

Supplementary for Embrace Contacts: humanoid shadowing with full body ground contacts

Anonymous Author(s)

Affiliation

Address

email

1 We attach a video describing the main idea and the real-world experiments in real-time with no
2 accelerations. We perform real-world tests on all three types of motions. We encourage readers to see
3 the video for a more comprehensive understanding of this work.

4 1 More Results

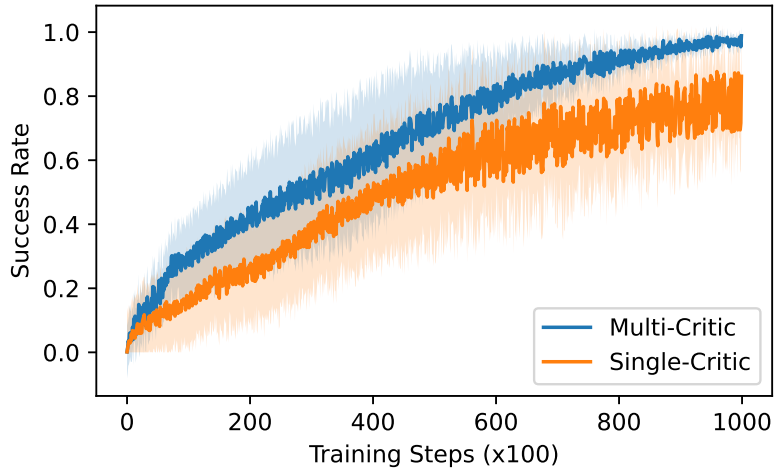


Figure 1: The success rate reported during training when trained on all 4 extreme difficult contact-agnostic motion.

5 Figure 1 shows the training progress comparing multi-critic technique and single-critic technique.
6 Using multi-critic setting leads to faster training speed as well as faster convergence rate.

7 2 Simulation Details

8 Considering these extreme difficult humanoid motion reference data may lead to physically infeasible
9 situations, such as body parts penetrating the ground or robot base floating in the air, we set the
10 robot’s joint positions as the retarget robot pose in the first motion reference frame. We initialize
11 the robot’s base position by first adding a positive height to all motion reference frame so that no
12 motion reference frame is penetrating the ground. Then we add a tiny height offset to spawn the
13 robot, typically 0.06m. Also, to help the policy experience more states if the policy stuck as some
14 place, for example if it does not get up from the ground, we sample the initial pose of the robot not
15 only from the first frame of the motion reference trajectory, but from the start of the motion reference
16 to the 60% of it.

During each rollout, we select one motion reference to generate the motion command. At time t , we use a pre-sampled time interval t_{int} and sample the motion reference at $t + t_{\text{int}}, t + 2t_{\text{int}}, t + 3t_{\text{int}}, t + 4t_{\text{int}}, t + 5t_{\text{int}}$ respectively. We also compute the base reference position and orientation in the base frame at time t .

	Names	Value
Environments	Number of robots	4096
Domain randomization	Scaling body mass	0.8 \sim 1.2
	Center of mass position	-0.02m \sim 0.02m
	Scaling motor stiffness	0.9 \sim 1.1
	Scaling motor damping	0.9 \sim 1.1
	Motor delays	0.0s \sim 0.03s
Initialize pose	Height offset	0.04m
	Sampling frame ratio from the trajectory	0.0 \sim 0.6

Table 1: Detailed parameters for running the system in the simulator

3 Training Details

Dataset	Subject	Motion names
CMU	140	Get up from ground
KIT	3	Crawling
Internet Video	Bilibili BV1L34y1t71x	Popping dance movement
Internet Video	BiliBili BV1Nm4y1k7wP	Breaking dance movement
Internet Video	Bilibili BV1mv411G7WM	Jiu-jitsu movement

Table 2: The motion reference we select to build fine motion command dataset.

As described in the main text, we select a handful of motion commands to train the humanoid whole-body controller to overcome the unbalanced issue of the precollected dataset. Shown in Table 2, we train our policy in simulation by extracting motion command from these motion references.

Reward group	Reward term	Expression
Task	Base position tracking	$\Psi(\Delta(p_b), 0.4)$
	Base orientation tracking	$\Psi(\Delta(q_b), 0.8)$
	Joint position tracking	$\Psi(\ \theta^j - \hat{\theta}^j\ , 0.3)$
Regularization	Action rate	$\Psi(\ a_t^j - a_{t-1}^j\ , 1.0)$
	Joint acceleration	$\Psi(\ \ddot{\theta}^j\ , 500)$
	Joint velocity	$\Psi(\ \dot{\theta}^j\ , 15)$
Safety	Joint position limit	$\Psi(\max(\theta^j - \theta_{\max}^j, \theta_{\min}^j - \theta^j), 0.1)$
	Joint torque limit	$\Psi(\max(\tau^j - 0.9 \tau_{\max}^j, 0), 0.1)$

Table 3: Reward terms and their expressions

In Table 3, the function Ψ is a Gaussian kernel where,

$$\Psi(a, b) = \exp(-a/b^2) \quad (1)$$

Shown in Table 3, we build these reward functions in the range of 0 1 so that everything is positive, potentially preventing active termination behavior. Then we **multiply** all reward terms in each reward group so that the algorithm will not completely ignore any of these terms. For the experiment variant using single critic, the reward terms within each group are multiplied and the reward groups are sum together weighted by the same weight parameters of the advantage mixing to get the scalar reward.

The policy network consist of actor and multiple critics with the same structure. We use a transformer-based encoder block to encoder all motion command. The encoder outputs a sequence of embedding, which we select the embedding whose ‘time-to-target’ attribute is the smallest positive value. We

Hyperparameters	Value
Optimizer	AdamW
β_1, β_2	0.9, 0.999
Learning rate	$1e-4$
Batch size	4096
Clip param	0.2
Entropy coefficient	0
min_std clip	0.2
Desired KL	0.01
Maximum gradient norm	1
Num minibatches	4
γ	0.99
λ	0.95
Advantage mixing weights	[0.7, 0.1, 0.2]

Table 4: Parameters in Algorithm implementation

Hyperparameters	Value
Encoder Activation	GELU
Encoder Project Activation	ReLU
Encoder num heads	1
Encoder num layers	2
Encoder d_model	128
Encoder feedforward dimension	128
Encoder output size	128
MLP hidden sizes	[512, 256, 256]
MLP Activation	ELU

Table 5: The detailed network parameters for the low-level policy, which runs onboard

then concatenate this embedding with a stacked history proprioception observation and feed them to a Multi-Layer Perceptron. The MLP layers outputs the 29-dof action as the target position to the robot motors. Detailed parameters for the network are shown in Table 5.

We train our algorithm on a Nvidia 4090D GPU with 4096 robots in parallel for about 72 hours from scratch. We build the simulation environment using IsaacLab and modify the reinforcement learning framework based on rsl_rl.

4 Deployment and Real-World Experiment Details

Joint name	Stiffness (kp)	Damping (kd)
Left/right shoulder pitch/roll/yaw	25	1.0
Left/right elbow	25	1.0
Left/right wrist roll	25	1.0
Left/right wrist pitch/yaw	5	0.5
Waist roll/pitch	60	2.5
Waist yaw	90	2.5
Left/right hip pitch/roll/yaw	90	2.0
Left/right knee	140	2.5
Left/right ankle pitch/roll	20	1.0

Table 6: Parameters that runs on the hardware

To run the trained policy on the real robot, we deploy the entire system on an Nvidia Jetson Orin NX and a laptop running Intel i5 CPU. We export the policy (including the transformer-based encoder) as an ONNX program. All components communicate using ROS2 in the network of Unitree G1 robot. We then run the policy on the Jetson board at 50Hz. Since the policy outputs the action as the target joint position of each motor on the robot, we use the built-in PD controller on the Unitree G1 robot,

which runs at 1000Hz, with the kp/kd setting as shown in Table 6. These kp/kd parameters are also the same when training in simulator.

To acquire the target link position and their error respectively, we use Pytorch_Kinematics [1] and ONNX [2] to export the forward kinematics computation as an ONNX program. The exported ONNX program gets the joint positions and outputs the target link positions in the robot’s base frame, which runs in real time on Nvidia Jetson Orin NX.

Since this work is also a proof-of-concept for building a hierarchical general humanoid controller, with a low-level whole-body control policy and a high-level command sender, we use another laptop to send the high-level command which simultaneously test the communication latency. Considering our high-level motion command is defined with base pose sequence under the robot frame when the command is generated, the high-level motion command for the real-world testing cannot be played directly from SMPL-based motion file. We play each motion in simulation using the well-trained low-level policy and record the motion command, as well as the base pose command under the robot’s base frame in simulation. We then play this base pose command in the real world and ignore the difference between the robot trajectory in the simulator and in the real world.

In the real-world testing, it is important to determine whether the testing motion succeeded, while we don’t install additional motion capture system. For each extreme motion, we determine the success of each motion as finishing the entire motion command sequence with no unexpected head contacting the ground. For getting-up-from-ground task, we terminate the test when the robot’s torso orientation significantly deviate from the motion command. In our real-world experiment, we also visualize the motion command in the laptop that sends the motion command sequence.

5 Data distribution analysis

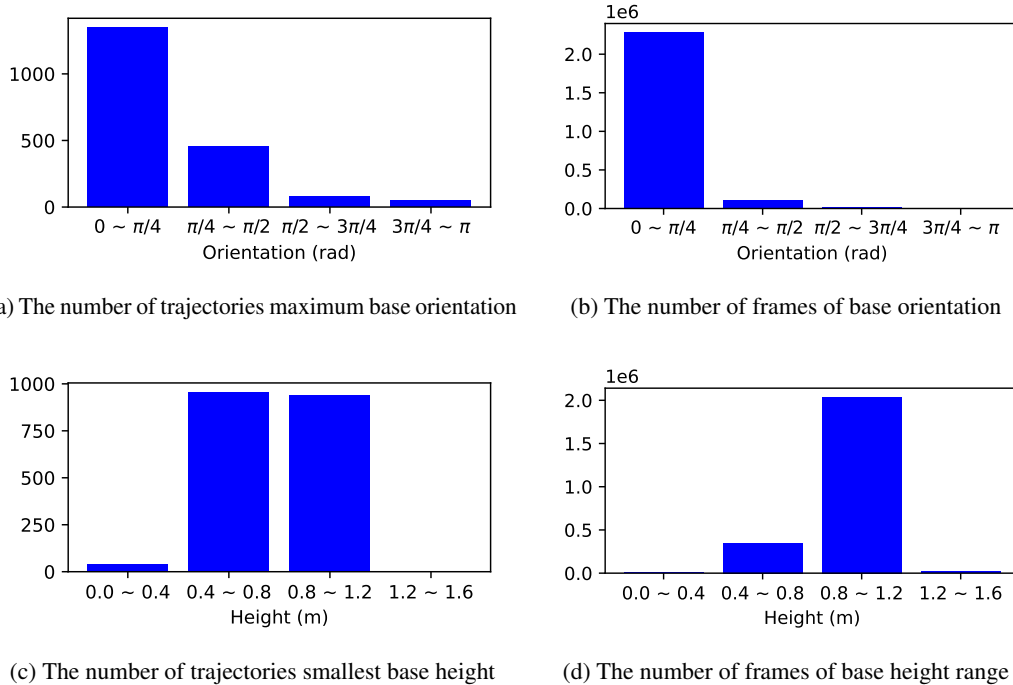


Figure 2: Histogram of the number of motions in terms of their maximum roll/pitch and the minimum base height.

As shown in Figure 2, we count the number of frames and the number of files whose maximum base orientation and minimum base heights. Figure 2c counts the number of motion files whose minimum base height reaches a certain range. As shown in Figure 2a and Figure 2b, most of the motion files

71 are performed when the base position is standing straight. As shown in Figure 2c and Figure 2d, most
72 of the motions are performed when the robot base is over 0.4m. In this case, not many contact-rich
73 motion is presented in the dataset.

74 **References**

- 75 [1] S. Zhong, T. Power, A. Gupta, and P. Mitrano. PyTorch Kinematics, Feb. 2024.
- 76 [2] O. R. developers. Onnx runtime. <https://onnxruntime.ai/>, 2021.