Multi-critic Learning for Whole-body End-effector Twist Tracking

Aravind Elanjimattathil Vijayan[*] [†], **Andrei Cramariuc** [†], **Mattia Risiglione** [†], **Christian Gehring** [§], **Marco Hutter** [†]
ETH Zurich[†], ANYbotics AG[§]

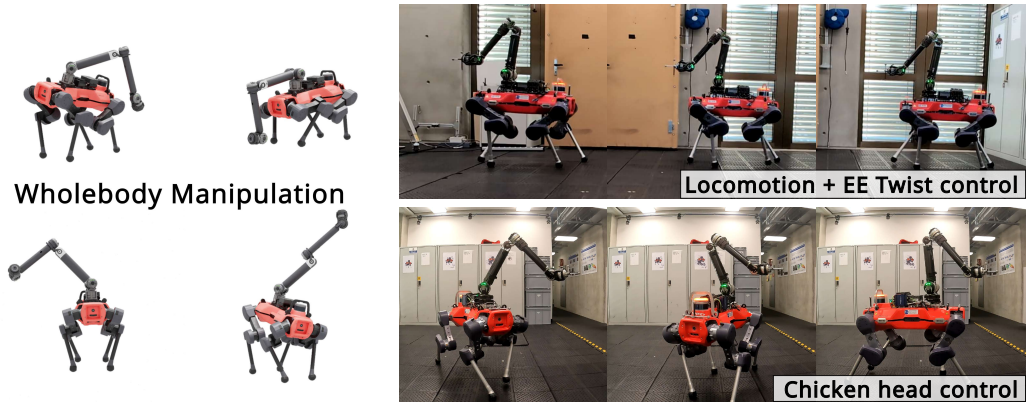

Figure 1: **Smooth and precise whole-body loco-manipulation.** We present a framework that enables quadrupedal robots to perform precise loco-manipulation through a multi-critic architecture and twist-based task space control. Our approach generates coordinated whole-body behaviors (Left) while maintaining accurate end-effector control during locomotion (Right).

Abstract: Learning whole-body control for locomotion and arm motions in a single policy has challenges, as the two tasks have conflicting goals. For instance, efficient locomotion typically favors a horizontal base orientation, while end-effector tracking may require the base to tilt to extend reachability. Additionally, current Reinforcement Learning (RL) approaches using a pose-based task specification lack the ability to directly control the end-effector velocity, making smoothly executing trajectories very challenging. To address these limitations, we propose an RL-based framework that allows for dynamic, velocity-aware whole-body end-effector control. Our method introduces a multi-critic actor architecture that decouples the reward signals for locomotion and manipulation, simplifying reward tuning and allowing the policy to resolve task conflicts more effectively. Furthermore, we design a twist-based end-effector task formulation that can track both discrete poses and motion trajectories. We validate our approach through a set of simulation and hardware experiments using a quadruped robot equipped with a robotic arm. The resulting controller can simultaneously walk and move its end-effector and shows emergent whole-body behaviors, where the base assists the arm in extending the workspace, despite a lack of explicit formulations. Videos and supplementary material can be found at multi-critic-locomanipulation.github.io.

Keywords: Loco-Manipulation, Multi-critic Reinforcement Learning, Whole-Body Control

[*] Correspondence to earavind@ethz.ch.

9th Conference on Robot Learning (CoRL 2025), Seoul, Korea.

1 Introduction

Robots need basic body control to perform manipulation tasks in various scenarios, such as industrial applications or service robotics. A robust low-level whole-body controller that can track body and end-effector poses and trajectories enables the seamless integration and execution of high-level tasks [1]. Multiple approaches to whole-body control exist, mainly divided between model-based control [2, 3, 4, 5], reinforcement learning (RL) based methods [6, 7, 1, 8, 9, 10, 11], and mixes of the two [12, 13, 14]. Model-based approaches have limitations and lack robustness due to the difficulty in accurately modeling interactions and the environment. RL has emerged as a staple framework for training locomotion controllers that can work in challenging environments [15, 16, 17, 18].

Whole-body control using RL faces several challenges. Firstly, locomotion and end-effector control have conflicting goals. Locomotion tends towards a horizontal body orientation for energy-efficient gaits [15, 16, 17], but tilting the base can increase the reachability of the arm. Similarly, velocity tracking of the base is far coarser than needed to achieve precise end-effector control. Similarly, the coarse velocity tracking of the base is insufficient for precise end-effector control, leading to low accuracy in unified policies [7, 1]. Solutions using separate controllers [12] or different policies [8] have limitations, such as requiring controller switching, not using the full workspace, or being unable to perform arm movements while in motion. A second challenge is formulating the end-effector tracking task. Current methods use goal pose tracking [13, 1, 10, 19, 8] or force tracking [11]. However, trajectory following using a goal pose tracking controller may not always lead to smooth end-effector control since the controller attempts to rigidly track each control point. This is a fundamental issue with such task formulation due to its lack of velocity specification.

To address the first challenge of diverging goals, we propose using a multi-critic actor architecture [20, 21] that decouples the reward signals for locomotion, manipulation, and contact scheduling into individual reward groups. This simplifies the reward tuning, as separate critics allow the policy to resolve task conflicts more effectively. Furthermore, having separate critics seamlessly combines dense tracking rewards with sparse contact schedule rewards. We show that this approach is easier to use and also has desirable emergent behaviors. As a second contribution, we design a 6D twist-based end-effector task formulation that enables direct control over the end-effector velocity while precisely tracking both discrete poses and continuous motion trajectories. Through comprehensive simulation and hardware experiments, we validate that the twist-based formulation leads to smooth end-effector tracking during stationary operation and during dynamic locomotion of the quadrupedal robot. In summary, our contributions are:

- A multi-critic approach that trains a joint whole-body control policy that can simultaneously walk and accurately control the end-effector motion.
- A twist-based formulation for whole-body end-effector tracking that allows for smoothly following any end-effector trajectory at varying velocities.
- Experiments on simulation and hardware to demonstrate the effectiveness of our approach.

2 Related Work

Approaches towards whole-body control can be split into two major categories: learning-based or model-based control. Model predictive controllers (MPCs) can both locomote and perform high-accuracy end-effector motions [2, 3, 4], while avoiding self-collisions [5]. Although these methods can be further robustified by extending the model [22] or improving the optimization framework [23, 24], model-based approaches remain vulnerable to unmodeled disturbances. Especially for locomotion, reinforcement learning (RL) has emerged as the best technique to create robust and versatile controllers [15, 16, 17, 18], using large-scale parallel simulation [25, 26].

Similarly, for end-effector control, several RL-based controllers have been proposed [1, 6, 7, 8, 9, 10]. Some works tackle arm and base control as separate policies, where the arm is trained to reach a pose target and the base follows a separate velocity command. These methods either use RL for both [13], or a mix of model-based and RL control [14, 12]. Such split hierarchical approaches have

limitations, as the arm cannot fully leverage the base policy to extend its reachability or improve accuracy by directly commanding the leg motors.

Conflicting goals also arise when training whole-body control policies. Locomotion rewards favor horizontal base orientation for energy-efficient gaits [15, 16, 17, 18], while end-effector accuracy requires a stiff, potentially tilted base to extend arm reach. Methods balancing this trade-off [7, 1] suffer from accuracy issues with tracking errors in tens of centimeters. Portela *et al.* [8] splits walking from end-effector motions but requires controller switching. In legged-wheeled systems, arm motions are more easily executed, as the presence of wheels reduces the necessity for base movements associated with walking gaits [9, 10]. Our proposed approach tackles this problem using a multi-critic approach [20, 21], enabling the locomotion and end-effector tracking problems to have separate sets of rewards, but one single policy. This emerging approach has been explored for style mixing [20] or locomotion [27, 28]. We showcase further uses and how multi-critic RL training simplifies the tuning problem and emergently leads to desired behaviors, benefiting from accuracy and synergy between differing tasks on hardware.

Another critical aspect is target formulation for arm movement. Early works used 3D position tracking [7, 6], evolving to 6D pose tracking with various representations [13, 1, 10, 19, 8]. However, pose-based approaches struggle with smooth trajectory tracking and require high-level monitoring [1, 9]. An alternative to pose tracking is force tracking, as demonstrated by Portela *et al.* [11], which allows the possibility of building an impedance controller on top. In contrast, our approach circumvents these issues by directly performing twist tracking, resulting in simpler reward tuning for the entire problem. Furthermore, the twist tracking formulation provides explicit control over end-effector velocity, enabling the robot to follow complex trajectories.

3 Method

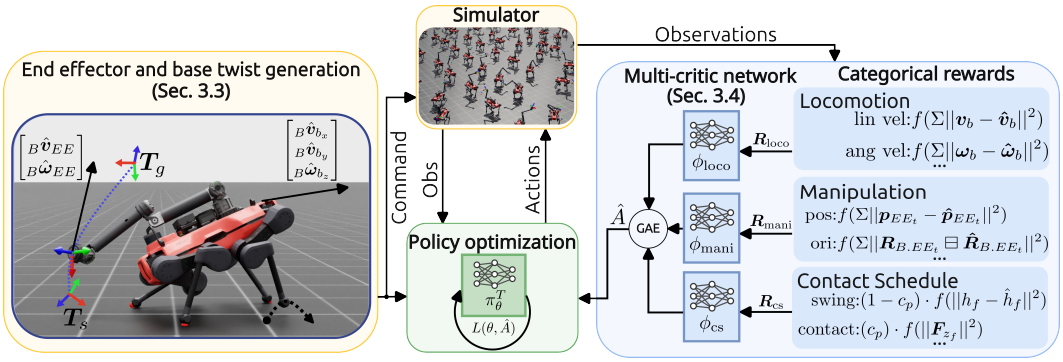

Figure 2: **Architecture for the teacher training pipeline**: Given a randomly sampled start and goal pose, the desired end-effector and base twist, along with the desired foot height, are provided as commands to the policy by the command generator. Rewards are categorically computed and consumed by separate critics, leading to individual value functions. The advantage is estimated per critic, normalized, and summed to compute the total advantage. Policy optimization is performed using Proximal Policy Optimization (PPO) to optimize the teacher policy.

We aim to train a controller that takes locomotion and end-effector task commands and outputs joint position residuals, enabling coordinated whole-body motion. More specifically, we want the ability to smoothly track 6D motion trajectories with explicit control over the end-effector velocity. Lastly, we want to ensure that torso motions do not affect the manipulation task and that the robot can perform the manipulation task while stationary and during walking.

3.1 Task definition

We formulate the task as a Partially Observable Markov Decision Process (POMDP) and train the policy using a teacher-student architecture (Fig. 2) that can follow a target command $\boldsymbol{c}_t$ defined as

$$\boldsymbol{c}_t = [{}_B\boldsymbol{v}_b,\ {}_B\boldsymbol{\omega}_b,\ {}_B\boldsymbol{v}_{EE},\ {}_B\boldsymbol{\omega}_{EE},\ {}_B\hat{\boldsymbol{p}}_G,\ \hat{\boldsymbol{R}}_{BG},\ \boldsymbol{h}_f], \tag{1}$$

where ${}_B\boldsymbol{v}_b$ and ${}_B\boldsymbol{\omega}_b$ are the base linear and angular velocity targets expressed in robot's base frame, ${}_B\boldsymbol{v}_{EE}$ and ${}_B\boldsymbol{\omega}_{EE}$ are the end-effector linear and angular velocity targets expressed in robot's base frame, ${}_B\hat{\boldsymbol{p}}_G$ and $\hat{\boldsymbol{R}}_{BG}$ define the final pose in the command trajectory expressed in robot's base frame, and $\boldsymbol{h}_f$ is the desired swing heights for each foot.

At each control time step t, the teacher policy π_θ^T generates an action $\boldsymbol{a}_t$ taking as input $\boldsymbol{s}_t = [\boldsymbol{h}_t,\ \boldsymbol{o}_t,\ \boldsymbol{p}_t,\ \boldsymbol{a}_{t-1},\ \boldsymbol{c}_t] \in \mathbb{R}^{187}$ consisting of the 4 step history of previous joint positions $\boldsymbol{h}_t \in \mathbb{R}^{72}$, a set of observations concerning the robot state $\boldsymbol{o}_t \in \mathbb{R}^{45}$, privilege information obtained from the simulator $\boldsymbol{p}_t \in \mathbb{R}^{32}$, the action at previous time step $\boldsymbol{a}_{t-1} \in \mathbb{R}^{18}$, and the command $\boldsymbol{c}_t \in \mathbb{R}^{20}$. For more details, please refer to the Appendix. We normalize the observations using a running buffer to ensure that the policy can learn a robust representation of the state space.

Given the observations, the policy π_θ^T generates an action $\boldsymbol{a}_t = [\boldsymbol{a}_t^{arm},\ \boldsymbol{a}_t^{robot}]$ where $\boldsymbol{a}_t^{arm} \in \mathbb{R}^6$ and $\boldsymbol{a}_t^{robot} \in \mathbb{R}^{12}$ are the relative offsets to be applied. Unlike the standard approach[8] of applying actions to predefined default joint positions, we apply these offsets directly to the current joint positions. Assuming a Gaussian action distribution, the actions are normalized with a fixed standard deviation σ_a and clamped to be within $[-1, 1]$. The joint position targets for the next time step $\boldsymbol{q}_{t+1}$ are computed as,

$$\boldsymbol{q}_{t+1} = \boldsymbol{q}_t + \text{clamp}((\boldsymbol{a}_t/\sigma_a),\ -1,\ 1) \tag{2}$$

3.2 Command Formulation

End-effector trajectory sampling: During training, we generate manipulation trajectories by initializing the start pose $\boldsymbol{T}_s$ at the current end-effector location, while the goal pose $\boldsymbol{T}_g$ is randomly sampled using a two-stage approach. First, we sample positions within a unit sphere centered at the robot's shoulder. Any samples falling within a predefined cuboid representing the robot's torso and hip region are rejected to ensure physically feasible goals, which enables smoother training [8]. For orientation determination, we construct a reference orientation where the x-axis aligns with gravity and the z-axis points along the vector from the torso center to the sampled position. This reference orientation is then perturbed with random rotations about each axis, bounded within $\frac{\pi}{4}$ radians. This approach ensures that a wide range of the sampled poses are within the robot's feasible workspace. Given the start and goal pose, we implement a temporal trajectory formulation that prioritizes continuous tracking rather than terminal goal achievement. We assign a random duration to each trajectory and generate intermediate waypoints to be tracked using linear interpolation in position and spherical linear interpolation in orientation. The intermediate goals are defined as

$$\begin{aligned} \boldsymbol{r}_i &= \boldsymbol{r}_s + \alpha \cdot (\boldsymbol{r}_g - \boldsymbol{r}_s) \quad \boldsymbol{r} \in \mathbb{R}^3 \\ \boldsymbol{\theta}_i &= \boldsymbol{\theta}_s \boxplus \alpha \cdot (\boldsymbol{\theta}_g \boxminus \boldsymbol{\theta}_s) \quad \boldsymbol{\theta} \in SO(3) \end{aligned} \tag{3}$$

where $\boldsymbol{r}_i$ and $\boldsymbol{\theta}_i$ represent the intermediate position and orientation, $\boldsymbol{r}_s$ and $\boldsymbol{\theta}_s$ represent the start position and orientation, and $\boldsymbol{r}_g$ and $\boldsymbol{\theta}_g$ represent the final goal position and orientation, and α is the interpolation factor. The operators $\boxplus$ and $\boxminus$ denote addition and subtraction in the Lie group $SO(3)$.

End-effector command: While most pose tracking methods in the literature [13, 1, 10, 19, 8] define the task to track a final goal pose within a given time, our task formulation is a continuous trajectory tracking problem with explicit velocity control. Hierarchical approaches, which send intermediate goal poses to a pose-based tracking controller to achieve trajectories, lack critical information about the required end-effector velocity. This deficiency leads to jerky end-effector motions as the policy attempts to rigidly maintain each intermediate pose without smoothly transitioning between them. To address these limitations, we introduce an end-effector twist-based command formulation that explicitly incorporates velocity information. More specifically, we define the command as:

$$\boldsymbol{v}_{EE} = \frac{1}{\Delta t}(\boldsymbol{r}_i - \boldsymbol{r}_{EE}) \ \text{ and } \ \boldsymbol{\omega}_{EE} = \frac{1}{\Delta t}(\boldsymbol{\theta}_i \boxminus \boldsymbol{\theta}_{EE}) \tag{4}$$

where, $\boldsymbol{r}_{EE}$ and $\boldsymbol{\theta}_{EE}$ are the current end-effector position and orientation respectively, and Δt is the control time step. Finally, the end-effector command to the policy is defined as the stacked vector of $\boldsymbol{v}_{EE}$, $\boldsymbol{\omega}_{EE}$, and the final goal pose $\boldsymbol{T}_g$ along the entire trajectory.

We observe that training exclusively with global feed-forward trajectories as defined above can lead to a sparse distribution of positive tracking examples, leading to extended training times before the policy grasps the task. To mitigate this, we define a subset of the trajectories locally. A local trajectory is created by re-initializing the start pose of the trajectory at every control time step. This approach ensures that the policy frequently encounters states near the desired trajectory, providing a denser representation of the reward landscape and accelerating the learning process. Similarly to [8], we represent the end-effector trajectory in a robot-centric task frame that is agnostic to the torso height, pitch, and roll orientation. This choice decouples the task from the torso motion, allowing the policy to learn a coordinated whole-body motion to satisfy a precise end-effector motion.

Foot trajectory generation: computes the desired foot heights above the ground during locomotion. A gait phase ϕ and foot phase offsets $\boldsymbol{\theta} = [\theta_{LF},\ \theta_{RF},\ \theta_{LH},\ \theta_{RH}]$ are used to generate a sinusoidal swing foot trajectory pattern [14], where the subscripts denote the four legs: Left Front (LF), Right Front (RF), Left Hind (LH), and Right Hind (RH) . The height of the feet is computed as:

$$\begin{aligned} \boldsymbol{h}_f &= h_{max}[\sin(2\pi\theta_{LF}),\ \sin(2\pi\theta_{RF}),\ \sin(2\pi\theta_{LH}),\ \sin(2\pi\theta_{RH})] \\ \boldsymbol{\theta} &= [\phi + \theta_{LF},\ \phi + \theta_{RF},\ \phi + \theta_{LH},\ \phi + \theta_{RH}] \end{aligned} \tag{5}$$

During an episode, the gait phase is updated according to a fixed gait frequency, and the foot phase offsets are fixed to correspond to a static gait pattern.

The base command: consists of the linear velocity of the torso ${}_B\boldsymbol{v}_b = [{}_B\boldsymbol{v}_{b_x}, {}_B\,\boldsymbol{v}_{b_y}]$ and the angular velocity ${}_B\boldsymbol{\omega}_b$ along the Z-axis wrt. the base frame. During training, this velocity command is periodically resampled during the episode from a uniform distribution within the range $[-0.25, 0.25]$.

3.3 Multi-critic Actor Learning

Multi-critic actor learning has previously been shown to be effective in multi-task RL problems [20] and to effectively manage mixtures of dense and sparse rewards [27]. Fu *et al.* [7] introduced two critic networks and used curriculum learning to gradually mix two advantage functions to combine the learning of arm and leg actions in a unified policy. In this work, we categorize rewards into three distinct groups: locomotion, manipulation, and foot contact schedule. For each group, we define a dedicated critic ϕ_i that estimates the corresponding value function $V_{\phi_i}(.)$. The overall advantage $\hat{A}$ is calculated as the sum of the normalized advantage estimates $\hat{A}_i$ from each critic. Following the PPO framework, we compute the policy surrogate loss as

$$L(\theta) = \hat{\mathbb{E}}[min(r_t(\theta)\hat{A},\ clip(r_t(\theta),\ 1-\epsilon,\ 1+\epsilon)\hat{A})]$$

where $r_t(\theta)$ is the ratio of new to the old policy probabilities, ϵ is the PPO clipping parameter, and $\hat{\mathbb{E}}$ is the empirical average over the batch. The three reward groups are structured as follows:

- **Locomotion**: base velocity tracking rewards and regularization terms.
- **Manipulation**: end-effector pose tracking rewards and regularization terms.
- **Contact schedule**: rewards and regularization terms for tracking foot contact states.

Notably, we introduced the contact schedule as a separate group due to the low-frequency nature of gait patterns, which results in a relatively sparse reward distribution compared to the other groups.

4 Results

We deploy our controller on the ANYmal D robot [29] from ANYbotics AG mounted with a Dynaarm manipulator from Duatic AG. The robot has a total of 18 actuators, of which 12 actuate the legs. While ANYmal's motion controller is executed at 400 Hz, our control policy is executed with a decimation rate of 8:1, resulting in 50 Hz on the onboard computer [8, 17, 30].

Tracking performance of different trajectory types on hardware: To assess the tracking performance of the policy, we evaluate its ability to track the following trajectory types:

- **Straight line**: The policy is commanded to move 30 cm along each axis.

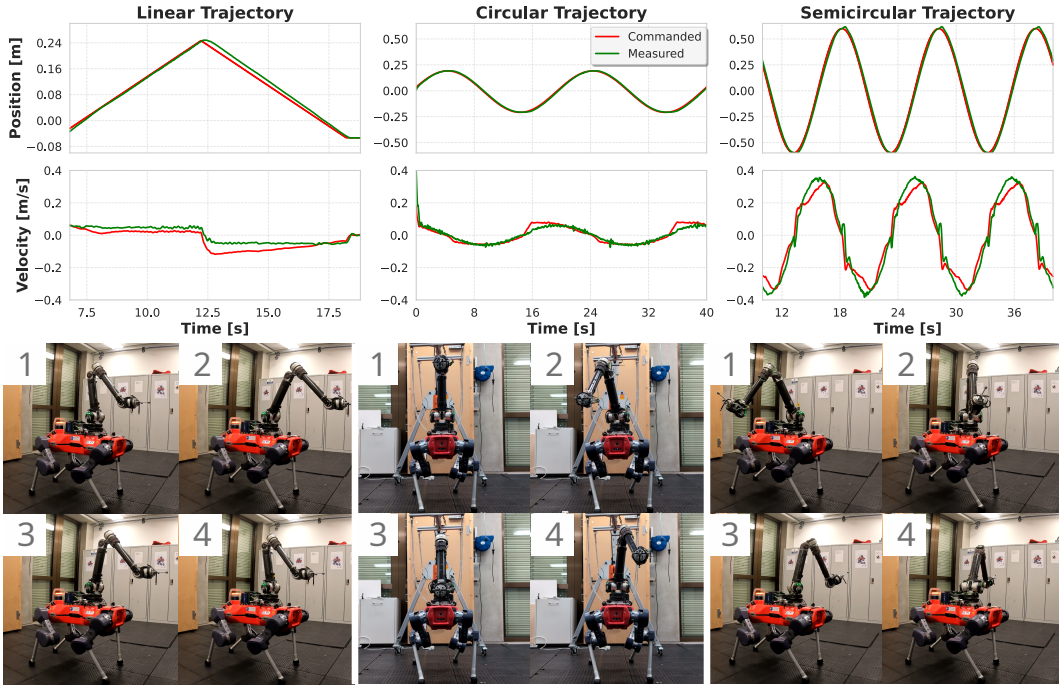

Figure 3: **End-Effector Tracking Performance for Different Trajectories**: Desired and measured position and velocity for tracking linear, circular, and semicircular trajectories (left to right).

- **Circle**: The policy is commanded to move in a circle on the YZ plane with a radius of 20 cm at a fixed distance in front of the robot.
- **Semicircle around robot**: To evaluate trajectories over an extended workspace, the policy is commanded a trajectory of radius 60 cm radius around the robot.

During these experiments, the end effector orientation is actively controlled to point outwards along the vector from the torso to the end effector while aligning the X-axis of the end-effector frame with gravity vector. In Fig. 3, we show the tracking performance for all three trajectory types.

Tracking trajectories with varying velocities on hardware:

Motion	$\hat{\boldsymbol{v}}_{EE}$	$\delta \boldsymbol{r}$	$\delta \dot{\boldsymbol{r}}$
Linear	0.05	0.0166	0.0519
	0.1	0.0175	0.0623
Circular	0.05	0.0220	0.0609
	0.1	0.0274	0.0898
	0.2	0.0348	0.1781
Semicircular	0.1	0.0480	0.2339
	0.2	0.0732	0.3590

Table 1: Tracking performance in end-effector position error (L2 norm) $\delta \boldsymbol{r}$ (m) and velocity error $\delta \dot{\boldsymbol{r}}$ (m/s) for robot linear motions XYZ executed on hardware at specified distances and speeds $\hat{\boldsymbol{v}}_{EE}$ (m/s).

To demonstrate the capability of the controller to execute trajectories at varying velocities, we execute the three trajectory types described above at different commanded velocities. The RMSE end-effector tracking error in position and velocity while tracking trajectories at different speeds is listed in Table 1. As also shown in Fig. 3, the controller achieves good velocity tracking across different velocity ranges. Although the accuracy is lower at slower speeds, particularly for linear trajectories, the policy maintains low position errors throughout the execution. It must be noted that circular and semicircular trajectories being evaluated are not part of the training distribution, since the trajectories in training are always computed between start and goal pose through linear interpolation.

Effect of multi-critic architecture on gait tracking in simulation: Multi-critic architectures have previously demonstrated effectiveness in multi-task RL problems. While the agent is trained only with a static walking gait pattern, we observe that the policy generalizes to perform a trotting gait at runtime by simply modifying the commanded foot swing trajectories. The measured foot trajectories for both static walk and trotting gaits are shown in Fig. 4. This result demonstrates that separating the contact schedule into a separate critic enables the policy to learn a more general representation

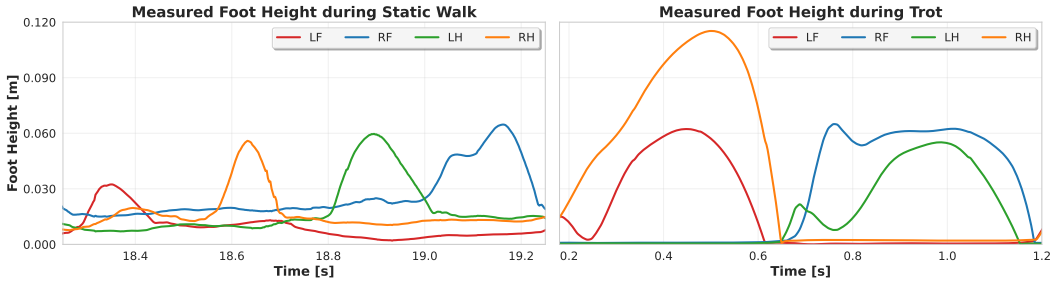

Figure 4: **Contact Schedule Adaptation to Unseen Gait Patterns**: Measured height of Left Fore (LF), Right Fore (RF), Left Hind (LH) and Right Hind (RH) feet for (a) static walking measured on hardware, and (b) trot gait measured in simulation.

of gait patterns, allowing it to adapt foot timings based on the input command. We note that such generalization has not been previously reported in the literature, highlighting the efficacy of our multi-critic architecture.

Comparison with existing methods in simulation: To evaluate our proposed approach against existing methods, we conduct comparative experiments using a semicircular end-effector trajectory, both while the robot is stationary and during locomotion. We integrate our arm control with three different locomotion policies: Ma [12], Taka 3-DoF [17], and Taka 6-DoF [30].

State	Controller	$\delta \boldsymbol{r}$ [m]	$\delta \boldsymbol{\theta}$ [°]
Standing	Proposed	**0.0176**	1.815
	Taka 3-DoF [17]	0.0580	4.196
	Taka 6-DoF [30]	0.0439	3.608
	Ma [12]	0.0386	3.515
	Portela [8]	0.0391	**1.756**
Walking	Proposed	**0.0358**	**2.872**
	Taka 3-DoF [17]	0.4180	4.711
	Taka 6-DoF [30]	0.0436	4.974
	Ma [12]	0.0483	6.432
	Portela [8]	-	-

Table 2: Average end-effector tracking errors: $\delta \boldsymbol{r}$ (mean Euclidian norm) for position and $\delta \boldsymbol{\theta}$ (mean angular error) for orientation in simulation.

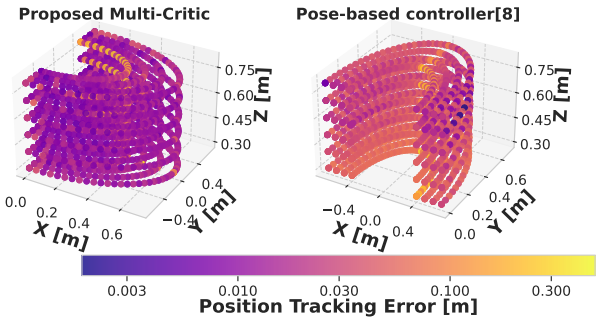

Figure 5: Workspace sweep trajectory wrt. base used for evaluating tracking performance in simulation.

To achieve this, we apply the leg actions generated by the respective locomotion policy while executing the arm actions produced by our proposed approach. We compute the tracking end effector tracking errors using the measurements obtained from the state estimator. As shown in Table 2, while all combinations achieve satisfactory end-effector tracking, our proposed whole-body locomotion policy demonstrates superior performance with the lowest tracking errors.

For comparison with existing pose-based controllers, we evaluate against the whole-body policy from Portela et al. [8]. Since their policy is not designed for locomotion, we present results only for stationary experiments. Our approach achieves significantly better task space trajectory tracking in position while maintaining comparable orientation accuracy. Notably, stationary policies typically have a substantial advantage during training due to enhanced stability from continuous leg contact. This makes our results particularly significant, as our proposed approach delivers superior performance while being able to perform loco-manipulation.

In addition to the linear and circular trajectories introduced above, we introduce a workspace sweep trajectory that extends the semicircular trajectory by sweeping across various radii and heights. As in Fig. 5, our proposed approach demonstrates improved end-effector tracking performance across the evaluated workspace compared to a pose-based alternative. In Table 3, we present an extended comparison of end-effector tracking performance for linear, circular, and workspace sweep trajectories, and provide results also against a whole-body MPC controller [2]. Our proposed approach consistently outperforms all other methods across all trajectory types and both standing and walking scenarios, achieving the lowest position and orientation errors as well as velocity tracking errors, demonstrating whole-body loco-manipulation capabilities.

Type	Controller	Standing				Walking			
		$\delta \boldsymbol{r}$ [m]	$\delta \boldsymbol{\theta}$ [°]	$\delta \dot{\boldsymbol{r}}$ [m/s]	$\delta \dot{\boldsymbol{\theta}}$ [rad/s]	$\delta \boldsymbol{r}$ [m]	$\delta \boldsymbol{\theta}$ [°]	$\delta \dot{\boldsymbol{r}}$ [m/s]	$\delta \dot{\boldsymbol{\theta}}$ [rad/s]
Linear	**Multi-critic (Ours)**	**0.0095**	**1.6084**	**0.0568**	**0.1756**	**0.0093**	**1.9926**	**0.0559**	**0.2176**
	Taka 3-DoF [17]	0.0474	10.1269	0.2934	1.1056	0.0408	16.5597	0.2514	1.8080
	Taka 6-DoF [30]	0.0271	8.2354	0.1648	0.8983	0.0370	9.9586	0.2285	1.0868
	Ma [12]	0.0232	8.0863	0.1411	0.8823	0.0573	14.2652	0.3562	1.5563
	Portela [8]	0.0240	1.7285	0.1421	0.7562	-	-	-	-
	Whole-body MPC	0.0207	3.9522	0.1291	0.4311	0.0195	2.8417	0.1221	0.3100
Circular	**Multi-critic (Ours)**	**0.0080**	**0.9338**	**0.0506**	**0.1018**	**0.0137**	**1.5921**	**0.0865**	**0.1736**
	Taka 3-DoF [17]	0.0446	12.3341	0.2774	1.3449	0.0259	11.8966	0.1611	1.2986
	Taka 6-DoF [30]	0.0234	9.7642	0.1447	1.0658	0.0460	40.1593	0.2865	4.3794
	Ma [12]	0.0269	36.6286	0.1663	3.9952	0.0425	7.8882	0.2653	0.8600
	Portela [8]	0.0399	2.7328	0.2358	1.1981	-	-	-	-
	Whole-body MPC	0.0239	3.2007	0.1497	0.3491	0.0327	4.2238	0.2044	0.4607
Workspace sweep	**Multi-critic (Ours)**	**0.0233**	**3.9257**	**0.1477**	**0.4323**	**0.0408**	**8.1143**	**0.2520**	**0.8876**
	Taka 3-DoF [17]	0.1415	13.6202	0.8812	1.4878	0.0858	9.2738	0.5329	1.0137
	Taka 6-DoF [30]	0.0821	7.6319	0.5094	0.5094	0.0900	9.8273	0.5591	1.0736
	Ma [12]	0.0906	8.6375	0.5626	0.9454	0.1514	16.8287	0.9439	1.8366
	Portela [8]	0.0486	2.1633	0.2804	0.9737	-	-	-	-
	Whole-body MPC	0.3244	38.2293	2.0277	4.1702	0.3346	39.5757	2.0915	4.3170

Table 3: Extended comparison with trajectory and velocity tracking errors in simulation.

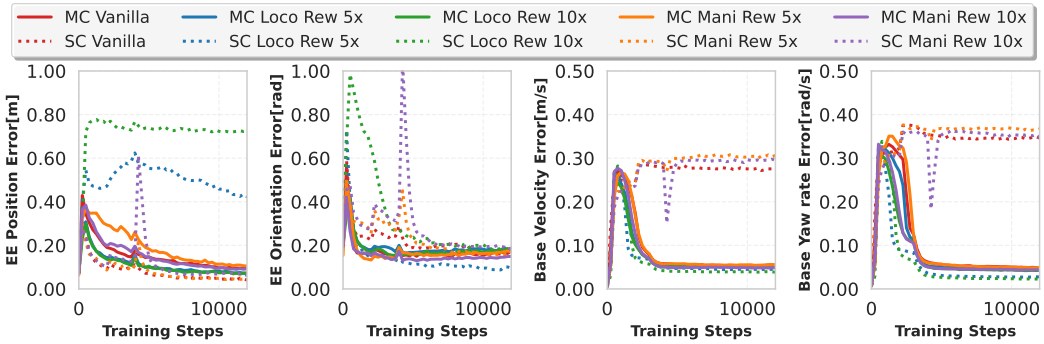

Figure 6: **Reward sensitivity analysis of single-critic vs multi-critic frameworks**: Multi-critic policy is robust against reward scaling across categories whereas the single-critic policy only learns to perform the objective with higher reward scale.

Reward Sensitivity Analysis - Single-Critic vs Multi-Critic Learning: Since our approach splits rewards categorically into locomotion, manipulation, etc., we multiply all rewards within each category by a fixed multiplicative factor and evaluate the resulting policy performance for reward sensitivity analysis. Specifically, we conduct experiments scaling locomotion and manipulation rewards by factors of 5 and 10. As shown in Fig. 6, the multi-critic approach remains insensitive to these weighting factors, while the single-critic approach only learns to perform either locomotion or manipulation and not to perform both simultaneously. When scaling the locomotion reward in the single-critic approach, we observe improved locomotion performance but deteriorated end-effector tracking. In contrast, scaling the manipulation reward improves the end-effector tracking performance but degrades locomotion quality. These experiments demonstrate that the proposed multi-critic approach is robust to variations in relative weightage between reward categories, making it effective for combining multiple learning objectives without careful reward tuning.

5 Conclusion

In this work, we presented a whole-body loco-manipulation framework for quadrupedal robots that enables continuous and smooth trajectory tracking while stationary and during locomotion. Through extensive simulations and hardware experiments, we demonstrated the effectiveness of our twist-based command formulation in achieving precise end-effector velocity control. Our novel multi-critic architecture successfully combines locomotion, manipulation, and contact scheduling without requiring complex training curricula or reward tuning. Comparative evaluations against existing non-whole-body and pose-based controllers validate the superior performance of our approach.

6 Limitations

The current implementation has two main limitations. First, locomotion performance is restricted to moderately rough terrains, as robust rough-terrain controllers were not the focus of this work. In future work, we plan to address this by including reward formulations and perception mechanisms from existing rough-terrain locomotion frameworks. Our multi-critic formulation should greatly simplify the tuning involved in incorporating these new terms. However, having to train the whole-body teacher to incorporate both robust arm maneuvers and perceptive locomotion will significantly increase training times and make later tuning of individual behaviors more difficult. The second limitation is that during whole-body maneuvers, the robot occasionally reaches the limits of the shoulder joint, triggering safety mechanisms. In the future, this can be addressed by incorporating joint limits during policy training, either as penalties, terminations, or, for example, through the use of constrained PPO.

Acknowledgments

We thank the reviewers of CoRL 2025 for their constructive and insightful feedback. This research was supported by ANYbotics AG and the Swiss National Science Foundation through the National Centre of Competence in Automation (NCCR automation). This project also received funding under the European Union's Horizon Europe Framework Programme under grant agreement No 101121321.

References

[1] H. Ha, Y. Gao, Z. Fu, J. Tan, and S. Song. UMI on Legs: Making Manipulation Policies Mobile with Manipulation-Centric Whole-body Controllers. In *Conference on Robot Learning (CoRL)*, 2024.

[2] J.-P. Sleiman, F. Farshidian, M. V. Minniti, and M. Hutter. A Unified MPC Framework for Whole-Body Dynamic Locomotion and Manipulation. *IEEE Robotics and Automation Letters (RA-L)*, 6(3):4688–4695, 2021.

[3] M. Mittal, D. Hoeller, F. Farshidian, M. Hutter, and A. Garg. Articulated Object Interaction in Unknown Scenes with Whole-Body Mobile Manipulation. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1647–1654, 2022.

[4] S. Zimmermann, R. Poranne, and S. Coros. Go Fetch! - Dynamic Grasps using Boston Dynamics Spot with External Robotic Arm. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 4488–4494, 2021.

[5] J.-R. Chiu, J.-P. Sleiman, M. Mittal, F. Farshidian, and M. Hutter. A Collision-Free MPC for Whole-Body Dynamic Locomotion and Manipulation, 2022.

[6] P. Arm, M. Mittal, H. Kolvenbach, and M. Hutter. Pedipulate: Enabling Manipulation Skills using a Quadruped Robot's Leg. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 5717–5723, 2024.

[7] Z. Fu, X. Cheng, and D. Pathak. Deep Whole-Body Control: Learning a Unified Policy for Manipulation and Locomotion. In *Conference on Robot Learning (CoRL)*, pages 138–149, 2023.

[8] T. Portela, A. Cramariuc, M. Mittal, and M. Hutter. Whole-body end-effector pose tracking. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2025.

[9] Z. Wang, Y. Jia, L. Shi, H. Wang, H. Zhao, X. Li, J. Zhou, J. Ma, and G. Zhou. Arm-Constrained Curriculum Learning for Loco-Manipulation of a Wheel-Legged Robot. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 10770–10776, 2024.

[10] K. Jiang, Z. Fu, J. Guo, W. Zhang, and H. Chen. Learning Whole-Body Loco-Manipulation for Omni-Directional Task Space Pose Tracking With a Wheeled-Quadrupedal-Manipulator. *IEEE Robotics and Automation Letters (RA-L)*, 10(2):1481–1488, 2025.

[11] T. Portela, G. B. Margolis, Y. Ji, and P. Agrawal. Learning Force Control for Legged Manipulation. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 15366–15372, 2024.

[12] Y. Ma, F. Farshidian, T. Miki, J. Lee, and M. Hutter. Combining Learning-Based Locomotion Policy With Model-Based Manipulation for Legged Mobile Manipulators. *IEEE Robotics and Automation Letters (RA-L)*, 7(2):2377–2384, 2022.

[13] M. Liu, Z. Chen, X. Cheng, Y. Ji, R. Yang, and X. Wang. Visual Whole-Body Control for Legged Loco-Manipulation. In *Conference on Robot Learning (CoRL)*, 2024.

[14] G. Pan, Q. Ben, Z. Yuan, G. Jiang, Y. Ji, S. Li, J. Pang, H. Liu, and H. Xu. RoboDuet: Whole-body Legged Loco-Manipulation with Cross-Embodiment Deployment, 2024.

[15] J. Hwangbo, J. Lee, A. Dosovitskiy, D. Bellicoso, V. Tsounis, V. Koltun, and M. Hutter. Learning agile and dynamic motor skills for legged robots. *Science Robotics*, 4(26):eaau5872, 2019.

[16] J. Lee, J. Hwangbo, L. Wellhausen, V. Koltun, and M. Hutter. Learning quadrupedal locomotion over challenging terrain. *Science Robotics*, 5(47):eabc5986, 2020.

[17] T. Miki, J. Lee, J. Hwangbo, L. Wellhausen, V. Koltun, and M. Hutter. Learning robust perceptive locomotion for quadrupedal robots in the wild. *Science Robotics*, 7(62), 2022. ISSN 2470-9476.

[18] D. Hoeller, N. Rudin, D. Sako, and M. Hutter. ANYmal parkour: Learning agile navigation for quadrupedal robots. *Science Robotics*, 9(88):eadi7566, 2024.

[19] Y. Zhou, C. Barnes, J. Lu, J. Yang, and H. Li. On the Continuity of Rotation Representations in Neural Networks. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5745–5753, 2019.

[20] S. Mysore, G. Cheng, Y. Zhao, K. Saenko, and M. Wu. Multi-Critic Actor Learning: Teaching RL Policies to Act with Style . In *International Conference on Learning Representations (ICLR)*, 2022.

[21] G. Cheng, L. Dong, W. Cai, and C. Sun. Multi-Task Reinforcement Learning With Attention-Based Mixture of Experts. *IEEE Robotics and Automation Letters (RA-L)*, 8(6):3812–3819, 2023.

[22] A. Carron, E. Arcari, M. Wermelinger, L. Hewing, M. Hutter, and M. N. Zeilinger. Data-driven model predictive control for trajectory tracking with a robotic arm. *IEEE Robotics and Automation Letters (RA-L)*, 4(4):3758–3765, 2019.

[23] H. Ferrolho, W. Merkt, V. Ivan, W. Wolfslag, and S. Vijayakumar. Optimizing Dynamic Trajectories for Robustness to Disturbances Using Polytopic Projections. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 7477–7484, 2020.

[24] H. Ferrolho, V. Ivan, W. Merkt, I. Havoutis, and S. Vijayakumar. RoLoMa: Robust loco-manipulation for quadruped robots with arms. *Autonomous Robots*, 47(8):1463–1481, 2023.

[25] N. Rudin, D. Hoeller, P. Reist, and M. Hutter. Learning to Walk in Minutes Using Massively Parallel Deep Reinforcement Learning. In *Conference on Robot Learning (CoRL)*, pages 91–100, 2022.

[26] M. Mittal, C. Yu, Q. Yu, J. Liu, N. Rudin, D. Hoeller, J. L. Yuan, R. Singh, Y. Guo, H. Mazhar, A. Mandlekar, B. Babich, G. State, M. Hutter, and A. Garg. Orbit: A Unified Simulation Framework for Interactive Robot Learning Environments. *IEEE Robotics and Automation Letters (RA-L)*, 8(6):3740–3747, 2023. ISSN 2377-3774.

[27] F. Zargarbashi, J. Cheng, D. Kang, R. Sumner, and S. Coros. RobotKeyframing: Learning Locomotion with High-Level Objectives via Mixture of Dense and Sparse Rewards. *arXiv preprint arXiv:2407.11562*, 2024.

[28] J. Lee, L. Schroth, V. Klemm, M. Bjelonic, A. Reske, and M. Hutter. Exploring Constrained Reinforcement Learning Algorithms for Quadrupedal Locomotion. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 11132–11138, 2024.

[29] M. Hutter, C. Gehring, D. Jud, A. Lauber, C. D. Bellicoso, V. Tsounis, J. Hwangbo, K. Bodie, P. Fankhauser, M. Bloesch, et al. Anymal-a highly mobile and dynamic quadrupedal robot. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 38–44. IEEE, 2016.

[30] T. Miki, J. Lee, L. Wellhausen, and M. Hutter. Learning to walk in confined spaces using 3d representation. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 8649–8656, 2024. doi:10.1109/ICRA57147.2024.10610271.

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

7 Appendix

7.1 Training Details

7.1.1 Observation Space

The observation vector $\boldsymbol{s}_t \in \mathbb{R}^{187}$ consists of five main components: joint position history, proprioceptive feedback, privileged information, previous actions, and command signals. Table 4 summarizes each observation component. The student policy is trained without any privileged information

Observation Group	**Observation Name**	**Dimension**	**Group Dim**
History $\boldsymbol{h}_t$	Joint position history	72	72
Proprioception $\boldsymbol{o}_t$	Projected gravity	3	45
	Base linear velocity	3	
	Base angular velocity	3	
	Joint position	18	
	Joint velocity	18	
Privileged info $\boldsymbol{p}_t$	Feet contact state	4	32
	Static friction	4	
	Feet air time	4	
	Base external wrench	6	
	Base external push velocity	6	
	Base mass disturbance	1	
	End-effector external wrench	6	
	End-effector mass disturbance	1	
Previous action $\boldsymbol{a}_{t-1}$	Previous actions	18	18
Commands $\boldsymbol{c}_t$	Desired Base Velocity	3	20
	Desired End-Effector twist command	6	
	Desired End-Effector final goal pose	7	
	Desired Feet swing heights	4	

Table 4: Observation space for the teacher policy.

Observation Group	**Observation Name**	**Noise range**
History	Joint position history	$[-0.01, 0.01]$
Proprioception	Projected gravity	$[-0.01, 0.01]$
	Base linear velocity	$[-0.01, 0.01]$
	Base angular velocity	$[-0.1, 0.1]$
	Joint position	$[-0.01, 0.01]$
	Joint velocity	$[-0.2, 0.2]$
Previous actions	Previous actions	-
Commands	Desired Base Velocity	-
	Desired EE twist	-
	Desired Feet swing heights	-
	Desired Feet swing heights	-

Table 5: Observation space for the student policy and the range of noise applied during teacher-student distillation.

such that it can be deployed on hardware. During teacher-student distillation, uniform noise is added to the observations that are passed to the student policy. Table 5 summarizes the noise added to each observation group.

7.1.2 Rewards

The rewards are categorized into three groups: locomotion, manipulation, and contact schedule. For most quantities, we employ a Gaussian tracking reward function $\Phi(\boldsymbol{v}, \sigma^2) = \exp(-\boldsymbol{v}^T\boldsymbol{v}/\sigma^2)$. Table 6 summarizes the reward functions, while Table 7 describes the symbols used in these functions. We observe that multi-critic learning requires significantly less reward tuning compared to single-critic approaches. Unlike previous works [7, 27], our method avoids weighted averaging during advantage estimation, which further reduces the number of hyperparameters that require tuning.

Group	Reward Name	Reward Function	Weight
Loco	Base linear velocity	$\Phi(\hat{\boldsymbol{v}}_{b_{x,y}} - \boldsymbol{v}_{b_{x,y}}, 0.1)$	2.0
	Base angular velocity	$\Phi(\hat{\boldsymbol{\omega}}_{b_z} - \boldsymbol{\omega}_{b_z}, 0.05)$	2.0
	Torso height	$\Phi(\hat{\boldsymbol{h}}_{b_z} - \boldsymbol{h}_{b_z}, 0.1)$	0.5
	Base roll pitch angles	$\Phi(\boldsymbol{\theta}_{b_{xy}}, 0.1)$	0.1
	Torso linear velocity	$\Phi(\boldsymbol{v}_{b_z}, 0.2)$	0.5
	Torso roll pitch velocities	$\Phi(\boldsymbol{\omega}_{b_{xy}}, 0.2)$	2.5
	Is alive	$!z_{terminated}$	0.05
	Is terminated	$z_{terminated}$	-400.0
	Undesired robot contacts	$n_{contacts,robot}$	-1.0
	Robot action rate	$\Phi(\boldsymbol{a}_{t,robot} - \boldsymbol{a}_{t-1,robot}, 0.1)$	0.001
	Robot joint torque	$\Phi(\boldsymbol{\tau}_{t,robot}, 40.0)$	0.00001
	Robot joint velocity	$\Phi(\dot{\boldsymbol{q}}_{t,robot}, 4.0)$	0.0001
Mani	End-Effector position	$\Phi(r_{EE_t} - (r_{EE_{t-1}} + \hat{v}_{EE} \cdot \Delta t), 0.005)$	5.0
	End-Effector orientation	$\Phi(\boldsymbol{R}_{EE_t} \boxminus (\boldsymbol{R}_{EE_{t-1}} \boxplus \hat{\omega}_{EE} \cdot \Delta t), 0.01)$	4.0
	Undesired arm contacts	$n_{contacts,arm}$	-1.0
	Arm action rate	$\Phi(\boldsymbol{a}_{t,robot} - \boldsymbol{a}_{t-1,robot}, 0.5)$	0.1
	Arm joint torque	$\Phi(\boldsymbol{\tau}_{t,robot}, 40.0)$	0.00001
	Arm joint velocity	$\Phi(\dot{\boldsymbol{q}}_{t,robot}, 4.0)$	0.0001
CS	Feet Contact	$\Sigma_f(1 - C_f) \cdot \Phi(F_f, 1.0) \cdot \Phi(\hat{h_z} - h_z, 0.05) + \Sigma_f C_f \cdot n_{F_{f_z}>1.0} \cdot \Phi(v_{f_{xy}}, 0.01)$	1.0
	Feet air time variance	$\text{Var}(t_{air_{t-2..t}}) + \text{Var}(t_{contact_{t-2..t}})$	1.0
	Feet air time	$\Sigma_{i=1}^{4} n_{contacts,feet} \cdot t_{air_i}$	0.25

Table 6: Reward functions for training the teacher policy categorized into three groups, Loco (locomotion), Mani(manipulation and CS (contact schedule).

7.1.3 Training Setup

We train the policy by simulating 4096 parallel agents using IsaacLab [26], a GPU-accelerated simulation framework. The training process incorporates a curriculum approach where agents begin on flat terrain and progressively advance to more challenging, moderately rough terrains as their performance improves. To facilitate effective zero-shot sim-to-real transfer, we implement extensive domain randomization across multiple physical parameters, as comprehensively documented in Table 8. We observe that randomization of foot friction and the application of external disturbances to the torso contribute to the development of stable walking behaviors and enhance the robustness of the policy during rapid arm movements. The external torso push velocity components are implemented by augmenting the measured torso velocity with random values for variable durations during simulation. This perturbation strategy teaches the policy to maintain and recover balance when faced with unexpected environmental disturbances.

The teacher policy architecture comprises an actor MLP network with 3 hidden layers ($[512, 256, 128]$ units) and ReLU activations. The multi-critic implementation consists of 3 critic MLP networks with identical dimensions. The policy is trained using PPO [31]. During both teacher and student policy trainings, the control policy operates at 50 Hz. While the control policy runs at

Symbol	Description
$\hat{\boldsymbol{v}}_{b_{x,y}}$	Desired base linear velocity in x and y direction
$\hat{\boldsymbol{\omega}}_{b_z}$	Desired base angular velocity in z direction
$\hat{\boldsymbol{h}}_{b_z}$	Desired torso height
$\boldsymbol{\theta}_{b_{xy}}$	Base roll and pitch angles
$\boldsymbol{v}_{b_z}$	Torso linear velocity in z direction
$\boldsymbol{\omega}_{b_{xy}}$	Torso roll and pitch velocities
$z_{terminated}$	Termination signal
$n_{contacts,robot}$	Number of undesired robot contacts
$n_{contacts,arm}$	Number of undesired arm contacts
$\boldsymbol{a}_{t,robot}$	Robot action at time t
$\boldsymbol{\tau}_{t,robot}$	Robot joint torques at time t
$\dot{\boldsymbol{q}}_{t,robot}$	Robot joint velocities at time t
r_{EE_t}	End-effector position at time t
$\boldsymbol{R}_{EE_t}$	End-effector orientation at time t
$\hat{v}_{EE}$	Desired end-effector velocity
$\hat{\omega}_{EE}$	End-effector angular velocity
C_f	Feet contact probability
F_f	Feet contact force
h_z	Feet height
$v_{f_{xy}}$	Feet velocity in x and y direction
t_{air_i}	Feet air time
$t_{contact_i}$	Feet contact time
$n_{F_{f_z}>1.0}$	Number of feet force signal in z direction

Table 7: Description of symbols used in the reward function.

Disturbance	Range
Feet static friction	$[0.5, 1.2]$ N
Feet dynamic friction	$[0.3, 1.2]$ N
Torso mass	$[-10, 10]$ kg
End-effector mass	$[0, 1.8]$ kg
External force on torso	$[-50, 50]$ N
External torque on torso	$[-20, 20]$ Nm
External force on end-effector	$[-3, 3]$ N
Torso push linear velocity	$[-0.2, 0.2]$ m/s
Torso push angular velocity	$[-0.2, 0.2]$ rad/s

Table 8: Disturbances applied to the robot during training.

an 8:1 decimation ratio, the simulation itself runs at 400 Hz to accurately model the robot's main control frequency. We list the hyperparameters used for training in Table 9.

7.1.4 Teacher-Student Distillation

Our teacher policy is a 3-layer MLP with hidden layers of size $[512, 256, 128]$ with ELU activations. The teacher has access to privileged information inaccessible in the real world. To deploy the policy on hardware, a student policy with the same network structure as the teacher is distilled without access to privileged information using supervised learning [16]. During distillation, the student learns to implicitly estimate the privileged information by minimizing the mean squared error between the teacher and student actions. The hyperparameters used for training the student policy are listed in Table 10.

Hyperparameter	Value
Learning rate	$3.0e-4$
Entropy coefficient	0.002
Value loss coefficient	1.0
Clip parameter	0.2
Number of learning epochs	8
Number of mini-batches	4
Discount factor γ	0.99
GAE λ	0.95
Number of steps per environment	24
Number of environments	4096
Number of training iterations	50000

Table 9: Hyperparameters used for training the teacher policy.

Hyperparameter	Value
Learning rate	$1.0e-3$
Number of learning epochs	8
Gradient length	15
Loss type	Mean Squared Error

Table 10: Hyperparameters used for training the student policy.

7.2 Ablation Studies

7.2.1 Effect of Curriculum Learning on End-Effector Twist Commands

Similar to previous works [7, 11] in pose-based end-effector control, we formulate the end-effector twist commands in the control frame, which is defined as the robot-centric gravity-aligned frame that follows only the yaw of the robot's torso. However, as shown by the Control Frame signal in Figure 7, representing the twist command in the control frame makes it challenging for the policy to learn to precisely follow the end-effector trajectory. We observe that as the robot learns to stabi-

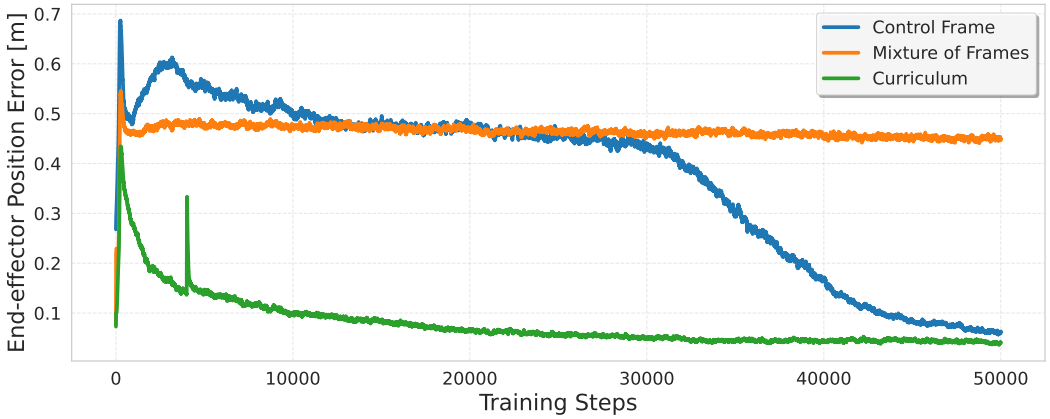

Figure 7: **End-effector position tracking for different command frame definitions**: The curriculum approach demonstrates superior performance compared to using either fixed frame representation or a mixture of frames.

lize its torso and walk, the control frame moves significantly, making the state-reward pairs sparse in representation and leading to difficulties in policy learning. As a solution, we attempted two approaches:

- **Mixture of frames**: In this approach, we provide 50% of the commands to the policy in the base frame, while representing the rest in the control frame.
- **Curriculum**: In this approach, we represent the commands in the base frame until a set number of iterations, after which the commands are provided with respect to the control frame.

In Figure 7, it can be seen that the mixture of frames does not lead to better end-effector position tracking. This could be due to the conflicting representations caused by the two frames in which

the commands are represented. In contrast, we observe that the curriculum approach leads to faster policy learning. In this case, we command the end-effector twist command with respect to the base frame until 3000 iterations, following which the frame switches to the control frame. The switching iteration is based on the observation that robots learn to walk with a stable torso within 3000 iterations. Although we see a jump in the tracking error soon after the frame switch, the tracking error continues to reduce for the rest of the training.

7.2.2 Single-Critic vs Multi-Critic Learning

In this section, we compare the performance of single-critic versus multi-critic learning approaches. We train policies with identical parameters, modifying only the critic architecture, while maintaining consistent reward functions and weights. As shown in Fig. 8, we observe an interesting trade-off: the single-critic policy achieves marginally lower end-effector tracking errors, but completely fails to learn locomotion behaviors. This superior end-effector performance can be attributed to the

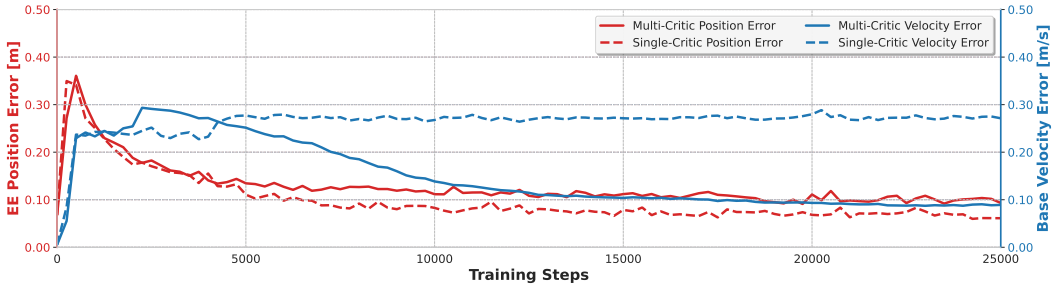

Figure 8: **Comparison of single-critic vs multi-critic frameworks**: Multi-critic policy learns to track both the base and end-effector commands whereas the single-critic policy only learns to track the end-effector commands.

policy that adopts a stationary posture at all times that essentially ignores base-velocity commands, allowing it to focus on the arm control task. In contrast, the multi-critic approach successfully learns to simultaneously satisfy both locomotion and manipulation objectives, demonstrating the ability to coordinate whole-body movement while maintaining precise end-effector motion.

7.2.3 Chicken head control performance while walking

To demonstrate the whole-body task coordination capabilities of the controller, we perform chicken head control with the end-effector while walking. The goal of this experiment is to demonstrate the whole-body loco-manipulation capability of the policy by holding a static end-effector pose in the world frame while moving the torso. During this experiment, the robot is commanded a base velocity within the range of [-0.2, 0.2] m/s while the end-effector is commanded twist corrections to hold the end-effector position in the robot's odometry frame. As shown in Fig. 9, the policy achieves very low end-effector position tracking errors while walking, demonstrating effective task space control during base motion.

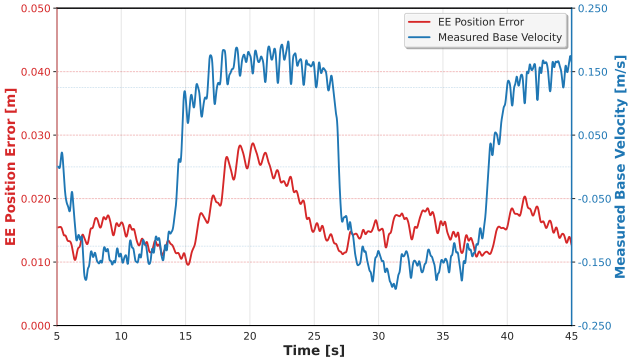

Figure 9: **End-Effector Tracking performance in chicken-head mode**: Tracking error in end effector position (red, left axis) and measured base velocity (blue, right axis) of the robot plotted over time. The position error has a mean of 0.0156 m and maximum of 0.0287 m.