

Figure S1: Systemic overview of the training scheme of our whole-body controller. We employ a multi-objective learning framework plus a GAN-like architecture for imitation learning. Our scheme allows imitating arm poses (orange) and those of the rest body parts (green) from different sources of reference motions simultaneously, and uses a goal-directed reward to fulfill the sparse tracking task for the head and hand poses.

Table S1: Hyperparameters

Parameter	Value
policy network learning rate	5×10^{-6}
critic network learning rate	1×10^{-4}
discriminator learning rate	1×10^{-5}
reward discount factor (γ)	0.95
GAE discount factor (λ)	0.95
surrogate clip range (ϵ)	0.2
gradient penalty coefficient (λ^{GP})	10
number of PPO workers (simulation instances)	1024
PPO replay buffer size	1024×8
PPO batch size	256
PPO optimization epochs	5
discriminator replay buffer size	$1024 \times 8 \times 2$
discriminator batch size	512

A Whole Body Controller

Figure S1 shows the overview of the systemic architecture of the whole body controller. We use IsaacGym [43] as the physics engine for policy training. The policy runs at 30Hz and controls the humanoid through a PD servo.

Instead of directly imitating full-body motions, based on the motion decoupling scheme from previous literature [38], we split the full-body motions into arm and torso groups, and employ two discriminators at the same time to evaluate the imitation performance for partial motions. Besides the current state of the humanoid, the control policy takes only the tracking target for the head and two hands as the goal input. Without needing the whole trajectory of full-body tracking, the discriminators evaluate the imitation performance of partial motions and allow the arm motions to be combined with the torso and lower-body motions from different motion clips in a free way.

Given an additional goal-directed reward for target tracking plus regularizations for sim-to-real consideration, we leverage the multi-objective learning framework [38] to balance the learning of multiple imitation and goal-directed objectives. The final optimization objective for policy training can be written as

$$\max \mathbb{E}_t \left[\sum_{\kappa} w_{\kappa} \bar{A}_{t,\kappa} \log \pi(\mathbf{a}_t | \mathbf{o}_t) \right] \quad (\text{S1})$$

where $\bar{A}_{t,k}$ is the standardized advantage that is estimated according to the achieved reward of each objective k , w_k is an associated weight, and \mathbf{o}_t is the observation including the humanoid’s state \mathbf{s}_t and the goal state \mathbf{g}_t . We choose $w_{\text{imit},i} = 0.2$ for each of the two imitation objectives and $w_g = 0.6$ for the goal-directed objective.

We use PPO [46] as the backbone reinforcement learning algorithm and take the Adam optimizer [47] to perform network optimization for policy training. The hyperparameters used for policy training are listed in Table S1 and the network structures are shown in Figure S2. We manually pick 1363 clips of locomotion and loc-manipulation motions from AMASS [36] and OMOMO [37]. The whole data set of motions is around 5 hours long.

A.1 Observation Space

The G1 humanoid has 33 links and 27 controllable joints, where the wrist and neck joints are fixed and the waist part only has 1 degree of freedom around the yaw axis. We take two historical frames as the input and process the state vector via a GRU [48]. This leads to a state space of $\mathbf{s}_t \in \mathbb{R}^{33 \times 7 \times 2}$ including the position and orientation (in quaternion) of each link in the local frame of the humanoid’s root link, and an action space of $\mathbf{a}_t \in \mathbb{R}^{27}$.

For control purposes, we take the root angular velocity and joint velocity locally as the control state $\mathbf{c}_t \in \mathbb{R}^{30}$. We ignore the linear velocity of the root link, since the linear velocity is hard to access from the humanoid when deployed in the real world.

The goal state vector $\mathbf{g} \in \mathbb{R}^{3 \times 3 \times 7}$ includes the target positions and orientations (in quaternion) of the three links (left hand, right hand and head) in the next three frames.

The final observation space \mathbf{o}_t is composed of the three components \mathbf{s}_t , \mathbf{c}_t , and \mathbf{g}_t .

A.2 Reward Terms

Instead of using a single discriminator for each motion group, we take the GAN-like architecture from ICCGAN [34], and employ an ensemble of 32 discriminators for each motion group. The imitation-related reward of the discriminator ensemble D_i is computed via

$$r_t^{\text{imit},i}(\bar{\mathbf{s}}_t^i, \bar{\mathbf{s}}_{t+1}^i) = \frac{1}{N} \sum_{n=1}^N \text{CLIP}(D_n^i(\bar{\mathbf{s}}_t^i, \bar{\mathbf{s}}_{t+1}^i), -1, 1), \quad (\text{S2})$$

where the subscript i indicates different imitation objectives (upper or lower body groups), $\bar{\mathbf{s}}_t^i$ is the partially observable character state for the imitation objective i , and the discriminator ensembles N discriminators each which is trained using hinge loss [49] with gradient penalty [50]. We choose $N = 32$ in our implementation.

The goal-directed reward mainly measures tracking errors and also consists of three terms to stabilize the motions for sim-to-real consideration:

$$r_t^g = r_{\text{tracking}} + 0.8r_{\text{in-air}} + 0.5r_{\text{sliding}} + 0.005r_{\text{energy}}. \quad (\text{S3})$$

For simplicity, here we omit the subscript t for each of the reward terms.

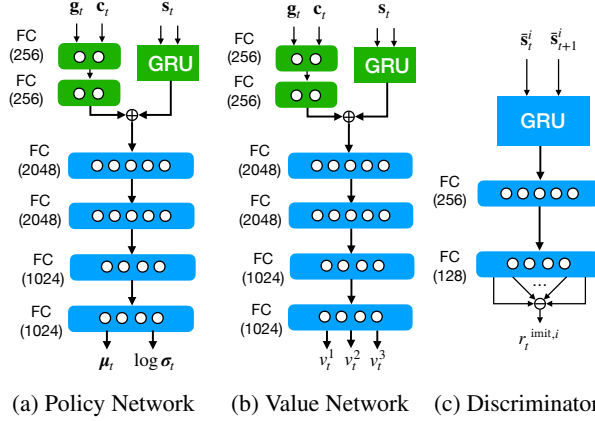


Figure S2: Network structures. We use \oplus denoting the add operator and \ominus denoting the average operator. For multi-objective learning, we employ a value network with 3-dimensional output for the two imitation objectives and one goal-directed objective.

Table S2: Domain Randomization Settings

Parameters	Value
Base Mass (kg)	$[-3, 3]$
Body Link Friction Coefficient	$[0.5, 1.25]$
Scale on PD Servo Gain (Kp)	$[0.7, 1.3]$
Scale on PD Servo Damping (Kd)	$[0.7, 1.3]$
Action Delay Ratio (ρ_{delay})	0.5

500 r_{tracking} is the reward measuring the tracking error:

$$r_{\text{tracking}} = 0.5 \exp \left(-\frac{5}{|\mathcal{I}|} \sum_{i \in \mathcal{I}} |\varepsilon_{\text{pos},i}| \right) + 0.5 \exp \left(-\frac{2}{|\mathcal{I}|} \sum_{i \in \mathcal{I}} |\varepsilon_{\text{orient},i}| \right) \quad (\text{S4})$$

501 where $\mathcal{I} = \{\text{left hand, right hand, head}\}$ is the set of links under tracking, $\varepsilon_{\text{pos},i}$ is the position error
 502 between the current position of the link i and its target position measured in Euclidean distance, and
 503 $\varepsilon_{\text{orient},i}$ is the orientation error measured by the angle between the orientation of the link i and the
 504 target.

505 $r_{\text{in-air}} = \min\{0, t_{\text{in-air}} - 0.5\}$ encourages the policy to keep the foot in the air for at least 0.5s during
 506 stepping. It is computed for the swing foot when it contacts the ground, and $t_{\text{in-air}}$ is the hanging
 507 time of that foot before the contact.

508 $r_{\text{sliding}} = -\sum_f c_f \|\mathbf{v}_f\|_2$ penalizes the linear velocity of the foot link f if it contacts the ground,
 509 where $c_f = 1$ or 0 indicating the contact state of the foot f .

510 $r_{\text{energy}} = -\sum_j (0.1|\tau_j v_j| + 0.005\tau_j^2)$ penalizes the energy cost for each joint j , where τ_j is the
 511 torque applied on joint j and v_j is the joint’s rotation velocity.

512 A.3 Domain Randomization

513 Parameters for domain randomization are listed in Table S2.

514 When testing the sim2real transfer, we found adding simulated delay during training time essential to
 515 prevent the robot from jittering. Compared to using a random delay across all environments, which is
 516 challenging to train, we found using a constant 1-step delay on a fixed portion ρ_{delay} of environments
 517 improves training speed, and can prevent the robot from jittering caused by the uncertainty delay
 518 during policy execution in the real world. We choose $\rho_{\text{delay}} = 0.5$, which means that the action
 519 delay is applied on half of the training environments.

520 B Image Augmentation of Navigation Data

521 We augment human data collected by Aria Glasses to resemble the robot’s view. First, we address
 522 image discrepancies caused by differences in capture devices. Aria Glasses use an RGB fisheye
 523 camera with a 110° horizontal and vertical field-of-view (HFOV, VFOV), while the robot camera
 524 has a 90° HFOV and 67.5° VFOV. We undistort the fisheye images to obtain I_u , which approxi-
 525 mates a pinhole camera view with intrinsics K_u . Second, we address discrepancies in camera pose
 526 (extrinsics) due to morphological differences. Humans tend to tilt their heads—e.g., looking down-
 527 ward when reaching for an object—whereas the robot’s navigation camera, fixed at the top of the
 528 head, always looks forward. As a result, the same object may appear in different regions of the
 529 image despite similar camera positions. To align viewing angles, we apply a homography to the
 530 undistorted image I_u to match the pitch angle. Given the robot camera’s intrinsics K_r and Aria’s
 531 effective intrinsics K_u , we apply $H = K_r R_{\text{pitch}} R_{[\theta]}^T K_u^{-1}$, where $R_{[\theta]}$ is the pitch component of the
 532 current human camera pose and R_{pitch} is the desired robot pitch.

533 **C Usage of motion capture system**

534 Although our whole-body controller and navigation model don't rely on world coordinates, in our
535 real-world experiment, we found it hard to keep the robot standing at the exact location without
536 drifting. Therefore, we attach mocap markers only to the robot's head and transform the camera
537 trajectory into the world frame; we found that having closed-loop tracking in the world frame can
538 significantly improve the accuracy of robot reaching.