

A Appendix

A.1 Observation and Action Space

The observation of the policy is composed of two main components: state observation and goal observation. State observation at time t include the linear velocity (\mathbf{v}) and angular velocity ($\boldsymbol{\omega}$) of the base in local coordinates, current joint angles ($\boldsymbol{\theta}_j$), current joint velocities ($\dot{\boldsymbol{\theta}}_j$), projected gravity in the base frame (\mathbf{g}_{proj}), base height (h) and previous actions (\mathbf{a}_{prev}),

$$\mathbf{s}_t = \{\mathbf{v}, \boldsymbol{\omega}, \boldsymbol{\theta}_j, \dot{\boldsymbol{\theta}}_j, \mathbf{g}_{proj}, h, \mathbf{a}_{prev}\}_t. \quad (6)$$

A variable number of keyframes $\mathbf{K} = (\mathbf{k}^1, \mathbf{k}^2, \dots, \mathbf{k}^{n_k})$ are specified as targets for the robot. At each time step t , each keyframe \mathbf{k}^i is transformed spatially and temporally into a robot-centric view. Then, the goal observation is prepared by calculating the remaining time to goal $\hat{t}^i - t$ and the error to target goals ($\Delta \mathbf{g}_t^i$),

$$\Delta \mathbf{g}_t^i \subset \{\Delta \mathbf{p}_b^i, \Delta \phi^i, \Delta \zeta^i, \Delta \psi^i, \Delta \boldsymbol{\theta}_j^i\}. \quad (7)$$

Here, $\Delta \mathbf{p}_b^i$ denotes the error between robot base position and keyframe position in the base coordinate frame, $\Delta \boldsymbol{\theta}_j^i$ is the error in joint angles, and $\Delta \phi^i$, $\Delta \zeta^i$ and $\Delta \psi^i$ denote the errors in roll, pitch and yaw angles, respectively, which are wrapped to $(-\pi, \pi]$.

The policy receives the sequence of tokens $\mathbf{X}_t = (\mathbf{x}_t^0, \dots, \mathbf{x}_t^{n_k})$ as input to the encoder, where $\mathbf{x}_t^0 = (\mathbf{s}_t, \mathbf{0}, 0)$, and $\mathbf{x}_t^i = (\mathbf{s}_t, \Delta \mathbf{g}_t^i, \hat{t}^i - t)$ for $i = 1, \dots, n_k$. Thanks to the transformer-based keyframe encoding, the extra tokens can be masked to enable arbitrary number of goals. In addition, keyframes with a time over one second past the current time are also masked to avoid any long-term influence on reaching the future goals.

The action (\mathbf{a}_t) space of the policy is set to target joint angles, which are tracked using a PD controller to compute the motor torques.

A.2 Reward Terms

We include three groups of rewards in this framework: regularization, style, and goal. For each reward group, the final reward is computed as a multiplication of individual reward terms,

$$r_{\text{group}} = \prod_{i \in \text{group}} r_i. \quad (8)$$

Regularization rewards are designed to provide a smooth output of the policy and consist of several terms defined in Table A1. Here, \mathcal{K} is an exponential kernel function defined in Eq. 9 where σ and δ are the sensitivity and tolerance of the kernel function, respectively.

$$\mathcal{K}(\mathbf{x}, \sigma, \delta) = \exp \left(- \left(\frac{\max(0, \|\mathbf{x}\| - \delta)}{\sigma} \right)^2 \right) \quad (9)$$

To generate natural motion between the keyframes, we use AMP proposed by Peng et al. [12], which involves training a discriminator \mathcal{D} to identify motions that are similar to those of the offline expert

Table A1: Regularization Reward Terms

Action rate	$\mathcal{K}(\dot{\mathbf{a}}, 8.0, 0)$
Base horizontal acceleration	$\mathcal{K}(\ddot{\mathbf{p}}_{xy}, 8.0, 0)$
Joint acceleration	$\mathcal{K}(\ddot{\boldsymbol{\theta}}_j, 150.0, 10.0)$
Joint soft limits	$\mathcal{K}(\max(\boldsymbol{\theta}_j - \boldsymbol{\theta}_{j,min}, \boldsymbol{\theta}_{j,max} - \boldsymbol{\theta}), 0.1, 0)$

dataset. The style reward is defined based on the discriminator output of the latest state transition of the robot ($\mathbf{s}_{t-1}, \mathbf{s}_t$),

$$r_{\text{style}} = \max(1 - 0.25(\mathcal{D}(\mathbf{s}_{t-1}, \mathbf{s}_t) - 1)^2, 0). \quad (10)$$

Goal rewards are defined with a temporally sparse kernel $\Phi^i(x)$

$$\Phi^i(x) = \begin{cases} x, & t = \hat{t}^i \\ 0, & \text{otherwise} \end{cases}, \quad (11)$$

and only activated when the corresponding timestep for that goal \hat{t}^i is reached in the episode. The detailed reward terms are defined in table A2.

Table A2: Goal Reward Terms

Goal position	$\Phi^i(\mathcal{K}(\mathbf{p} - \hat{\mathbf{p}}^i, 0.2, 0))$
Goal roll	$\Phi^i(\mathcal{K}(\phi - \hat{\phi}^i, 0.1, 0))$
Goal pitch	$\Phi^i(\mathcal{K}(\zeta - \hat{\zeta}^i, 0.1, 0))$
Goal yaw	$\Phi^i(\mathcal{K}(\psi - \hat{\psi}^i, 0.3, 0))$
Goal posture	$\Phi^i(\mathcal{K}(\ \boldsymbol{\theta}_j - \hat{\boldsymbol{\theta}}_j^i\ , 0.2, 0))$

A.3 Dataset Preparation

We use a database of motion capture from dogs introduced by Zhang et al. [44]. The motions are retargeted to the robot skeleton using inverse kinematics for the end-effectors' positions with some local offsets to compensate for the different proportions of the robot and dog. A subset of around 20 minutes of data was used, removing the undesired motions such as smelling the ground, walking on slopes, etc. We augment this dataset with other motion clips animated by artists to include more diversity in the dataset. The frame rate is adjusted to that of the simulation, i.e. 50 frames per second.

A.4 Training Procedure

We utilize Isaac Gym [55] for simulating the physical environment. At the start of each episode, the robot is either set to a default state or initialized according to a posture and height sampled from the dataset with Reference State Initialization (RSI). RSI plays a crucial role in capturing and learning the specific style of motion, as highlighted in previous studies such as Peng et al. [56]. Keyframes are derived either randomly or directly from a reference data trajectory. Our methodology incorporates a learning curriculum, beginning with keyframes entirely sourced from reference data and progressively increasing the proportion of randomly generated keyframes. To generate random keyframes, we start by selecting a time interval for each goal within a predetermined range. Subsequently, the distance and direction of the target position relative to the previous goal (or the initial position for the first goal) are sampled based on a specified range. The yaw angle is also chosen from a set range and adjusted relative to the previous goal. The robot's full posture is sampled from the dataset to ensure the target posture is feasible. The roll, pitch, and height of the keyframe are aligned with the corresponding attributes of the target posture frame.

The meticulous sampling of target keyframes is critical for ensuring their feasibility and preventing them from impeding effective policy learning. We train the policy to handle a maximum number of keyframes, randomly selecting the actual number of keyframes for each episode. To avoid negative impacts on training, unused goals are masked when input into the transformer encoder. For stability, the episode does not terminate immediately after the last goal is reached; instead, it terminates approximately one second later. The training setup for a full keyframe comprising time, position,

roll, pitch, yaw, and posture targets with up to 5 maximum keyframes requires approximately 17 hours on a system equipped with Nvidia GeForce RTX 4090.

A.5 Hardware Implementation Details

Domain randomization is added during training to achieve a robust policy that can be executed on hardware. Similar to Kang et al. [58], we randomize friction coefficients, motor stiffness and damping gains and actuator latency. Furthermore, we add external pushes during training. Although joint limits are softly taken into account in the simulation, we found it crucial to terminate episodes when reaching joint limits to ensure a stable deployment on hardware. We use a motion capture system to receive the global position and orientation of the robot. These are used to compute the relative errors to the target goals and are then passed to the policy. Other observations are computed based on the outputs from the state estimator.

A.6 Future goal anticipation

Details of target keyframes used for Table 1 are given in Table A3.

Table A3: Details of Keyframe Scenarios

Scenario	First Goal		Second Goal	
	Time (steps)	Position (m)	Time (steps)	Position (m)
Straight	50	(0, 0.32, 1.0)	75	(0, 0.32, 2.0)
Turn	50	(0, 0.32, 1.0)	75	(1.0, 0.32, 1.5)
Turn (Slow)	50	(0, 0.32, 1.0)	100	(1.0, 0.32, 1.5)

A.7 Training Hyperparameters

Table A4 provides details of hyperparameters used for training.

Table A4: Summary of Training Hyperparameters

Number of environments	4096
Number of mini-batches	4
Number of learning epochs	5
Learning rate	0.0001
Entropy coefficient	0.02
Target KL divergence	0.02
Gamma	0.99
Lambda	0.95
Discriminator learning rate	0.0003
Transformer encoder layers	2
Transformer heads	1
Transformer feed-forward dimensions	512
MLP dimensions	[512, 256]
Initial standard deviation	1.0
Activation function	ELU