

UMI-on-Legs: Making Manipulation Policies Mobile with a Manipulation-Centric Whole-body Controller

Anonymous Author(s)
Affiliation
Address
email

1	Contents	
2	1 Things that did not work	2
3	1.1 Privileged policy distillation and observation history.	2
4	1.2 Precise grasping for tossing.	2
5	1.3 System Reliability for Fully-untethered Deployment.	2
6	1.4 Velocity integration	2
7	2 Deployment	2
8	2.1 iPhone placement	3
9	2.2 Robot URDF	3
10	2.3 Latency	3
11	2.4 Safety	3
12	3 Training	4
13	3.1 Manipulation Policy	4
14	3.2 Whole-Body Dynamic Tossing	4
15	3.3 End-effector Reaching Leads to Robust Whole-body Pushing	4
16	3.4 Plug-and-play Cross-Embodiment Manipulation Policies	4
17	3.5 In-the-wild Cup Rearrangement Evaluation	4
18	3.6 Reward Terms	5
19	4 Evaluation	5
20	4.1 Real World Tossing	5
21	4.2 Simulation Ablations	6
22	5 Misc.	6

23 1 Things that did not work

24 In this section, we discuss things we tried, but did not improve the performance or caused other issues. We
25 hope that through openly discussing our unsuccessful attempts, the community could gain a more complete
26 understanding of the project, learn from our mistakes, and improve upon our attempts. To complement
27 this section, we’ve also included robot failure videos on our [supplementary website](#).

28 1.1 Privileged policy distillation and observation history.

29 We tried to include privileged information (kp, kd, friction, damping, ground truth poses, base velocities,
30 etc.) to train a privileged policy and distill it through supervised learning and online regularization.
31 However, it didn’t provide performance boost and introduce instability when we include observation
32 history longer than 1 step. This could be because that the Python ROS2 timer is imprecise, thus the
33 observation history can easily be out of distribution given the wrong history timestamps. This was less
34 of an issue on the Unitree A1 platform, which ships with a real-time kernel.

35 1.2 Precise grasping for tossing.

36 To train diffusion policies for grasping and tossing, we collected 500 episodes of human demonstration.
37 However in the grasping phase, the small shakes from the controller results in distributional drift in
38 visuo-motor manipulation policy. We hypothesize that robust, fully-autonomous grasping and tossing
39 could be achieved with more data.

40 1.3 System Reliability for Fully-untethered Deployment.

41 Although the system is capable of various tasks in the real world without any external cables, the reliability
42 is in general not high enough. We summarize some hardware and system level challenges we encountered
43 as follows. Given a small base-to-arm weight ratio, the joints of the dog are likely to overheat in 10 to 30
44 minutes according to the robot posture and then goes to an error mode. Therefore, we needed to frequently
45 cool down the motors and fine-tune the policy to a more energy-efficient posture. On the power front,
46 the battery supplies different voltages depending on how full it is. When fully charged, the voltage is too
47 high for the arm, and the setup requires a voltage adaptor. When closer to empty, the policy’s behavior
48 is more dampened.

49 1.4 Velocity integration

50 We used the iOS ARKit to run VIO on an iPhone 15 Pro. However, the estimated camera pose sometimes
51 drift heavily under dynamic actions including tossing. We also measure the delay of the pose estimator.
52 Although the Ethernet communication time (between the iPhone and the onboard Jetson) is negligible,
53 we found the the latency from the movement to the pose update is still roughly 140ms. This latency
54 introduced a significant Sim2Real gap during deployment, which manifests as low frequency oscillations
55 which slowly diverges.

56 Using low-latency foot contacts, joint position/velocities, and IMU readings, we can estimate the base
57 velocity for each timestamp. Using base velocity estimates from 140ms into the past, we can integrate
58 every received iPhone pose forward in time by 140ms. However, the shaking behavior due to the latency is
59 not addressed, which can only be mitigated by higher action rate regularization fine-tuning. As the ARKit
60 pose estimation runs at 60Hz, we expect a better implementation could achieve shorter latency.

61 2 Deployment

62 In this section we elaborate important details in real-world deployment and Sim2Real transfer.

63 2.1 iPhone placement

64 We mounted the iPhone on the back of the robot with following considerations: 1) Since the iPhone is
65 facing back, the robot arm will not be in the view during manipulation, so ARKit’s visual-inertial odometry
66 can better track the surrounding environment. 2) iPhone is mounted at a fixed angle of 60° to the back
67 plane of the Go2, thus the camera is pointing upwards even when Go2 is slightly bending down. This
68 increases the number of visual features in the iPhone camera, thus provides more robust tracking compared
69 to our original design, which was a 90° mount. 3) This position is kinematically unreachable for the robot
70 arm, so the arm will unlikely damage the iPhone.

71 2.2 Robot URDF

72 For our highly dynamic tasks, we observed that the domain randomization and adaptive policies [1, 2] were
73 insufficient to bridge the gap between a heavily mis-specified model (provided by the manufacturer) and the
74 real system, a gap that is exacerbated when performing dynamic arm movements. We found that disassembling the ARX5 arm, reweighing each component, and recomputing mass, center of mass and inertia matrices in the URDF* were crucial for both simulation stability as well as successfully real-world deployment.

77 2.3 Latency

78 While all observations and actions are perfectly synchronized in simulation, the communication and code
79 runtime of the real-world robot system introduce a significant amount of latency in different aspects.

80 The most important latency comes from the robot joint state observation and the joint command execution.
81 This includes the motor encoder readout, ROS2 communication, whole-body controller inference, action
82 sent back to motors and being executed on motors. Since we were unable to exactly measure all the latency
83 sources, we swept the end-to-end latency from 0ms to 30ms with 5ms interval in simulation and find
84 that 20ms works the best in our real-world system. We observed high frequency shaking if the latency
85 is mismatched in simulator and real-world robots.

86 Robot pose estimators (motion capture and iPhone) also introduce latency. We observed an 8ms latency from
87 the motion capture system and 140ms latency from the iPhone ARKit. We simulated a 10ms pose latency in
88 simulation to close the sim2real gap. We also implemented inertial-legged velocity estimation to integrate
89 the latest pose for iPhone, but the performance didn’t improve significantly. See 1.4 for detailed explanation.

90 We also observe latency in the Python ROS2 program. Due to the global interpreter lock, all the Python
91 ROS2 callback functions have to run alternately, so one callback function will block the others if it takes
92 too long. We optimize the run time of all callback functions and detach the callback functions that take
93 too long to other ROS2 nodes. This allows the joint observation update to run closer to 200Hz and policy
94 inference closer to 50Hz. A C++ implementation with more precise timer and lower latency can achieve
95 better performance given the same checkpoint and evaluation setup.

96 2.4 Safety

97 We observed that directly deploying the whole-body controller checkpoints in the simulation section led
98 to some safety issues and addressed them by fine-tuning the checkpoints with more conservative reward
99 schemes:

- 100 • **Shaking:** To reduce the shaking and oscillation behavior of the robot, we increased the action
101 rate regularization. The controller will get less reward if the predicted action is farther away from
102 the previous action. We also disabled the on board lidar that introduces shaking behavior.
- 103 • **Overheat Shutdowns:** The calf joint of the robot uses a linkage configuration rather than directly
104 applies the output torque. Therefore, the required output torque of the motor is higher than the

*We will open-source all code, data and checkpoints after publication.

other joints, leading to frequent overheating and emergency shutdown. We observed that increased torque regularization during training allows the controller to run up to 30 minutes continuously[†]

- **Unsafe Configurations:** In the simulator, the controller is likely to twist the legs to achieve higher precision with less movements, but this makes it unsafe to deploy in the real world. We added reward terms to regularize the body to a more balanced pose, so that the center of mass can roughly stay in the center.

3 Training

3.1 Manipulation Policy

In this section, we report hyperparameters used for training our manipulation diffusion policy. Visual observation and proprioception horizon means how many history image and robot states (with 0.1s interval) are used as input to the policy. Output Steps means the action length of the diffusion policy output. Execution Steps and Execution Frequency indicate how many steps of the diffusion policy output is actually executed in the real-world robot. To enable more stable end-effector trajectory updates, we add linear interpolation between the executed trajectory and the newly-updated trajectory for a duration of “Trajectory Update Smoothing Time”.

3.2 Whole-Body Dynamic Tossing

We increased the inference and execution steps for a smoother toss, since the trajectory is less likely to update during the dynamic toss. We also increased the execution frequency to get farther toss. Please refer to Table 1 for detailed parameters.

Hyperparameters	Values
Training Set Trajectory Number	500
Diffusion Policy Visual Observation Horizon	2
Diffusion Policy Proprioception Horizon	4
Diffusion Policy Output Steps	64
Diffusion Policy Execution Steps	40
Diffusion Policy Execution Frequency	12Hz
Trajectory Update Smoothing Time	0.1s

Table 1: Hyper parameter set of the dynamic tossing task.

3.3 End-effector Reaching Leads to Robust Whole-body Pushing

Since the trajectory motion is much simpler, we collected fewer human demonstrations for the diffusion policy. We increased the “Trajectory Update Smoothing Time” to prevent heavy shakes due to the trajectory update. Please refer to Table 1 for detailed parameters.

3.4 Plug-and-play Cross-Embodiment Manipulation Policies

As the UMI cup rearrange task requires much higher precision but is quasi-static, we decreased the execution frequency for better stability. As mentioned in the main text, we directly use the publicly released checkpoint from Chi et al.. We did not collect any extra data for this task.

3.5 In-the-wild Cup Rearrangement Evaluation

We evaluated the whole-body controller on the cup rearrangement task in the wild. We tested it in some challenging scenarios including unstable terrain (grass, dirt, collapsible table), direct sunlight which made

[†]Please check out evaluation videos on <https://corlpaper.github.io>, which shows our controller running continuously for 20-30 minutes.

Hyperparameters	Values
Training Set Trajectory Number	25
Diffusion Policy Visual Observation Horizon	2
Diffusion Policy Proprioception Horizon	2
Diffusion Policy Output Steps	32
Diffusion Policy Execution Steps	10
Diffusion Policy Execution Frequency	10Hz
Trajectory Update Smoothing Time	0.3s

Table 2: **Hyper parameter set of the whole-body pushing task.**

Hyperparameters	Values
Training Set Trajectory Number	1400
Diffusion Policy Visual Observation Horizon	1
Diffusion Policy Proprioception Horizon	2
Diffusion Policy Output Steps	16
Diffusion Policy Execution Steps	8
Diffusion Policy Execution Frequency	5Hz
Trajectory Update Smoothing Time	0.1s

Table 3: **Hyper parameter set of the UMI cup rearrangement task.**

the robot easily overheated, and unseen tables and cups to show the generalizability of the diffusion policy. Please checkout the supplementary video for more details.

3.6 Reward Terms

During RL training of the whole body controller, we include the following reward terms:

- **Joint Limit, Joint Acceleration, Joint Torque, Root Height, Collision, Action Rate:** We use the same definition as prior works [2, 4].
- **Body-EE Alignment:** We regularize joint 0 and joint 3 of the arm (the joints that mostly affect the yaw angle of the gripper) to stay close to their initial pose. This allows the arm to be aligned with the body.
- **Even Mass Distribution:** We regularize the standard deviation of the force of the four feet to be lower. The motors are less likely to overheat if the body mass is distributed more evenly on the four legs.
- **Feet Under Hips:** We regularize the feet’s planar position to be close to that of their respective hips, for all four legs. This regularization improves the stability of the standing pose.
- **Pose Reaching:** We minimize the position error ϵ_{pos} and orientation error ϵ_{om} to the target pose using an unified reward function: $\exp(-(\frac{\epsilon_{\text{pos}}}{\sigma_{\text{pos}}} + \frac{\epsilon_{\text{om}}}{\sigma_{\text{om}}}))$. We apply σ curriculum to the reward functions: as the error get smaller, we decrease σ for a more peaky reward function, which encourages the controller to further reduce reaching error. σ_{pos} is set to $[2, 0.1, 0.5, 0.1, 0.05, 0.01, 0.005]$ when the position error is smaller than $[100, 1.0, 0.8, 0.5, 0.4, 0.2, 0.1]$ respectively; σ_{om} is set to $[8.0, 4.0, 2.0, 1.0, 0.5]$ when the orientation error is smaller than $[100.0, 1.0, 0.8, 0.6, 0.2]$ respectively.

4 Evaluation

4.1 Real World Tossing

To achieve a pre-grasped state during tossing evaluations, we handed the policy the tennis ball. We count episodes where the robot falls during the toss as a failure. Finally, we measure the distance from the

Name	Weight
Joint Limit	-10
Joint Acceleration	-2.5e-7
Joint Torque	-1e-4
Root Height	-1
Collision	-1
Action Rate	-0.01
Body-EE Alignment	-1
Even Mass Distribution	-1
Feet Under Hips	-1
Pose Reaching	4

Table 4: **Reward terms.**

location where the ball first lands to the center of the bin, and report a success if it lands within 40cm of the bin’s center.

4.2 Simulation Ablations

In each episode, a random grasp and toss end-effector trajectory is sampled from our dataset (§ 3.2), the robot is uniformly randomly initialized in the range of [-5.0cm,5.0cm] for x and y position, and [-0.1rad,0.1rad] for z orientation, and its initial joint positions are sampled uniformly within 5% of that joints’ full range, centered around the default joint configuration. Position and orientation errors are averaged over all timesteps, while survival is 1 only if the policy lasts for 17 seconds without a terminal collision. Here, a terminal collision is defined as any robot part contact that is not the robot’s feet or gripper.

Similar to training, evaluation includes a pose latency of 10ms and the full range of domain randomization reported in Table 5.

Hyperparameters	Values
Init XY Position	[-0.1m,0.1m]
Init Z Orientation	[-0.05rad,0.05rad]
Joint Damping	[0.01,0.5]
Joint Friction	[0.0,0.05]
Geometry Friction	[0.1,8.0]
Mass Randomization	[-0.25,0.25]
Center of Mass Randomization	[-0.1m,0.1m]

Table 5: **Domain Randomization Hyperparameters.**

All approaches were trained for 4000 iterations, and the performance at checkpoint 4000 is reported. Power usage report is electrical power usage based on real hardware’s voltages, manufacturers’ reported torque constants, and the simulation’s motor torques.

5 Misc.

Cost of the system. We report the cost of our entire robot system based on the market price in Table 6, which is roughly 1/4 of other quadruped manipulation systems in [5, 6]

References

- [1] A. Kumar, Z. Fu, D. Pathak, and J. Malik. Rma: Rapid motor adaptation for legged robots. *arXiv preprint arXiv:2107.04034*, 2021.
- [2] Z. Fu, X. Cheng, and D. Pathak. Deep whole-body control: Learning a unified policy for manipulation and locomotion. In *Conference on Robot Learning*, pages 138–149. PMLR, 2023.

Item	Cost(\$)
Unitree Go2 Edu Plus	12,500.00
ARX5 Robot Arm	10,000.00
GoPro Hero9	210.99
GoPro Media Mod	79.99
GoPro Max Lens Mod	68.69
iPhone 15 Pro	999.00
Elgato Capture Card	147.34
Total	24,006.01

Table 6: **System Cost.**

- 181 [3] C. Chi, Z. Xu, C. Pan, E. Cousineau, B. Burchfiel, S. Feng, R. Tedrake, and S. Song. Universal
182 manipulation interface: In-the-wild robot teaching without in-the-wild robots. In *Proceedings of*
183 *Robotics: Science and Systems (RSS)*, 2024.
- 184 [4] N. Rudin, D. Hoeller, P. Reist, and M. Hutter. Learning to walk in minutes using massively parallel
185 deep reinforcement learning. In *Conference on Robot Learning*, pages 91–100. PMLR, 2022.
- 186 [5] M. Liu, Z. Chen, X. Cheng, Y. Ji, R. Yang, and X. Wang. Visual whole-body control for legged
187 loco-manipulation. *arXiv preprint arXiv:2403.16967*, 2024.
- 188 [6] T. Portela, G. B. Margolis, Y. Ji, and P. Agrawal. Learning force control for legged manipulation.
189 *arXiv preprint arXiv:2405.01402*, 2024.