switching modes changes only the goal observations and masks; model weights and sensor processing remain unchanged.

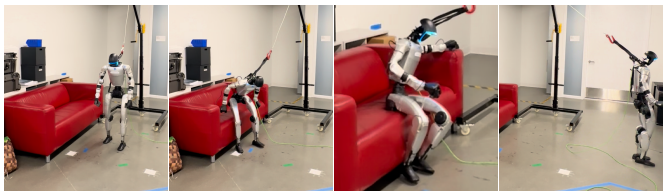
Steering and gait refinement: We use an Xbox controller for interactive steering. One thumbstick controls heading, the other sets travel direction and speed. The steering skill adds timed foot-placement goals that produce more decisive steps and markedly improve walking quality, especially on hardware. Without these goals, the policy produces less committed foot swings and more hesitation. No policy retraining is required (Section IV).

Teleoperation: A PICO headset provides steering via thumbsticks and maps the user's relative hand motions to target locations for the robot's hands. This model enables the user to direct the robot to pick objects, such as an apple.

Sitting on objects: A skill can also be designed to enable the physical G1 to sit on a sofa and stand back up (Figure 5). The robot is positioned near the sofa and the seat's bounding box is provided as an input condition. The skill then generates torso and foot targets.

C. Recovery and Compliance

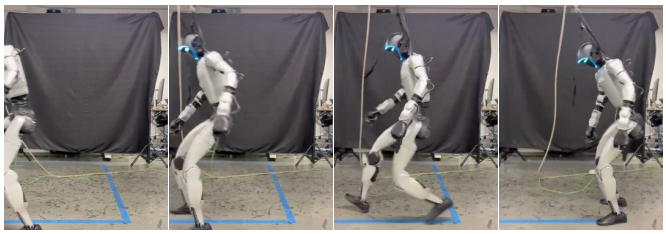
The generative controller can be prompted to exhibit human-like responsive behaviors. Temporally sparse goals leave flexibility in the motion between targets, allowing



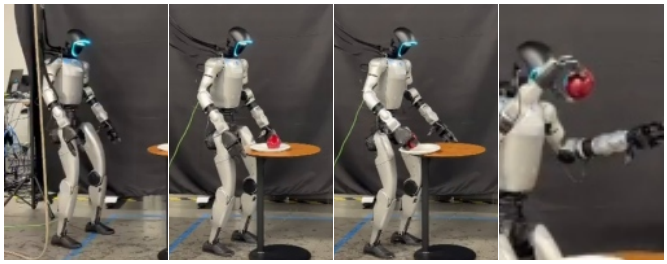
Scene interaction: G1 turns and sits on a sofa, then stands up and walks away.



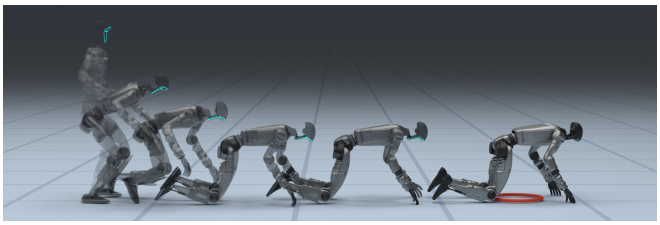
Compliance: Temporally sparse goals allow guidance.



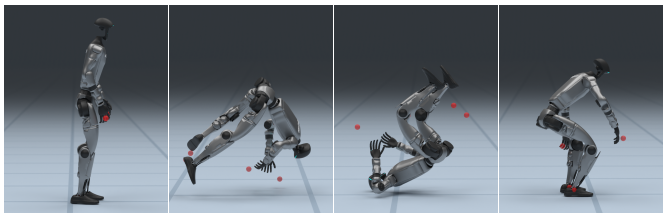
Walking backwards (steering): An Xbox controller specifies heading and travel velocity.



Teleoperation: VR headset provides teleoperation goals.



Style control (simulation): An $[x, y]$ goal and five static crawling poses guide goal reaching.



Tracking (simulation): Dense targets specify a front roll.

Fig. 5: **Multiple control modes through one G1 policy.** Each mode specifies different goals for the same frozen controller.

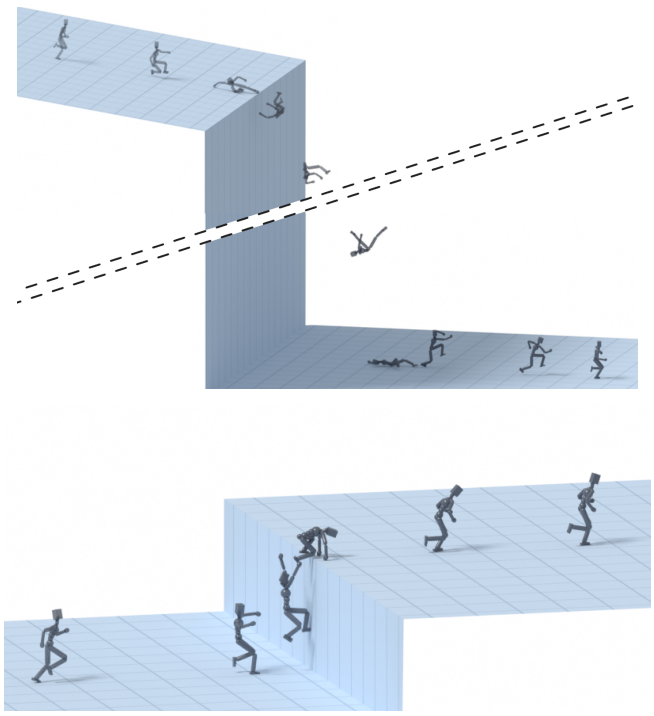


Fig. 6: **SOMA:** On a simulated SOMA character, with the same recipe, METATRON runs, slides off a ledge, recovers from the fall, then runs and climbs up a wall.

the robot to accommodate physical guidance during human–robot interactions. By training the controller with synthetic recovery data, the controller is able to get-up from diverse fallen configurations on the ground, including awkward configurations that are not recorded in the original mocap data. Hardware experiments show recovery from lying on the back, lying on the front, sitting, and other fallen poses (Figure 7), extending the controller's repertoire beyond the original demonstrations. Recovery is triggered automatically when torso height falls below a safety threshold (Figure 4).

D. Simulation Capabilities

To avoid potential damage to the physical G1 and its three-finger hands, we demonstrate highly dynamic skills in MuJoCo simulation. Dense body targets produce behaviors such as a front roll (Figure 5). Sparse style frames bias the motion toward example poses: a torso $[x, y]$ target and arrival time specify where and when to reach, while five unordered crawling poses guide the style without imposing their original timing.

In addition to the simulated G1 model, we also train a policy for the SOMA character using a METATRON recipe similar to G1's. In Figure 6, a stateful skill supplies velocity, heading, and terrain-relative pelvis-height goals for locomotion. A low pelvis target and temporary right-foot target initiate sliding. After falling off the ledge, only velocity

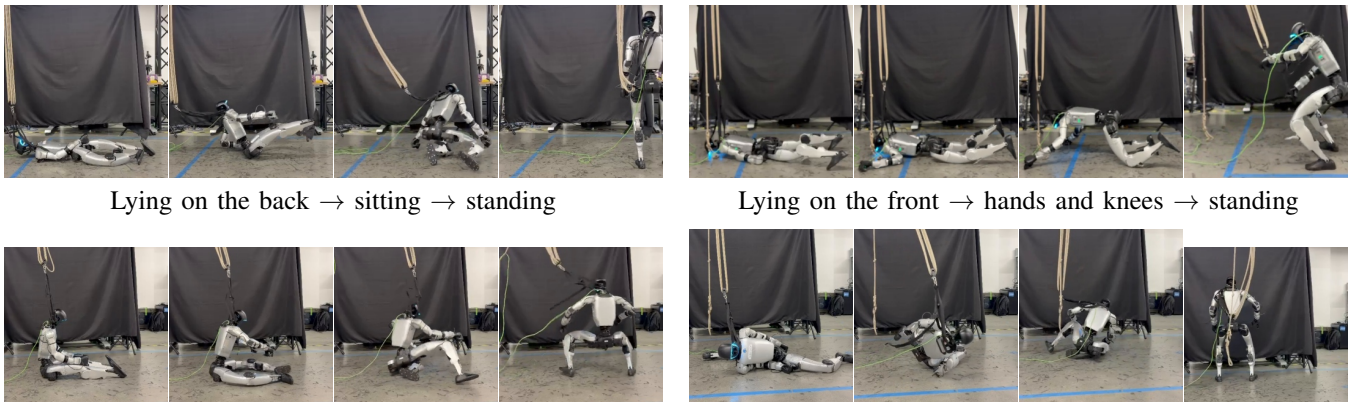


Fig. 7: **Recovery:** METATRON recovers from a range of initial configurations.

and heading remain through the fall and recovery. Climbing uses heading and a pelvis target above and beyond the ledge, leaving hand and foot placement unconstrained. The frozen policy executes the sequence without reference trajectories or full-body landing and recovery poses.

E. Comparison with Reference-Tracking Pipelines

A common alternative to a native whole-body controller is a reference-tracking pipeline: a learned motion generator produces a reference trajectory, and a dense tracker follows it. We compare METATRON against two such pipelines on the same G1 embodiment. The first uses ARDY [32], a state-of-the-art realtime autoregressive diffusion model that generates whole-body reference poses including hand keyframes. The second uses a fast GRU locomotion generator that we train to produce walking references; for reaching, we additionally overlay analytic arm IK on its output. Both pipelines use the same SONIC [12] dense tracker. For this comparison, METATRON is trained without synthetic recovery data (section V).

Whole-body reaching: (Table I). We evaluate wrist target reaching in Mujoco using 54 targets spanning three distances (0.35–1.8 m, so the G1 will need to take steps), three heights (0.55–1.35 m), and both hands; a target counts as reached if the wrist is held within 10 cm while the robot remains upright. All methods that reach a target achieve comparable arm precision; what separates them is the reached rate. ARDY conditions on the hand target through classifier-free guidance rather than hard inpainting, so the held wrist oscillates across successive re-plans. The gap is largest at targets requiring whole-body adjustment: high reaches that demand counterbalance and low reaches that require a hip fold, where a generator that plans at a fixed standing height cannot adapt.

Push robustness during steering: (Table II). During a 1 m/s forward walk, a force pulse is applied at the torso for 100 ms, with directions spanning eight headings. Both reference-tracking pipelines have zero success above 600 N, whereas METATRON succeeds in 75% of trials at 800 and 1000 N. This comparison measures the complete control pipelines; it does not isolate which training or interface choice causes the difference. The non-monotonic rates at 600–1000 N should not be interpreted as improved robustness to stronger pushes.

TABLE I: Whole-body wrist reaching on a 54-target grid. METATRON is trained without synthetic recovery data for this comparison. A target is reached if the wrist is held within 10 cm of it while the robot remains upright. Error is measured **only on reached targets**.

Method	Reached (%)	Error (cm)	Upright (%)
METATRON	**83**	5.9	**100**
ARDY + SONIC	39	6.4	82
GRU + IK + SONIC	28	5.6	81
GRU + SONIC	0	—	87

TABLE II: Push robustness during steering (100 ms force pulse at torso, 8 directions). Success: upright after 8 s. METATRON trained **without** recovery data. Results: success (%).

	Perturbation force (N)				
Method	200	400	600	800	1000
METATRON	**100**	**100**	**62**	**75**	**75**
ARDY+SONIC	**100**	**100**	17	0	0
GRU+SONIC	**100**	75	25	0	0

F. Ablation Study

We test which training components support robustness by requesting planar waypoint goals directly, without skill-generated foot targets or height constraints, under repeated horizontal torso pushes (Table III). All models use the same training budget and matched parameter counts; the budget is given in the supplementary material. We evaluate 500 episodes per method and force, measuring upright arrival and reporting 95% Wilson confidence intervals.

We compare METATRON with variants without recovery data (*Mocap only*), without latent auxiliary losses (*No aux.*), and with only global spatio-temporal prompts (*Global only*). We also construct MaskedMimic^{+} [8] and HOVER^{+} [17] by adapting the original methods to our expert, training pipeline, motion-capture corpus, and compute budget, with substantially larger networks than their original implementations.

TABLE III: Push robustness during XY waypoint reaching. Horizontal torso pushes last 100 ms, recurring every 1–2 s during the first 5 s until arrival. Success: reaching within 0.5 m of the goal within 10 s, with relative torso height ≥ 0.45 m and tilt $\leq 60°$ at arrival. Entries report success (%) with upper/lower 95% Wilson CI deviations in percentage points over 500 episodes. Bold marks CI overlap with the highest rate at each force (unrounded bounds).

	Perturbation force (N)				
Method	0	200	400	600	800
METATRON	$\mathbf{97.4^{+1.1}_{-1.8}}$	$\mathbf{94.8^{+1.6}_{-2.3}}$	$\mathbf{33.0^{+4.2}_{-4.0}}$	$\mathbf{7.4^{+2.6}_{-2.0}}$	$\mathbf{6.2^{+2.5}_{-1.8}}$
Global only	$87.6^{+2.6}_{-3.2}$	$87.8^{+2.6}_{-3.2}$	$\mathbf{38.2^{+4.3}_{-4.2}}$	$\mathbf{9.6^{+2.9}_{-2.3}}$	$\mathbf{6.2^{+2.5}_{-1.8}}$
Mocap only	$92.0^{+2.1}_{-2.7}$	$84.8^{+2.9}_{-3.4}$	$\mathbf{31.8^{+4.2}_{-3.9}}$	$\mathbf{6.4^{+2.5}_{-1.8}}$	$\mathbf{5.0^{+2.3}_{-1.6}}$
No aux.	$81.6^{+3.2}_{-3.6}$	$78.8^{+3.4}_{-3.8}$	$29.2^{+4.1}_{-3.8}$	$\mathbf{5.4^{+2.3}_{-1.7}}$	$\mathbf{4.6^{+2.2}_{-1.5}}$
MaskedMimic$^+$	$\mathbf{98.2^{+0.9}_{-1.6}}$	$90.0^{+2.3}_{-2.9}$	$27.8^{+4.1}_{-3.7}$	$4.2^{+2.1}_{-1.4}$	$\mathbf{4.6^{+2.2}_{-1.5}}$
HOVER$^+$	$66.8^{+4.0}_{-4.2}$	$71.0^{+3.8}_{-4.1}$	$\mathbf{30.0^{+4.2}_{-3.9}}$	$3.0^{+1.9}_{-1.2}$	$1.2^{+1.4}_{-0.6}$

These adaptations serve as combined ablations of METATRON. MaskedMimic$^+$ combines all three ablations above and replaces our deterministic and sphere-projected stochastic streams with a single unbounded Gaussian latent. HOVER$^+$ further removes the conditional VAE and replaces the transformer with a deterministic MLP policy.

The full model achieves higher observed success than both the recovery-data and auxiliary-loss ablations at every tested force, with the largest absolute gains at 0–200 N. It also exceeds *Global only* at these forces, while *Global only* has higher success at 400–600 N and ties at 800 N; their confidence intervals overlap at all three higher forces. Broader prompt support therefore remains competitive in this evaluation, although interval overlap does not establish equivalence. Compared with the adapted MaskedMimic$^+$ and HOVER$^+$ baselines, METATRON achieves higher observed success at every nonzero force, while MaskedMimic$^+$ leads slightly without pushes. However, several intervals overlap, and all methods fall below 10% success at 600–800 N, highlighting remaining limitations under strong perturbations.

VII. LIMITATIONS

Motion coverage and conditioning: METATRON's repertoire depends on training coverage, and underspecified goals or incidental style cues can produce unintended motions. For example, abruptly commanding $2\,\mathrm{m/s}$ from rest often triggers a front roll. Additional motion capture and targeted synthetic trajectories could broaden the repertoire, while refining skill-generated goals can improve motion selection without retraining. We smooth velocity goals with an exponential moving average to increase the requested speed gradually.

Externally supplied goals: The task system remains responsible for goal selection and long-horizon planning. We place the G1 near a chair or sofa and supply a nearby-object cue; autonomous approach followed by sitting is not demonstrated. Localization and perception could enable automatic goal generation, while LLM-generated programs could sequence goals for longer-horizon tasks [33].

Hardware evaluation: Our hardware study demonstrates diverse behaviors with a single frozen controller, including recovery from tested configurations and qualitative gait improvements. Trials across initial and operating conditions would quantify recovery reliability and gait stability and assess contact safety, actuation limits, and failure modes.

VIII. CONCLUSION

METATRON converts dense motion supervision into a generative whole-body controller for partial goals. Synthetic recovery data expand its repertoire, and body-level prompts support runtime control in simulation and on G1. Within the learned repertoire, new tasks require goal-generating programs, not policy retraining. Skills specify goals, timing, and transitions; the frozen policy coordinates the body.

REFERENCES

[1] A. A. Orlov, *The Enoch-Metatron Tradition*, ser. Texts and Studies in Ancient Judaism. Tübingen: Mohr Siebeck, 2005, vol. 107.

[2] G. Scholem, *Major Trends in Jewish Mysticism*. Jerusalem: Schocken Publishing House, 1941.

[3] X. Gu, Y.-J. Wang, and J. Chen, "Humanoid-gym: Reinforcement learning for humanoid robot with zero-shot sim2real transfer," *arXiv:2404.05695*, 2024.

[4] Z. Fu, Q. Zhao, Q. Wu, G. Wetzstein, and C. Finn, "Humanplus: Humanoid shadowing and imitation from humans," in *8th Annual Conference on Robot Learning*, 2024. [Online]. Available: https://openreview.net/forum?id=WnSl42M9Z4

[5] X. He, R. Dong, Z. Chen, and S. Gupta, "Learning getting-up policies for real-world humanoid robots," in *Proceedings of Robotics: Science and Systems*, 2025.

[6] G. B. Margolis, M. Wang, N. Fey, and P. Agrawal, "Softmimic: Learning compliant whole-body control from examples," *arXiv:2510.17792*, 2025.

[7] Q. Liao, T. E. Truong, X. Huang, Y. Gao, G. Tevet, K. Sreenath, and C. K. Liu, "BeyondMimic: From motion tracking to versatile humanoid control via guided diffusion," *Science Robotics*, vol. 11, no. 117, p. eadx8924, 2026.

[8] C. Tessler, Y. Guo, O. Nabati, G. Chechik, and X. B. Peng, "MaskedMimic: Unified physics-based character control through masked motion inpainting," *ACM Transactions on Graphics*, vol. 43, no. 6, 2024.

[9] X. B. Peng, P. Abbeel, S. Levine, and M. van de Panne, "DeepMimic: Example-guided deep reinforcement learning of physics-based character skills," *ACM Transactions on Graphics*, vol. 37, no. 4, 2018.

[10] Z. Luo, J. Cao, A. Winkler, K. Kitani, and W. Xu, "Perpetual humanoid control for real-time simulated avatars," in *IEEE/CVF International Conference on Computer Vision*, 2023.

[11] Z. Chen, M. Ji, X. Cheng, X. Peng, X. B. Peng, and X. Wang, "GMT: General motion tracking for humanoid whole-body control," *arXiv:2506.14770*, 2025.

[12] Z. Luo, Y. Yuan, T. Wang, C. Li, S. Chen, F. Castañeda, Z.-A. Cao, J. Li *et al.*, "SONIC: Supersizing motion tracking for natural humanoid whole-body control," *arXiv:2511.07820*, 2025.

[13] X. B. Peng, Y. Guo, L. Halper, S. Levine, and S. Fidler, "ASE: Large-scale reusable adversarial skill embeddings for physically simulated characters," *ACM Transactions on Graphics*, vol. 41, no. 4, 2022.

[14] C. Tessler, Y. Kasten, Y. Guo, S. Mannor, G. Chechik, and X. B. Peng, "CALM: Conditional adversarial latent models for directable virtual characters," in *ACM SIGGRAPH Conference Proceedings*, 2023.

[15] Z. Luo, J. Cao, J. Merel, A. Winkler, J. Huang, K. Kitani, and W. Xu, "Universal humanoid motion representations for physics-based control," in *International Conference on Learning Representations*, 2024.

[16] Y. Shi, Y. Jiang, C. Tessler, and X. B. Peng, "GPC: Large-scale generative pretraining for transferable motor control," in *ACM SIGGRAPH Conference Proceedings*, 2026.

[17] T. He, W. Xiao, T. Lin, Z. Luo, Z. Xu, Z. Jiang, J. Kautz, C. Liu, G. Shi, X. Wang *et al.*, "Hover: Versatile neural whole-body controller for humanoid robots," in *2025 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2025, pp. 9989–9996.

[18] S. Xu, S. Schulter, M. Ziyadi, X. He, X. Fei, Y.-X. Wang, and L. Gui, "InterPrior: Scaling generative control for physics-based human-object interactions," *arXiv:2602.06035*, 2026.

[19] D. Li, Q. Wu, X. Chen, L. Li, Y. Lin, S. Wu, G. Zhang, M. Zhou, D. Xiang, Q. Zhang *et al.*, "Vaic: Vision-guided humanoid agile object interaction control via decoupled commands," *arXiv preprint arXiv:2606.09286*, 2026.

[20] R. Yu, X. Lin, J. Ma, Y. Wang, K. Luo, J. Ji, H. Wang, W. Wang, R. Zhang, P. Tan *et al.*, "Omnicontact: Chaining meta-skills via contact flow for generalizable humanoid loco-manipulation," *arXiv preprint arXiv:2606.26201*, 2026.

[21] W. Zeng, K. Yin, X. Niu, S. Lu, W. Zhong, J. Chen, F. Jia, X. Chen, Z. Wang, F. Xu *et al.*, "Scaling behavior foundation model for humanoid robots," *arXiv preprint arXiv:2607.15163*, 2026.

[22] T. He, Z. Luo, X. He, W. Xiao, C. Zhang, W. Zhang, K. Kitani, C. Liu, and G. Shi, "OmniH2O: Universal and dexterous human-to-humanoid whole-body teleoperation and learning," *arXiv:2406.08858*, 2024.

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

[24] M. Schwarzer, A. Anand, R. Goel, R. D. Hjelm, A. Courville, and P. Bachman, "Data-efficient reinforcement learning with self-predictive representations," in *International Conference on Learning Representations*, 2021.

[25] N. Hansen, H. Su, and X. Wang, "TD-MPC2: Scalable, robust world models for continuous control," in *International Conference on Learning Representations*, 2024.

[26] H. Fu, C. Li, X. Liu, J. Gao, A. Celikyilmaz, and L. Carin, "Cyclical annealing schedule: A simple approach to mitigating KL vanishing," in *Proceedings of NAACL-HLT*, 2019, pp. 240–250.

[27] C. M. Kim*, B. Yi*, H. Choi, Y. Ma, K. Goldberg, and A. Kanazawa, "Pyroki: A modular toolkit for robot kinematic optimization," in *2025 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2025. [Online]. Available: https://arxiv.org/abs/2505.03728

[28] Bones Studio, "BONES-SEED: Skeletal everyday embodiment dataset," Hugging Face, 2026, accessed: Sep. 13, 2026. [Online]. Available: https://huggingface.co/datasets/bones-studio/seed

[29] M. Hassan, D. Ceylan, R. Villegas, J. Saito, J. Yang, Y. Zhou, and M. J. Black, "Stochastic scene-aware motion prediction," in *IEEE/CVF International Conference on Computer Vision*, 2021.

[30] J. Wang, Y. Jiang, H. Zhang, C. Tessler, D. Rempe, J. Hodgins, and X. B. Peng, "Hil: Hybrid imitation learning for dynamic athletic control," *ACM Transactions on Graphics*, 2026.

[31] C. Tessler, Y. Jiang, X. B. Peng, E. Coumans, Y. Shi, H. Zhang, D. Rempe, G. Chechik, and S. Fidler, "ProtoMotions3: An open-source framework for humanoid simulation and control," GitHub repository, 2025. [Online]. Available: https://github.com/NVlabs/ProtoMotions

[32] K. Zhao, M. Petrovich, H. Zhang, T. Wang, S. Tang, and D. Rempe, "Autoregressive diffusion with hybrid representation for interactive human motion generation," *ACM Transactions on Graphics (TOG)*, vol. 45, no. 4, pp. 1–14, 2026.

[33] J. Liang, W. Huang, F. Xia, P. Xu, K. Hausman, B. Ichter, P. Florence, and A. Zeng, "Code as policies: Language model programs for embodied control," in *2023 IEEE International conference on robotics and automation (ICRA)*. IEEE, 2023, pp. 9493–9500.

[34] M. Mittal, P. Roth, J. Tigue, A. Richard, O. Zhang, P. Du, A. Serrano-Muñoz, X. Yao, R. Zurbrügg, N. Rudin *et al.*, "Isaac Lab: A GPU-accelerated simulation framework for multi-modal robot learning," *arXiv:2511.04831*, 2025.

APPENDIX I
IMPLEMENTATION DETAILS

This appendix separates three stages: PPO trains motion-tracking experts; supervised distillation trains the prompt-conditioned student; runtime skills construct prompts for the frozen student. We give their settings below, followed by observation construction and deployment details. Notation follows section III.

A. Training Setup

We train in ProtoMotions [31] using Isaac Lab [34]. G1 distills four frozen experts into one student: BONES [28], AMASS+LAFAN, recovery, and SAMP. These groups use 12, 4, 2, and 2 GPUs, respectively. Each GPU simulates 2048 environments, collects 32 control steps per rollout, and optimizes minibatches of 4096 samples for two passes. Averaging equal-sized batches across GPUs gives the groups relative training weights $(0.6, 0.2, 0.1, 0.1)$. All GPUs share the student parameters; each group uses its own expert, motions, and scenes.

G1 simulates at 200 Hz with four physics steps per action (50 Hz control); SOMA uses 120 Hz physics and 30 Hz control. Runtime skills use the trained student without further weight updates.

a) Compute budgets: Student training uses 20 GPUs. The ablations use eight A100 GPUs for $5{,}000$ training iterations with matched parameter counts. These budgets exclude expert training.

b) Training data: The dataset includes BONES motion capture and synthetic recovery motions; G1 additionally uses AMASS+LAFAN and SAMP. The SOMA scene dataset contains wall-climbing and ground-sliding motions with collision geometry.

B. Dense Reference Trackers

a) Data and observations: The SOMA teacher learns from flat-ground motion capture, recovery, climbing, and sliding. Walls and ramps are included in the collision geometry, and a local 16×16 height map provides terrain context. G1's BONES, AMASS+LAFAN, and recovery groups use flat ground; SAMP supplies scene interactions. Trackers observe privileged body positions, orientations, and velocities, together with complete reference poses sampled at future times.

b) G1 BONES expert: The G1 BONES expert uses finite scalar quantization (FSQ). An encoder with hidden widths $(2048, 1024, 512, 512)$ maps future poses of the torso, hands, and feet, together with joint targets, to 64 latent coordinates with 33 quantization levels each. Its actor decoder predicts the mean of a Gaussian action distribution; a separate critic predicts state value. Both have two hidden layers at each width, in order: 4096, 2048, 1024, and 512, with SiLU activations. These eight-layer networks belong to the expert; the deployed student's decoder has three 1024-unit layers. The expert outputs 29 joint commands.

The G1 BONES expert learns one log-standard-deviation vector per motion, shared across states of that motion. The bounds are $\log(0.001)$ and $\log(0.5)$, its learning rate is 5×10^{-4}, and the entropy coefficient is 0.01. This table controls PPO exploration; the distilled student does not inherit it.

c) SOMA expert: The SOMA teacher has eight 1024-unit actor hidden layers, six 1024-unit critic hidden layers, and 66 action outputs. It uses fixed log-standard deviation -2.9. Its smoothness penalty acts on consecutive joint-position commands: $-0.1\|q_t^{\mathrm{des}} - q_{t-1}^{\mathrm{des}}\|_2$.

d) Expert optimization: Both experts use PPO with discount 0.99, generalized advantage estimation (GAE) parameter 0.95, and ratio clipping 0.2. Actor and critic Adam learning rates are 2×10^{-5} and 10^{-4}, with momentum coefficients $(0.95, 0.99)$. Both collect 32-step rollouts. Table IV lists the differing settings. G1 also uses clipped value loss.

TABLE IV: Expert PPO settings, distinct from student distillation.

Setting	G1 BONES	SOMA
Environments per GPU	8192	4096
Minibatch size per GPU	16384	8192
Optimization passes per rollout	2	1
Gradient-norm limit	1	50
Entropy coefficient	0.01	0.005
Reward normalization	No	Yes

G1 rewards are multiplied by 0.8 before PPO optimization.

e) Speed-dependent precision: The SOMA teacher uses speed-gated precision rewards; the G1 BONES expert in table V does not. For SOMA, let $\bar{v}_b$ be the average over selected future samples of reference displacement magnitude divided by time offset, and let $\bar{\omega}_b$ use relative rotation angle instead of displacement. The per-body gates are

$$\eta_b^p = \exp[-(\bar{v}_b/\rho_p)^2], \qquad \eta_b^R = \exp[-(\bar{\omega}_b/\rho_R)^2], \quad (7)$$

with $\rho_p = 0.5\,\mathrm{m/s}$ and $\rho_R = 1\,\mathrm{rad/s}$. For a selected body set B, the reward is

$$r_B^k = \exp\left[-\frac{\lambda_k}{|B|}\sum_{b\in B}\eta_b^k e_b^k\right]. \quad (8)$$

Position error is $e_b^p = \|p_b - p_b^*\|^2/3$; rotation error is the squared relative rotation angle. The mean is taken *inside* the exponential. Position and rotation sensitivities are 5000 and 200. Hand terms each have weight 1; foot, anchor, and head terms each have weight 0.5. At high reference speeds the gates approach zero, flattening the reward rather than increasing its weight. The teacher also uses ordinary position, rotation, linear-velocity, angular-velocity, and root-height terms with $(w, \lambda) = (0.5, 100), (0.3, 5), (0.1, 0.5), (0.1, 0.1), (0.2, 100)$, processed-command smoothness, and a power penalty. These tracker rewards are distinct from the student's supervised losses.

C. Masked On-Policy Distillation

a) Prior encoder: Separate encoders map the prompt streams and sensor history into 1024-dimensional tokens. The transformer has six layers, four attention heads, 4096-dimensional feed-forward blocks, and zero dropout. Separate

TABLE V: G1 BONES expert rewards. Tracking terms are $w\exp(-e/\sigma^2)$. Position and velocity errors sum squared coordinate errors, then average over the selected bodies; rotation error is the mean squared relative rotation angle.

Term	w	σ
Torso, hands, feet: global position	1	0.3
Torso, hands, feet: global rotation	1	0.4
Torso and hands: global position	2	0.1
Torso and hands: global rotation	0.4	0.1
Anchor-relative body position	1	0.3
Anchor-relative body rotation	1	0.4
Body linear velocity	1	1.0
Body angular velocity	1	3.14
Soft joint-limit violation	-10	—
Raw action squared difference	-0.1	—
Contact mismatch	-0.1	—
Self-contact count	-0.1	—

$(512, 256, 128)$ MLP heads predict the 64-dimensional condition d_t, prior mean, and prior log-variance. Masks are explicit inputs; unavailable tokens are excluded by attention and invisible style frames by masked pooling. A zero-valued visible target remains different from an absent target.

b) Fusion and action decoder: The fusion MLP maps $[d_t, z_t]$ to the 64-dimensional representation h_t using hidden widths $(512, 256, 128)$. The action decoder combines h_t with normalized current observations and the previous action, then applies three 1024-unit hidden layers. G1 uses body poses reconstructed by forward kinematics (FK), joint measurements, and root-state measurements, and outputs 29 commands. SOMA uses simulator-provided body states and a height map, and outputs 66 commands. G1 uses SiLU and SOMA uses ReLU.

c) Training-only networks: The privileged posterior uses five 1024-unit hidden layers and a 512-dimensional output, followed by separate mean-correction and log-variance heads with hidden widths $(512, 256, 128)$. It receives full reference features and prompt features. The dynamics predictor maps h_t^q to the next condition d_{t+1} through hidden widths $(512, 256, 128)$. SOMA supplies height-map and root-height information to both the prior and posterior.

The prompt encoders use three 1024-unit hidden layers; style frames share an encoder and are pooled over the visible frames. The posterior's canonicalized- and world-target encoders use $(512, 512)$ hidden layers with 256-dimensional outputs. G1 additionally encodes scene-box corners with a 1024-unit hidden layer and 1024-dimensional output. Normalized observations are clipped to $[-5, 5]$; masks are not normalized. Inference uses the normalization statistics learned during training.

d) Sampling and rollouts: At reset, each environment draws one Gaussian noise vector and retains it for the episode. The prior and posterior share this vector, but their means and variances change with the observations and prompt. Their latent samples therefore remain context-dependent. Each sample is projected onto the unit sphere before fusion with d_t.

At the same reset, control is assigned to the posterior-guided student with probability 0.9 and to the frozen tracker otherwise. That controller acts for the whole episode. The tracker labels every visited state under either controller, providing supervision on both student- and expert-induced states. At deployment, actions use the prior path.

e) Losses: In eq. (5), $\lambda_{\mathrm{dyn}} = \lambda_{\mathrm{tc}} = \lambda_\mu = 10^{-3}$ and $\lambda_{\mathrm{act}} = 5 \times 10^{-2}$. Both students use Adam with learning rate 2×10^{-5}, momentum coefficients $(0.9, 0.999)$, $\epsilon = 10^{-8}$, no weight decay, and gradient-norm clipping at 50. The imitation loss averages squared differences between student and expert actions over action coordinates, before conversion to joint-position commands. Student training uses supervised losses rather than PPO rewards.

f) Gaussian KL and spherical decoding: Write $\ell_j^p = \log \sigma_j^2$ and $\ell_j^q = \log {\sigma'_j}^2$. The implemented per-sample KL is

$$D_{\mathrm{KL}}(q\|p) = \frac{1}{2}\sum_{j=1}^{64}\left[\ell_j^p - \ell_j^q + \frac{\exp(\ell_j^q) + (\Delta\mu_j)^2}{\exp(\ell_j^p)} - 1\right]. \tag{9}$$

The KL is summed over latent coordinates and averaged over samples. The posterior predicts its own log-variance and a residual mean relative to the prior. Sampling uses $\tilde{z}^q = \mu + \Delta\mu + \exp(\ell^q/2) \odot \epsilon$ for the posterior and $\tilde{z}^p = \mu + \exp(\ell^p/2) \odot \epsilon$ for the prior. KL and mean-norm regularization act on these Gaussian parameters; the sampled latent is normalized to unit length before decoding.

g) KL schedule: G1 uses cyclical annealing. With epoch e, define

$$\hat{e} = \begin{cases} e, & e \le 2000, \\ 500 + ((e - 500) \bmod 1501), & e > 2000, \end{cases}$$
$$\beta_e = 10^{-4} + (10^{-2} - 10^{-4})\,\mathrm{clip}\left(\frac{\hat{e} - 500}{1500}, 0, 1\right). \tag{10}$$

The weight rises from 10^{-4} to 10^{-2} over epochs 500–2000, then repeats the ramp. SOMA uses the same initial ramp and holds 10^{-2} thereafter. An epoch denotes one rollout followed by optimization.

h) Temporal targets: Temporal losses use consecutive steps within the same episode and exclude pairs that cross a reset. The dynamics loss predicts the next condition d_{t+1} from h_t^q; consistency and action smoothing compare the current representation or posterior action with its previous value. Targets are stored during rollout and treated as constants during optimization (stop-gradient). Pairs are formed before minibatch shuffling, and errors are averaged over coordinates and valid pairs. Action smoothing compares student predictions even in expert-controlled episodes. These losses use per-step differences without division by Δt.

i) SOMA latent diversity: SOMA additionally encourages different latents to produce different actions; G1 omits this loss. Holding the observation and condition fixed, draw two independent unit-sphere latents z_1, z_2 and decode actions

a_1, a_2. With $c(a) = \mathrm{clip}(a, -1, 1)$ and A action coordinates,

$$\mathcal{L}_{\mathrm{div}} = 10^{-4}\,\mathbb{E}\left[\left(1 - \frac{A^{-1}\|c(a_1) - c(a_2)\|_2^2}{\frac{1}{2}(1 - z_1^\top z_2) + 10^{-5}}\right)^2\right]. \quad (11)$$

The square is applied per sample before averaging. Both decoded actions receive gradients; the sampled latents and conditioning inputs do not. This term supplements, rather than replaces, the temporal losses.

j) Training iteration: At reset, choose the motion and prompt masks, sample latent noise for the episode, and choose student or expert control. At each step, compute both the expert label and the student's posterior action, execute the chosen controller, and record the transition. After a 32-step rollout, form within-episode temporal pairs, shuffle transitions into minibatches, and perform two passes of supervised updates. Student gradients are averaged across GPUs; experts remain frozen. Inference uses only the prior, fusion network, and action decoder.

D. Prompt and Style-Cue Sampling

Training varies both the information in a prompt and its time horizon. For each body-target stream and episode, a spacing $\Delta \in [0.1, 2.0]$ s places five future slots at $t + k\Delta$, $k \in \{1, \ldots, 5\}$. Dense, intermediate, and sparse schedules are chosen with probabilities $(0.2, 0.6, 0.2)$, each using its own spacing subrange. Dense schedules expose every slot, sparse schedules expose one, and intermediate schedules expose each slot with probability 0.8. Masks also select body identities and target properties. Position targets support separate planar (x, y) and vertical z constraints. Anchor orientation can be specified as heading alone or as a full three-dimensional rotation; position and orientation goals can be selected independently.

The world-target, canonicalized-target, and planar-velocity streams are each active with probability 0.5. The style cue samples up to five frames from a two-second sliding window. Its visible-frame-count distribution, for zero through five frames, is $(0.50, 0.15, 0.15, 0.10, 0.07, 0.03)$, so it too is absent in half the episodes. The four presence decisions are independent, making all sixteen combinations equally likely, including no motion goal.

Scene context is separate from these four masks. G1 accepts one object represented by up to two boxes. Valid boxes remain visible in the SAMP group; unused boxes and all box tokens in scene-free groups are masked out. This fixed input shape lets the same student operate with or without furniture. SOMA instead receives scene geometry through its local height map.

Each style frame is canonicalized like a local target pose. Heading-aligned linear and angular body velocities are included with probability 0.5. The frames carry neither arrival times nor global placement. A shared frame encoder followed by masked mean pooling makes the style representation invariant to their order. This distinguishes an example-based distribution cue from timed body targets.

E. Observation Construction

a) SOMA: The simulator provides body positions, orientations, and velocities, root height, and a local 16×16 height map. These inputs support both current-state observations and comparisons with world-frame goals.

b) G1 current pose: Let A denote the torso anchor, R_t^A its orientation reconstructed from IMU and joint measurements, and H_t the yaw-only rotation extracted from R_t^A. The G1 sensor suite places the IMU on the pelvis; torso orientation is derived through FK rather than treated as an independent torso-mounted measurement. FK applied to measured joint angles q_t gives each body's torso-relative position $p_b(q_t)$ and orientation $R_b(q_t)$. Rotating into the heading frame gives $\bar{p}_t^b = H_t^\top R_t^A p_b(q_t)$. We then set the lowest body point to zero height:

$$\begin{aligned} \gamma_t &= \min_{j \in \mathcal{J}} \bar{p}_{t,z}^j, \\ \hat{p}_t^b &= \bar{p}_t^b - \gamma_t e_z, \\ \hat{R}_t^b &= H_t^\top R_t^A R_b(q_t), \qquad b \in \mathcal{B}, \end{aligned} \quad (12)$$

where $\mathcal{J}$ contains all bodies, $\mathcal{B}$ contains the torso, hands, and feet used for prompts, and e_z is the vertical unit vector. With a 6D rotation representation, this gives nine pose features per prompted body. This normalization defines a local pose convention; it does not estimate ground height.

The history encoder receives five steps of joint position, joint velocity, root orientation, angular velocity in the root frame, and reconstructed torso orientation, together with the time elapsed since each measurement. The current-state encoder and action decoder also receive the previous action. Sensor noise is applied before FK: independent uniform perturbations have half-widths 0.01 rad for joint position, 0.5 rad/s for joint velocity, 0.05 per IMU quaternion component, and 0.2 rad/s for gyroscope readings. The perturbed quaternion is renormalized. Localization, base velocity, and ground clearance receive no added noise during training.

c) Physical randomization: All four G1 groups add action offsets in $[-0.025, 0.025]$ and use 64 sampled material settings with static friction in $[0.3, 1.6]$, dynamic friction in $[0.3, 1.2]$, and restitution in $[0, 0.5]$. Torso center-of-mass offsets span ± 0.025 m in x and ± 0.05 m in y, z. Hand masses are sampled log-uniformly between 0.001 and 1.0 kg. Pushes are implemented as velocity perturbations every 1–3 s, with per-axis linear bounds $(0.5, 0.5, 0.2)$ m/s and angular bounds $(0.52, 0.52, 0.78)$ rad/s. These training perturbations differ from the force pulses used in evaluation. G1 training adds no sensor or action latency. SOMA disables domain randomization.

d) Canonicalized targets: Each target pose uses its own torso position and heading at target time τ. Let $(p_\tau^{*,b}, R_\tau^{*,b})$ be the reference body poses and H_τ^* the target torso heading. With $\gamma_\tau = \min_{j \in \mathcal{J}} p_{\tau,z}^{*,j}$, the target features are

$$\begin{aligned} p_{\tau,xy}^{\ell,b} &= \left[H_\tau^{*\top}(p_\tau^{*,b} - p_\tau^{*,A})\right]_{xy}, \\ p_{\tau,z}^{\ell,b} &= p_{\tau,z}^{*,b} - \gamma_\tau, \\ R_\tau^{\ell,b} &= H_\tau^{*\top} R_\tau^{*,b}. \end{aligned} \quad (13)$$

These targets and the current FK view are encoded separately for the network to compare. Training canonicalizes the complete target pose before masking bodies and properties, so the vertical origin remains well defined. A runtime skill uses the same convention when constructing a target. Canonicalized target values depend on the requested pose, not the current robot state.

e) World-frame targets: A localization estimate $\tilde{p}_t^A$ places the FK pose in the world: $\tilde{p}_t^b = \tilde{p}_t^A + R_t^A p_b(q_t)$ and $\tilde{R}_t^b = R_t^A R_b(q_t)$. Target features then encode position differences in the current heading frame and relative rotations:

$$\begin{aligned} \delta p_{t,\tau}^b &= H_t^\top (p_\tau^{*,b} - \tilde{p}_t^b), \\ \delta R_{t,\tau}^b &= (\tilde{R}_t^b)^\top R_\tau^{*,b}. \end{aligned} \tag{14}$$

Rotations use the 6D representation, and masks select the requested properties. World-position deltas require the target and robot poses to share an origin: planar targets need x–y localization, and absolute-height targets also need vertical registration. Subtracting the same floor offset from both poses leaves their difference unchanged. Canonicalized targets omit global placement and do not require this registration.

F. Recovery-Data Generation

We extend HIL [30] with MaskedMimic-style spatio-temporal conditioning [8], masking body constraints and future times. The trained policy generates five-second recovery clips offline from randomized ground poses, using 1024 parallel environments.

a) Initial states: We sample a root orientation and joint targets, simulate 150 physics steps, and zero all velocities. The sampled poses thus settle under contact dynamics before recording begins.

b) Targets and acceptance: Each rollout receives a sparse timed target from a source motion, with at least one position-constrained body. During collection, termination conditions are disabled. A clip is accepted if it contains no reset and its final mean Euclidean error over position-constrained bodies is below $0.25\,\mathrm{m}$. This filter checks endpoint tracking, not sustained balance, contact quality, or hardware safety.

c) Storage and use: Accepted clips store body and joint poses, velocities, and contacts. We subtract only the initial root x–y translation, preserving height and yaw. The clips enter the motion library in the same format as captured motions and supervise the same tracking and distillation stages. The generator is used only for data collection; it is not part of the deployed controller.

G. Harness Skills

Each skill is a stateful function or finite-state machine that constructs prompts from task input. The harness runs these skills and manages transitions between them. Table VI summarizes the implemented skills; none requires training a new motor policy.

a) Steering trajectories: For the rule in section IV, the fitted coefficients are $A(v) = 0.75(0.13 + 0.065v)$, $f(v) = \min(2, 0.83 + 0.93v)$, $W = 0.12\,\mathrm{m}$, and $H = 0.08\,\mathrm{m}$, with speed in m/s and cadence in Hz. Here W controls lateral widening, not the nominal stance width. At each control step, phase advances by $2\pi f(v)\Delta t$. The skill evaluates foot targets at arrival times $\tau \in \{0.1, 0.2, 0.3, 0.4, 0.5\}\,\mathrm{s}$ and transforms them from the planned torso frame into world coordinates using the torso position and heading targets at each arrival time.

Deployment additionally enforces a nominal stance half-width of at least $0.07\,\mathrm{m}$. At or below $0.05\,\mathrm{m/s}$, stationary-support targets replace the periodic gait. The skill specifies body-position goals; the learned policy determines joint commands and physical contacts.

b) Recovery state machine: The simulation skill compares torso height h above the lowest body with the corresponding height H of the default upright pose. It suspends the task prompt when $h < 0.65H$ and restores it when $h > 0.9H$. During recovery, every future slot requests the canonicalized default upright pose for all bodies, including torso orientation; velocity and style are masked.

The first slot's arrival time is

$$\tau_0(h) = \mathrm{clip}\left(\frac{H - h}{v_{\mathrm{rise}}}, 0.25\,\mathrm{s}, 1\,\mathrm{s}\right). \tag{15}$$

Here, $v_{\mathrm{rise}} = 0.8\,\mathrm{m/s}$ sets the requested arrival time, not the robot's actual vertical velocity. Each later slot adds $0.25\,\mathrm{s}$, up to $2\,\mathrm{s}$. On exit, the skill restores the suspended task prompt. Only the prompt changes between the task and recovery states.

c) G1 recovery: The deployment skill follows the same height-triggered structure but requests only torso height, leaving torso planar position, all orientations, and limb targets unconstrained. Each of five future slots requests the calibrated upright torso height; velocity and style cues are masked. The first arrival time uses the height-gap rule in eq. (15), with an upper bound of $1.25\,\mathrm{s}$ rather than $1\,\mathrm{s}$. Later slots add $0.25\,\mathrm{s}$ each, capped at $2\,\mathrm{s}$.

Torso height is computed by FK relative to the lowest reconstructed body point. Recovery begins below $0.65H$ and ends above $0.90H$, where H is the calibrated upright torso height. Dense tracking and unconditional generation disable automatic recovery. During headset-controlled crouching, height is measured relative to the lower foot and the entry threshold is based on the commanded torso height, bounded below by $0.65\,\mathrm{m}$. The exit threshold remains $0.90H$, allowing intentional crouching while requiring recovery to finish upright.

H. Deployment on the G1

The C++ deployment stack uses TensorRT with CUDA graphs on a laptop equipped with an NVIDIA RTX 5090 GPU. The measured latency from sensor acquisition to joint-command transmission is $1.3\,\mathrm{ms}$; this is the sensor-to-command processing time, not the control period. An Xbox controller and a PICO headset supply runtime commands.

TABLE VI: Harness skills and the prompts they construct. Each embodiment uses one frozen policy across its supported skills. The recovery row describes the simulation state machine; hardware get-up demonstrations are reported in section VI-B.

Skill	Input	Prompt fields	Update
Dense tracking	reference clip	timed body positions and orientations	reference clock
Steering	joystick velocity and turn	planar velocity; five timed torso, heading, and alternating foot targets	every step
Path following	waypoint list	timed torso waypoints and planar velocity	receding horizon
Location	target position or orientation	timed body constraint; planar position, height, heading, or full anchor orientation	on target change
Style	example clip	unordered motion-state cue	on reset or override
Sparse goal	pose from a trajectory	one timed future pose	on target transition
Recovery (simulation)	torso height	canonicalized upright pose for all bodies; velocity and style masked	height-triggered transitions

All hardware demonstrations use the deterministic prior mean: we set the sampling noise to zero, project μ_t onto the unit sphere, and decode it together with d_t. This differs from the sampled latent used during distillation. The deployment stack also supports stochastic sampling, but that option is not used for the reported hardware demonstrations.

G1 follows BeyondMimic [7] for action representation. The setpoint is $q_j^{\mathrm{des}} = q_j^0 + \alpha_j a_j$, where $\alpha_j = 0.25\tau_j^{\max}/k_{p,j}$. These offsets are relative to the nominal pose, not the reference motion. The policy updates them every 20 ms (50 Hz), with four simulator physics steps per action. SOMA instead uses $q_j^{\mathrm{des}} = \pi \tanh(a_j)$ with fixed PD gains. Deployment must preserve joint order, offsets, scales, and gains. World-position skills additionally consume a localization estimate; canonicalized skills use the local pose convention above.

Control-mode selection updates the goal-related observations and masks while retaining the same policy weights and joint-control interface. For the sitting demonstrations, the robot is placed near the object and supplied with a nearby-object cue. This setup provides task context externally rather than estimating the object location or planning an approach autonomously.

I. Evaluation Protocol

Reference reconstruction and task execution evaluate different capabilities. Reconstruction uses the posterior with privileged reference information, whereas task and hardware evaluations use the prior. Task comparisons use matched initial states, commands, perturbations, horizons, and success criteria. The reference-tracking comparison excludes synthetic recovery data; the four-expert policy includes recovery training.