

RobotKeyframing: Learning Locomotion with High-Level Objectives via Mixture of Dense and Sparse Rewards

Anonymous Author(s)

Affiliation

Address

email

1 **Abstract:** This paper presents a novel learning-based control framework that uses
2 keyframing to incorporate high-level objectives in natural locomotion for legged
3 robots. These high-level objectives are specified as a variable number of partial
4 or complete pose targets that are spaced arbitrarily in time. Our proposed frame-
5 work utilizes a multi-critic reinforcement learning algorithm to effectively handle
6 the mixture of dense and sparse rewards. Additionally, it employs a transformer-
7 based encoder to accommodate a variable number of input targets, each associated
8 with specific time-to-arrivals. Throughout simulation and hardware experiments,
9 we demonstrate that our framework can effectively satisfy the target keyframe
10 sequence at the required times. The experiments also show that the multi-critic
11 method significantly reduces the effort for hyperparameter tuning compared to
12 the standard single-critic alternative. Moreover, the proposed transformer-based
13 architecture enables robots to anticipate future goals, which results in quantitative
14 improvements in their ability to reach their targets.

15 **Keywords:** Legged Robots, Multi-Critic Reinforcement Learning, Motion Imita-
16 tion



Figure 1: *RobotKeyframing*: Locomotion policy trained with our framework meets the keyframes with position and full posture targets (yellow) at specified times on hardware experiments.

17 1 Introduction

18 Legged robots hold a significant promise for becoming household companions or automated per-
19 formers in the entertainment industry [1, 2, 3]. In these applications, it is crucial for robot controllers
20 to perform natural and directable behavior from simple high-level user command inputs beyond the
21 common commands used in the robotic domain such as joystick velocity commands [4, 5] or target
22 base position [6, 7].

23 In the character animation domain, a widely used technique for specifying character behavior from
24 simple and sparse inputs is *keyframing* [8, 9]. It involves defining the target position or kinematic

25 pose of the character at particular points in time, allowing animators to create smooth movements
26 by interpolating between these keyframes. Despite its proven effectiveness within the kinematic
27 animation pipeline, incorporating keyframing for achieving time-specific targets remains unexplored
28 in the realm of physics-based robot control.

29 Inspired by character animation, we aim to equip legged robots with more refined control by in-
30 corporating sparse and temporal high-level objectives as keyframes. The primary goal of this work
31 is to develop a locomotion controller that enables the robot to fulfill specified partial or full-pose
32 targets while infilling natural behavior during the intermediate periods. This goal aligns with recent
33 advancements in using reinforcement learning (RL) for legged robots due to their promising robust-
34 ness and flexibility [10, 11]. However, learning a policy that accurately meets keyframes without
35 imposing undesired constraints at intermediate periods presents challenges, particularly due to the
36 need to handle sparsity in the keyframe objectives. Acquiring effective policies requires a meticu-
37 lous reward design procedure that carefully balances these sparse rewards with other dense rewards
38 which are crucial for regularizing and encouraging natural motion.

39 In this work, we present a novel framework that unifies timed high-level objectives with natural lo-
40 comotion of legged robots through temporal keyframes. Along with the imitation objective similar
41 to Peng et al. [12] for natural motion generation, our pipeline allows specifying full or partial high-
42 level targets, including base position, orientation, and joint postures. We propose using a multi-critic
43 RL framework to address the challenge of managing groups of sparse and dense rewards by learning
44 distinct value functions. Our method also employs a novel transformer-based architecture to encode
45 a variable number of goals with arbitrary time intervals. Unlike typical sequence-to-sequence trans-
46 formers [13], we propose a lightweight sequence-to-token module that can be used autoregressively
47 within a feedback control loop. We demonstrate the effectiveness of our framework through experi-
48 ments both in simulation and on real-world hardware. Our policies successfully guides the robot to
49 meet multiple keyframes at the required times, for both position and posture targets. Furthermore,
50 the multi-critic approach showcases better convergence with less hyperparameter tuning compared
51 to the conventional single-critic method. Our experiments also reveals that using a transformer-based
52 encoder to anticipate future goals significantly enhances goal-reaching accuracy.

53 The contribution of this paper is threefold: (i) We introduce *RobotKeyframing*, a novel learning-
54 based framework for integrating high-level objectives in natural locomotion of legged robots; (ii)
55 We propose using multi-critic RL to handle the mixture of dense and sparse rewards and a novel
56 sequence-to-token encoder to accommodate a variable number of keyframes; (iii) We validate the
57 effectiveness of our method through extensive experiments in simulation and on hardware.

58 **2 Related Work**

59 **2.1 Reinforcement Learning for Legged Robots**

60 Over the last decade, reinforcement learning has been increasingly applied to develop locomotion
61 policies for legged robots [4, 14, 15]. The primary focus has been to achieve robust control policies
62 that can accurately track velocity commands from joysticks [11, 16, 17]. More recently, researchers
63 have attempted to enhance the versatility of legged robot controllers by incorporating high-level ob-
64 jectives, particularly through position- or orientation-based targets [18, 19, 20]. This high-level con-
65 trol is typically accomplished through hierarchical frameworks, where a high-level policy is learned
66 to drive a low-level controller [7, 21, 22]. Conversely, end-to-end approaches aim to develop a uni-
67 fied policy for both high- and low-level control, allowing high-level objectives to directly influence
68 low-level decisions [6, 18, 23]. However, the aforementioned methods typically urge the robot to
69 reach a target as fast as possible, lacking refined control over the temporal profile of achieving the
70 target. Inspired by keyframing in animation, this work aims to further expand control over robot
71 motion by incorporating multiple keyframes as input to the control policy, thereby enabling robots
72 to generate diverse behaviors in reaching targets. We further enhance this versatility by allowing
73 partial or full targets, including base position, orientation, and joint postures.

74 2.2 Natural Motion for Characters and Robots

75 Synthesizing naturalistic behavior from existing motion datasets while fulfilling spatial or temporal
76 conditions has been extensively studied in the character animation domain [24, 25, 26, 27]. Existing
77 research for generating natural motions between keyframes [28, 29, 30] has mainly focused on the
78 kinematic properties of characters and thus cannot be directly applied to physics-based characters
79 or robots, whose dynamic interactions with the environment require consideration of both kinemat-
80 ics and dynamics. Various efforts have also been made to combine kinematic motion generation
81 with physically controlled robots to achieve natural behavior on hardware [31, 32, 33, 34]. Another
82 thread of research focuses on controlling characters in physically simulated environments, incorpo-
83 rating motion datasets as demonstrations [35, 36, 37, 38]. Some of these methods have also been
84 successfully transferred to robot control for quadrupeds or humanoids [39, 40, 41, 42]. Among
85 these works, Adversarial Motion Priors (AMP) [12] provides a flexible way to encourage the pol-
86 icy to have natural, expert-like behavior by connecting generative adversarial networks (GAN) [43]
87 with RL given an offline motion dataset. We also incorporate an AMP-based imitation objective to
88 encourage naturalistic motion for the policy and further extend it to infilling keyframes for robots.

89 3 Method

90 3.1 Problem Setup

91 To integrate high-level control objectives into the robotic control framework, we employ sparse
92 keyframes that require a robot to achieve specific goals at predetermined times. Each keyframe con-
93 tains a full or partial combination of a variety of targets such as global base position $\hat{\mathbf{p}} \in \mathbb{R}^3$, global
94 base orientation $(\hat{\phi}, \hat{\zeta}, \hat{\psi}) \in \mathbb{R}^3$ where ϕ, ζ, ψ denote roll, pitch, and yaw angles respectively, and
95 full posture specified by joint angles $\hat{\boldsymbol{\theta}}_j \in \mathbb{R}^{N_j}$ where N_j is the number of joints. Each keyframe is
96 also assigned with a specific time $\hat{t} \in \mathbb{R}$ in the future at which the robot is expected to meet the goals.
97 In summary, the high-level objectives are specified through these keyframes $\mathbf{K} = (\mathbf{k}^1, \mathbf{k}^2, \dots, \mathbf{k}^{n_k})$,
98 where $\mathbf{k}^i = (\hat{\mathbf{g}}, \hat{t})^i$ and $\hat{\mathbf{g}} \subset \{\hat{\mathbf{p}}, \hat{\phi}, \hat{\zeta}, \hat{\psi}, \hat{\boldsymbol{\theta}}_j\}$. Here, $n_k \leq N_k$ where n_k and N_k denote the ac-
99 tual and the maximal number of keyframes, respectively. We aim to support an arbitrary number of
100 keyframes, allowing for the flexible specification of high-level objectives only as needed.

101 The main goal is to train a locomotion policy for legged robots that not only meets these keyframes
102 but also maintains a natural style in the intervals between them. To avoid undesired restrictions on
103 the intermediate periods, policy’s task performance is evaluated exclusively at the designated times,
104 making the keyframe objectives temporally sparse. However, relying solely on keyframes to train
105 the control policy may result in undesirable motions. Thus, it is crucial to have additional rewards
106 for regularizing and promoting a natural motion style. In this regard, we incorporate AMP [12] as
107 a general style guide for the robot, encouraging the policy to behave naturally and similarly to an
108 offline motion dataset from real animals [44]. The style and regularization rewards are evaluated at
109 every step of the episode, making them temporally dense. The mixture of sparse and dense rewards
110 presents a unique challenge that is difficult to manage effectively with standard RL frameworks.
111 Further details on the observation, action, reward definitions, and training procedure can be found
112 in Appendix A.

113 3.2 Multi-Critic RL for Dense-Sparse Reward Mixture

114 Modern RL algorithms [45, 46, 47] typically employ the actor-critic paradigm, where the actor
115 decides the action to take, and the critic evaluates the action by estimating the value function. To
116 effectively manage a complex mixture of temporally dense and sparse rewards, we employ a multi-
117 critic (MuC) RL framework by Martinez-Piazuelo et al. [48] as shown in Fig. 2. It involves training
118 a set of critic networks $\{V_{\phi_i}\}_{i=0}^n$ to learn distinct value functions associated with different reward
119 groups $\{r_i\}_{i=0}^n$. Similar concepts have been used to balance a set of dense rewards [49, 50]; however,
120 we aim to adapt the multi-critic method to the context of dense and sparse reward combination.

121 We design each reward group to contain either exclusively dense or sparse rewards. This division
 122 is essential for effectively managing the distinct temporal characteristics of each reward type and
 123 facilitates value estimation.

124 We integrate the multi-critic concept to Proximal Policy Optimization (PPO) [46], as shown
 125 in Alg. 1. Particularly, each value network $V_{\phi_i}(\cdot)$ is trained independently for a specific reward
 126 group r_i with temporal difference loss,

$$L(\phi_i) = \hat{\mathbb{E}}_t \left[\|r_{i,t} + \gamma V_{\phi_i}(s_{t+1}) - V_{\phi_i}(s_t)\|^2 \right], \quad (1)$$

127 where $\hat{\mathbb{E}}_t$ is the empirical average and γ is the discount factor. The value functions calculated by each
 128 critic are used to individually estimate the advantage $\{\hat{A}_i\}_{i=0}^n$ for each reward group. Subsequently,
 129 these advantages are synthesized into a policy improvement step by calculating the multi-critic ad-
 130 vantage as a weighted sum of the normalized advantages from each reward group

$$\hat{A}_{MuC} = \sum_{i=0}^n w_i \cdot \frac{\hat{A}_i - \mu_{\hat{A}_i}}{\sigma_{\hat{A}_i}}, \quad (2)$$

131 where $\mu_{\hat{A}_i}$ and $\sigma_{\hat{A}_i}$ are the batch mean and standard deviation of the advantage from group i . Similar
 132 to PPO, the surrogate loss for policy gradient is clipped

$$L^{CLIP-MuC}(\theta) = \hat{\mathbb{E}}_t \left[\min \left(\alpha_t(\theta) \hat{A}_{MuC,t}, \text{clip}(\alpha_t(\theta), 1 - \epsilon, 1 + \epsilon) \hat{A}_{MuC,t} \right) \right], \quad (3)$$

133 where $\alpha_t(\theta)$ and ϵ respectively denote the probability ratio and the clipping hyperparameter. This
 134 formulation integrates feedback from both dense and sparse rewards into the policy update, faci-
 135 litating a balanced and effective learning process.

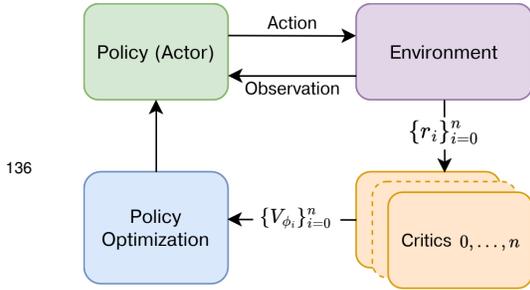


Figure 2: Multi-Critic RL.

Algorithm 1 Multi-Critic PPO

- 1: Initialize policy parameters θ and parameters of each critic, ϕ_i .
 - 2: **for** $n = 1$ to N **do**
 - 3: Rollout policy π_θ to fill the buffer.
 - 4: **for** each mini-batch **do**
 - 5: Estimate \hat{A}_i for each r_i .
 - 6: Compute \hat{A}_{MuC} with Eq. 2.
 - 7: Update policy with Eq. 3.
 - 8: Update each critic with Eq. 1.
 - 9: **end for**
 - 10: **end for**
-

137 Assigning distinct critics for dense and sparse rewards helps achieve each set of objectives more ef-
 138 fectively while reducing the reliance on extensive hyperparameter tuning. To illustrate this, consider
 139 a simple scenario with an episode length of T involving two types of rewards: a temporally dense
 140 reward r_d that is active at every step and a temporally sparse reward r_s that is only active at the final
 141 step of an episode

$$r_{s,t} = \begin{cases} \hat{r}_s, & t = T \\ 0, & \text{otherwise.} \end{cases} \quad (4)$$

142 In the conventional single-critic RL, the total reward of each time step t is typically computed as a
 143 linear combination of different reward terms $r_t = w_s r_{s,t} + w_d r_{d,t}$. The value in this scenario is

$$V(s_t) = \mathbb{E} \left[w_s \gamma^{(T-t)} \hat{r}_s + w_d \sum_{k=t}^T \gamma^k r_{d,k} \right]. \quad (5)$$

144 We define the reward sparsity ratio as the number of dense reward steps per sparse reward horizon,
 145 which is here equal to T . The second term in Eq. 5 consists of a summation over $T - t$ individual
 146 reward terms, whereas the first term includes only a single component. This highlights the impact of

147 different reward sparsities on the learning process, suggesting that the weight of reward groups must
 148 be adjusted for different sparsity ratios to achieve a proper balance. This challenge is amplified when
 149 the sparsity ratio changes between episodes, for example, when keyframe timings are randomly
 150 sampled within a range. These variations can complicate the hyperparameter tuning process and
 151 hinder the efficacy of the learning algorithm.

152 In the multi-critic approach, on the other hand, the advantage for each reward group is normalized
 153 independently, ensuring that a fixed weight ratio for the advantages is adequate to maintain the de-
 154 sired balance, regardless of variations in the sparsity ratio. This method decouples reward frequency
 155 and magnitude from the learning process, enabling more effective policy optimization and reducing
 156 the effort for manual hyperparameter tuning.

157 3.3 Transformer-based Keyframe Encoding

158 The transformer framework [51] has achieved great success in modeling sequential data not only
 159 in the natural language processing [52, 53] but also in other areas including robotics [54]. The
 160 attention mechanism, serving as the core of transformer networks, models the correlation between
 161 each element of the input sequence and reweights them accordingly. To handle a variable number of
 162 keyframes in our problem, we utilize a transformer-based encoder to process the sequence of goals
 163 for both the policy and critics. However, unlike the typical application of transformers in sequence-
 164 to-sequence tasks, we adapt the architecture to function in a sequence-to-token manner, as shown
 165 in Fig. 3. This adaptation makes it suitable for autoregressive feedback control in robotic systems.

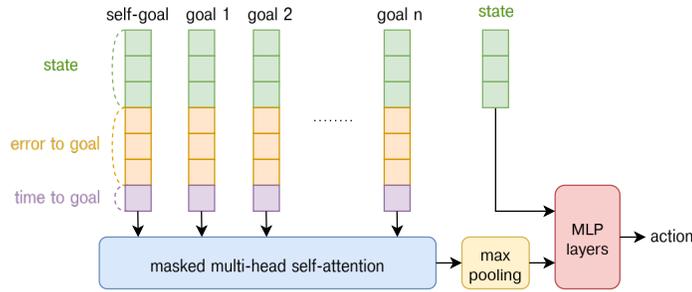


Figure 3: Policy with transformer-based keyframe encoder.

166 In our system, each input token corresponds to a particular keyframe. At every time step t , each
 167 keyframe k^i is transformed spatially and temporally into a robot-centric view, resulting in a goal
 168 error Δg_t^i and a calculated time to goal $\hat{t}^i - t$. These are then concatenated with the robot state
 169 s_t to form a single token. Additionally, we incorporate a *self-goal* keyframe, x_t^0 , as the first token
 170 in the sequence. This token represents a state with zero error and zero time to goal, which ensures
 171 that the control system remains operational despite the absence of active goals or after achieving
 172 all goals. The transformer encoder receives the sequence of tokens $\mathbf{X}_t = (x_t^0, \dots, x_t^{n_k})$, where
 173 $x_t^0 = (s_t, \mathbf{0}, 0)$, and $x_t^i = (s_t, \Delta g_t^i, \hat{t}^i - t)$ for $i = 1, \dots, n_k$.

174 In scenarios where the number of active keyframes is less than the maximum capacity of the system,
 175 we apply masking to ignore the surplus tokens and focus only on the relevant keyframes. Further-
 176 more, we also apply masking to keyframes once their designated time is reached and surpassed by a
 177 few steps. This practice prevents past goals from inappropriately influencing the long-term behavior
 178 of the policy. The output from the transformer encoder is then forwarded to a max-pooling layer,
 179 which condenses the encoded goal features for delivery to the subsequent multilayer perceptrons
 180 (MLP). By leveraging transformer’s ability to handle sequences of varying lengths, our architecture
 181 can effectively integrate multiple and arbitrary numbers of goals into the control process.

182 4 Results

183 The control policies are trained for quadruped robots with 12 degrees of freedom (DoF) using Isaac
 184 Gym [55]. At the start of each episode, the robot is either set to a default state or initialized according

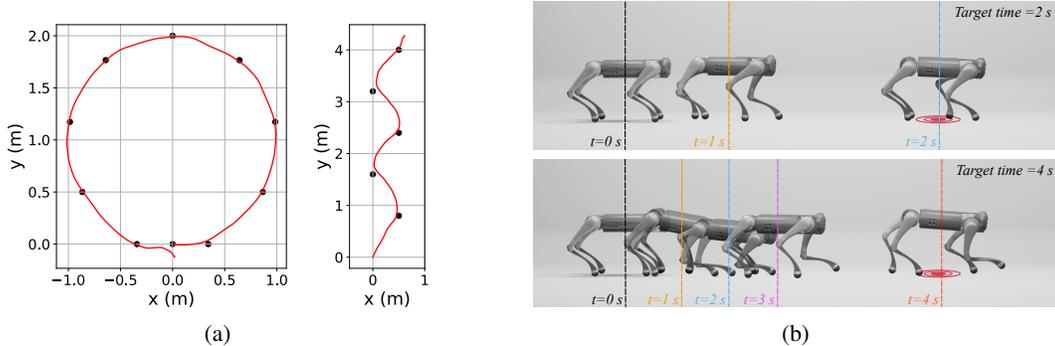


Figure 4: **a)** Horizontal trajectories of the robot base given two sets of position goals (dots). **b)** Specifying different temporal profiles generates diverse behaviors for the same position goal.

185 to a posture and height sampled from the dataset, a technique known as Reference State Initialization
 186 (RSI) [56]. We incorporate a learning curriculum, beginning with keyframes entirely sourced from
 187 reference data and progressively increasing the proportion of randomly generated keyframes, with
 188 time intervals, position targets, and yaw angles each sampled from a predetermined range. In this
 189 section, we present the qualitative and quantitative experiment results in simulation and on hardware.

190 4.1 Keyframe Tracking

191 We demonstrate that our trained policy effectively reaches keyframes at the designated times through
 192 several simulation experiments. Given keyframes consisting of position goals, our policy reaches its
 193 targets with notable precision, as illustrated in Fig. 4a by the horizontal trajectories for two example
 194 scenarios with different number of keyframes. Furthermore, our framework offers control over
 195 target reaching time and can generate diverse behaviors for the same targets by specifying different
 196 time profiles. This is depicted in Fig. 4b through snapshots of robot motion when provided with
 197 keyframes consisting of the same position goal, but different target times. Full posture targets are
 198 also supported along with position and orientation goals. Fig. 5 shows snapshots of the robot motion
 199 given different keyframe scenarios, highlighting that our policy accurately meets its full posture
 200 targets and maintains a natural style while reaching them.

201 4.2 Multi-Critic RL

202 In this section, we conduct a comparative analysis between multi-critic and single-critic approaches
 203 in the keyframing setup. Learning curves for both methods are presented in Fig. 6, with each method
 204 trained across three different ranges of sparsity ratios by sampling keyframes with varying time

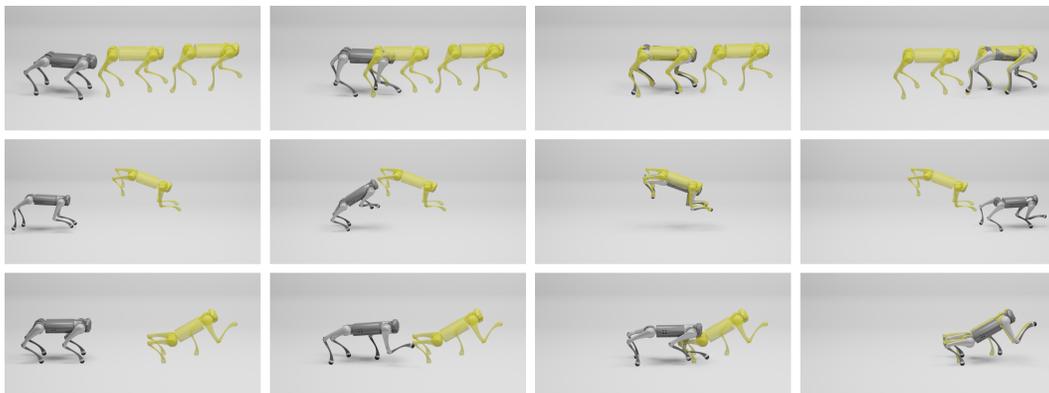


Figure 5: Snapshots of the robot motion given keyframes with full postures: moving forward (**top**), jumping (**middle**) and raising the paw up (**bottom**). Target keyframes are displayed in yellow.

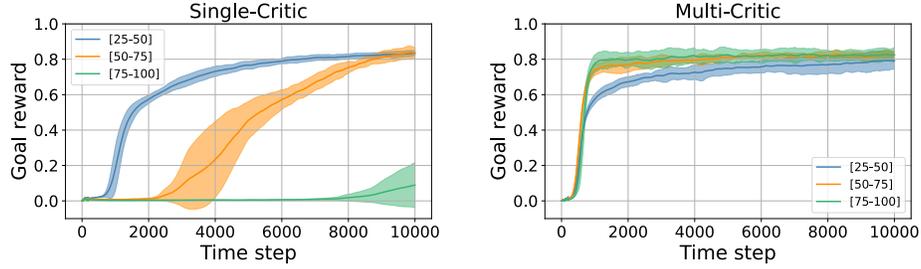


Figure 6: Convergence comparison of single-critic (**left**) and multi-critic (**right**) for different ranges of keyframe time horizons ($[25, 50]$, $[50, 75]$, $[75, 100]$) with fixed weights.

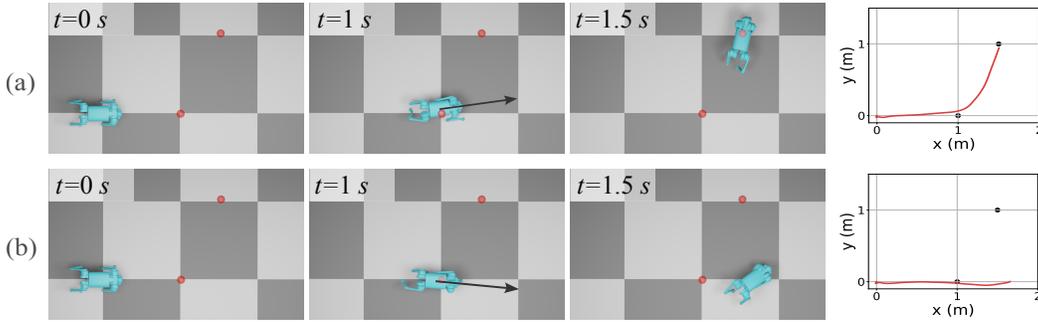


Figure 7: The policy aware of all goals (**a**) adjusts its yaw angle earlier to better reach the second goal compared to the policy only aware of the next goal (**b**). Keyframes are placed at 1 and 1.5 seconds in time. Left: snapshots, right: trajectories.

205 horizons. Initially, reward and advantage weights are tuned separately for single- and multi-critic
 206 according to the time horizon range $[25, 50]$. New policies are then trained using the same weights
 207 for another two scenarios of time horizons, $[50, 75]$ and $[75, 100]$. The learning curves reveal that the
 208 multi-critic algorithm achieves a similarly fast convergence without retuning the advantage weights
 209 for different scenarios. In contrast, the single-critic method displays significant delays in reward
 210 increase due to the sparser nature with longer keyframe horizons, underscoring the efficiency of the
 211 multi-critic in reducing the need for extensive manual hyperparameter tuning. This feature makes
 212 multi-critic particularly valuable in environments with varying reward sparsities.

213 4.3 Future Goal Anticipation

214 An advantage of using a transformer-based encoder is that it enables the policy to incorporate multi-
 215 ple and a varying number of goals as input. If the goals are temporally close to each other, awareness
 216 of future goals influences the robot’s motion to achieve all of them more accurately. The phe-
 217 nomenon of future goal anticipation is demonstrated in Fig. 7 where we compare a policy aware of
 218 all goals and a policy only aware of the immediate next goal, both trained with only position goals in
 219 the keyframe. The policy trained with multiple keyframes adopts a larger yaw angle at the first goal,
 220 leaning more towards the second one to be able to reach it with higher accuracy. Table 1 provides a
 221 quantitative comparison of the two policies across three different scenarios: straight, turn and slow
 222 turn, the latter featuring a longer time horizon for the second goal. The results indicate that future
 223 goal anticipation helps the policy to adjust its motion while approaching earlier goals to gain better
 224 accuracy for the subsequent targets. This is particularly important when keyframes are temporally
 225 close, resulting in higher accuracy gains in fast and dynamic movements, compared to slower ones.

226 4.4 Hardware Deployment

227 We validate our method through extensive hardware experiments using the Unitree Go2 [57], a
 228 12-DoF commercial quadruped robot. Fig. 8 illustrates the outcomes of a policy that manages up

First Goal	Straight	Turn	Turn (Slow)
Aware of all goals	0.0872 ± 0.0336	0.0781 ± 0.0236	0.0806 ± 0.0265
Aware of next goal	0.0898 ± 0.0317	0.0841 ± 0.0335	0.0787 ± 0.0208
Second Goal	Straight	Turn	Turn (Slow)
Aware of all goals	0.0472 ± 0.0187	0.3340 ± 0.1162	0.0566 ± 0.0804
Aware of next goal	0.1332 ± 0.0605	0.7271 ± 0.1528	0.1711 ± 0.1071

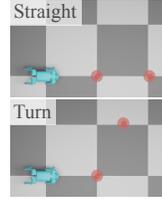


Table 1: Average position error (m) for three keyframe scenarios (depicted on right) across 20 experiments. The policy aware of all goals achieves better accuracy in reaching them.



Figure 8: Hardware deployment of *RobotKeyframing* for position targets (**top**), and full-pose targets (**bottom**). Posture keyframes are displayed in yellow.

229 to 5 positional goals arranged in different courses, and a policy trained for full pose targets that
 230 successfully drives the robot to achieve various posture keyframes. These experiments underscore
 231 the adaptability and effectiveness of our keyframing approach in enhancing high-level control in
 232 robotic systems. Readers are encouraged to watch the videos provided in the supplementary material
 233 for a more comprehensive presentation of these results.

234 5 Discussion

235 **Conclusion:** This paper presents *RobotKeyframing*, a learning-based control framework designed
 236 to incorporate high-level objectives into the natural locomotion of legged robots through a sequence
 237 of keyframes. Simulation and hardware experiments demonstrate the efficacy of our framework.
 238 The sparse reward imposed by keyframe objectives is effectively handled by a multi-critic PPO
 239 algorithm. In addition, the transformer-based architecture is adaptive to an arbitrary number of
 240 target keyframes and improves accuracy in reaching targets through future goal anticipation.

241 **Limitations and future work:** First, if the timing values are infeasible for the specified goals, the
 242 robot may fail to meet the targets. However, it is worth noting that such cases do not result in uncon-
 243 trolled behaviors, such as falling down. Second, our approach inherits the mode collapse issue from
 244 the AMP framework [12], which can be mitigated in future research through the integration of style
 245 embeddings. Third, the performance of our policy is currently limited by the motions in the dataset,
 246 restricting its ability to generalize to out of distribution motions or targets. Looking ahead, our
 247 method can be expanded to incorporate diverse types of goals in the keyframes, such as end-effector
 248 targets or more intuitive high-level inputs such as skill or text. Additionally, *RobotKeyframing* can be
 249 extended to more complex characters and potentially used for physics-based motion in-betweening
 250 in the character animation domain.

References

- 251
- 252 [1] M. Hopkins, G. Wiedebach, K. Cesare, J. Bishop, E. Knoop, and M. Bächer. Inter-
253 active design of stylized walking gaits for robotic characters. *ACM Transactions*
254 *on Graphics (ToG)*, 2024. URL [https://la.disneyresearch.com/publication/
255 interactive-design-of-stylized-walking-gaits-for-robotic-characters/](https://la.disneyresearch.com/publication/interactive-design-of-stylized-walking-gaits-for-robotic-characters/).
- 256 [2] Sony Electronics. Aibo companion robot. <https://electronics.sony.com/t/aibo>.
- 257 [3] Walt Disney Imagineering. A new approach to disney’s robotic character pipeline. [https://
258 //youtu.be/-cfIm06tcfA?si=G1y14ZTBdlYE8ZtP](https://youtu.be/-cfIm06tcfA?si=G1y14ZTBdlYE8ZtP), 2023.
- 259 [4] J. Lee, J. Hwangbo, L. Wellhausen, V. Koltun, and M. Hutter. Learning quadrupedal lo-
260 comotion over challenging terrain. *Science Robotics*, 5(47):eabc5986, 2020. doi:10.1126/
261 scirobotics.abc5986.
- 262 [5] T. Miki, J. Lee, J. Hwangbo, L. Wellhausen, V. Koltun, and M. Hutter. Learning robust per-
263 ceptive locomotion for quadrupedal robots in the wild. *Science Robotics*, 7(62):eabk2822,
264 2022.
- 265 [6] N. Rudin, D. Hoeller, M. Bjelonic, and M. Hutter. Advanced skills by learning locomotion and
266 local navigation end-to-end. In *2022 IEEE/RSJ International Conference on Intelligent Robots
267 and Systems (IROS)*, pages 2497–2503. IEEE, 2022.
- 268 [7] D. Hoeller, N. Rudin, D. Sako, and M. Hutter. Anymal parkour: Learning agile navigation for
269 quadrupedal robots. *Science Robotics*, 9(88):eadi7566, 2024.
- 270 [8] D. Sturman. Interactive key frame animation of 3-d articulated models. In *Graphics Interface*,
271 volume 86, 1984.
- 272 [9] F. Thomas and O. Johnston. *Disney animation: The illusion of life*. Disney Editions, 1981.
- 273 [10] J. Hwangbo, J. Lee, A. Dosovitskiy, D. Bellicoso, V. Tsounis, V. Koltun, and M. Hutter. Learn-
274 ing agile and dynamic motor skills for legged robots. *Science Robotics*, 4(26):eaa5872, 2019.
- 275 [11] A. Kumar, Z. Fu, D. Pathak, and J. Malik. Rma: Rapid motor adaptation for legged robots.
276 *Robotics: Science and Systems*, 2021.
- 277 [12] X. B. Peng, Z. Ma, P. Abbeel, S. Levine, and A. Kanazawa. Amp: adversarial motion priors
278 for stylized physics-based character control. *ACM Transactions on Graphics (ToG)*, 40(4), jul
279 2021. ISSN 0730-0301. doi:10.1145/3450626.3459670.
- 280 [13] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polo-
281 sukhin. Attention is all you need. *Advances in neural information processing systems*, 30,
282 2017.
- 283 [14] J. Tan, T. Zhang, E. Coumans, A. Iscen, Y. Bai, D. Hafner, S. Bohez, and V. Vanhoucke. Sim-
284 to-real: Learning agile locomotion for quadruped robots. *arXiv preprint arXiv:1804.10332*,
285 2018.
- 286 [15] J. Hwangbo, J. Lee, A. Dosovitskiy, D. Bellicoso, V. Tsounis, V. Koltun, and M. Hutter. Learn-
287 ing agile and dynamic motor skills for legged robots. *Science Robotics*, 4(26):eaa5872, 2019.
- 288 [16] A. Agarwal, A. Kumar, J. Malik, and D. Pathak. Legged locomotion in challenging terrains
289 using egocentric vision. In K. Liu, D. Kulis, and J. Ichnowski, editors, *Proceedings of The 6th
290 Conference on Robot Learning*, volume 205 of *Proceedings of Machine Learning Research*,
291 pages 403–415. PMLR, 14–18 Dec 2023.
- 292 [17] G. B. Margolis and P. Agrawal. Walk these ways: Tuning robot control for generalization with
293 multiplicity of behavior. In *Conference on Robot Learning*, pages 22–31. PMLR, 2023.

- 294 [18] J. Cheng, M. Vlastelica, P. Koley, C. Li, and G. Martius. Learning diverse skills for local
295 navigation under multi-constraint optimality. *arXiv preprint arXiv:2310.02440*, 2023.
- 296 [19] T. Miki, J. Lee, L. Wellhausen, and M. Hutter. Learning to walk in confined spaces using 3d
297 representation. *arXiv preprint arXiv:2403.00187*, 2024.
- 298 [20] P. Arm, M. Mittal, H. Kolvenbach, and M. Hutter. Pedipulate: Enabling manipulation skills
299 using a quadruped robot’s leg. *arXiv preprint arXiv:2402.10837*, 2024.
- 300 [21] J. Truong, D. Yarats, T. Li, F. Meier, S. Chernova, D. Batra, and A. Rai. Learning naviga-
301 tion skills for legged robots with learned robot embeddings. In *2021 IEEE/RSJ International
302 Conference on Intelligent Robots and Systems (IROS)*, pages 484–491. IEEE, 2021.
- 303 [22] J. Lee, M. Bjelonic, A. Reske, L. Wellhausen, T. Miki, and M. Hutter. Learning robust au-
304 tonomous navigation and locomotion for wheeled-legged robots. *Science Robotics*, 9(89):
305 eadi9641, 2024.
- 306 [23] X. Cheng, K. Shi, A. Agarwal, and D. Pathak. Extreme parkour with legged robots. *arXiv
307 preprint arXiv:2309.14341*, 2023.
- 308 [24] D. Holden, T. Komura, and J. Saito. Phase-functioned neural networks for character control.
309 *ACM Transactions on Graphics (TOG)*, 36(4):1–13, 2017.
- 310 [25] H. Zhang, S. Starke, T. Komura, and J. Saito. Mode-adaptive neural networks for quadruped
311 motion control. *ACM Transactions on Graphics (TOG)*, 37(4):1–11, 2018.
- 312 [26] H. Y. Ling, F. Zinno, G. Cheng, and M. Van De Panne. Character controllers using motion
313 vaes. *ACM Transactions on Graphics (TOG)*, 39(4):40–1, 2020.
- 314 [27] G. Tevet, S. Raab, B. Gordon, Y. Shafir, D. Cohen-or, and A. H. Bermano. Human motion
315 diffusion model. In *The Eleventh International Conference on Learning Representations*, 2023.
316 URL <https://openreview.net/forum?id=SJ1kSy02jwu>.
- 317 [28] F. G. Harvey, M. Yurick, D. Nowrouzezahrai, and C. Pal. Robust motion in-betweening. *ACM
318 Transactions on Graphics (TOG)*, 39(4):60–1, 2020.
- 319 [29] J. Qin, Y. Zheng, and K. Zhou. Motion in-betweening via two-stage transformers. *ACM
320 Transactions on Graphics (TOG)*, 41(6):184–1, 2022.
- 321 [30] P. Starke, S. Starke, T. Komura, and F. Steinicke. Motion in-betweening with phase manifolds.
322 *Proc. ACM Comput. Graph. Interact. Tech.*, 6(3), aug 2023. doi:10.1145/3606921.
- 323 [31] D. Kang, F. De Vincenti, N. C. Adami, and S. Coros. Animal motions on legged robots using
324 nonlinear model predictive control. In *2022 IEEE/RSJ International Conference on Intelligent
325 Robots and Systems (IROS)*, pages 11955–11962. IEEE, 2022.
- 326 [32] D. Kang, S. Zimmermann, and S. Coros. Animal gaits on quadrupedal robots using motion
327 matching and model-based control. In *2021 IEEE/RSJ International Conference on Intelligent
328 Robots and Systems (IROS)*, pages 8500–8507. IEEE, 2021.
- 329 [33] X. Huang, Y. Chi, R. Wang, Z. Li, X. B. Peng, S. Shao, B. Nikolic, and K. Sreenath. Diffuse-
330 loco: Real-time legged locomotion control with diffusion from offline datasets. *arXiv preprint
331 arXiv:2404.19264*, 2024.
- 332 [34] I. Radosavovic, B. Zhang, B. Shi, J. Rajasegaran, S. Kamat, T. Darrell, K. Sreenath, and J. Ma-
333 lik. Humanoid locomotion as next token prediction. *arXiv preprint arXiv:2402.19469*, 2024.
- 334 [35] X. B. Peng, P. Abbeel, S. Levine, and M. Van de Panne. Deepmimic: Example-guided deep re-
335 inforcement learning of physics-based character skills. *ACM Transactions On Graphics (TOG)*,
336 37(4):1–14, 2018.

- 337 [36] K. Bergamin, S. Clavet, D. Holden, and J. R. Forbes. Drecon: data-driven responsive control
338 of physics-based characters. *ACM Transactions on Graphics (TOG)*, 38(6):1–11, 2019.
- 339 [37] X. B. Peng, Z. Ma, P. Abbeel, S. Levine, and A. Kanazawa. Amp: Adversarial motion priors
340 for stylized physics-based character control. *ACM Transactions on Graphics (TOG)*, 40(4):
341 1–20, 2021.
- 342 [38] H. Yao, Z. Song, B. Chen, and L. Liu. Controlvae: Model-based learning of generative con-
343 trollers for physics-based characters. *ACM Transactions on Graphics (TOG)*, 41(6):1–16,
344 2022.
- 345 [39] A. Escontrela, X. B. Peng, W. Yu, T. Zhang, A. Iscen, K. Goldberg, and P. Abbeel. Adver-
346 sarial motion priors make good substitutes for complex reward functions. In *2022 IEEE/RSJ*
347 *International Conference on Intelligent Robots and Systems (IROS)*, pages 25–32. IEEE, 2022.
- 348 [40] T. Li, Y. Zhang, C. Zhang, Q. Zhu, J. Sheng, W. Chi, C. Zhou, and L. Han. Learning terrain-
349 adaptive locomotion with agile behaviors by imitating animals. In *2023 IEEE/RSJ Interna-*
350 *tional Conference on Intelligent Robots and Systems (IROS)*, pages 339–345. IEEE, 2023.
- 351 [41] H. Shi, T. Li, Q. Zhu, J. Sheng, L. Han, and M. Q.-H. Meng. An efficient model-based approach
352 on learning agile motor skills without reinforcement. *arXiv preprint arXiv:2403.01962*, 2024.
- 353 [42] Q. Zhang, P. Cui, D. Yan, J. Sun, Y. Duan, A. Zhang, and R. Xu. Whole-body humanoid robot
354 locomotion with human reference. *arXiv preprint arXiv:2402.18294*, 2024.
- 355 [43] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville,
356 and Y. Bengio. Generative adversarial networks. *Communications of the ACM*, 63(11):139–
357 144, 2020.
- 358 [44] H. Zhang, S. Starke, T. Komura, and J. Saito. Mode-adaptive neural networks for quadruped
359 motion control. *ACM Transactions on Graphics (ToG)*, 37(4), jul 2018. ISSN 0730-0301.
360 [doi:10.1145/3197517.3201366](https://doi.org/10.1145/3197517.3201366).
- 361 [45] J. Schulman, S. Levine, P. Abbeel, M. Jordan, and P. Moritz. Trust region policy optimization.
362 In *International conference on machine learning*, pages 1889–1897. PMLR, 2015.
- 363 [46] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov. Proximal policy optimization
364 algorithms. *arXiv preprint arXiv:1707.06347*, 2017.
- 365 [47] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine. Soft actor-critic: Off-policy maximum entropy
366 deep reinforcement learning with a stochastic actor. In *International conference on machine*
367 *learning*, pages 1861–1870. PMLR, 2018.
- 368 [48] J. Martinez-Piazuelo, D. E. Ochoa, N. Quijano, and L. F. Giraldo. A multi-critic reinforcement
369 learning method: An application to multi-tank water systems. *IEEE Access*, 8:173227–173238,
370 2020. [doi:10.1109/ACCESS.2020.3025194](https://doi.org/10.1109/ACCESS.2020.3025194).
- 371 [49] S. Mysore, G. Cheng, Y. Zhao, K. Saenko, and M. Wu. Multi-critic actor learning: Teaching
372 rl policies to act with style. In *International Conference on Learning Representations*, 2021.
- 373 [50] P. Xu, X. Shang, V. Zordan, and I. Karamouzas. Composite motion learning with task
374 control. *ACM Transactions on Graphics (TOG)*, 42(4), jul 2023. ISSN 0730-0301. [doi:](https://doi.org/10.1145/3592447)
375 [10.1145/3592447](https://doi.org/10.1145/3592447).
- 376 [51] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polos-
377 sukhin. Attention is all you need. *Advances in neural information processing systems*, 30,
378 2017.

- 379 [52] H. Touvron, T. Lavril, G. Izacard, X. Martinet, M.-A. Lachaux, T. Lacroix, B. Rozière,
380 N. Goyal, E. Hambro, F. Azhar, et al. Llama: Open and efficient foundation language models.
381 *arXiv preprint arXiv:2302.13971*, 2023.
- 382 [53] A. Chowdhery, S. Narang, J. Devlin, M. Bosma, G. Mishra, A. Roberts, P. Barham, H. W.
383 Chung, C. Sutton, S. Gehrmann, et al. Palm: Scaling language modeling with pathways.
384 *Journal of Machine Learning Research*, 24(240):1–113, 2023.
- 385 [54] A. Brohan, N. Brown, J. Carbajal, Y. Chebotar, J. Dabis, C. Finn, K. Gopalakrishnan, K. Haus-
386 man, A. Herzog, J. Hsu, et al. Rt-1: Robotics transformer for real-world control at scale. *arXiv*
387 *preprint arXiv:2212.06817*, 2022.
- 388 [55] V. Makoviychuk, L. Wawrzyniak, Y. Guo, M. Lu, K. Storey, M. Macklin, D. Hoeller, N. Rudin,
389 A. Allshire, A. Handa, et al. Isaac gym: High performance gpu-based physics simulation for
390 robot learning. *arXiv preprint arXiv:2108.10470*, 2021.
- 391 [56] X. B. Peng, P. Abbeel, S. Levine, and M. Van de Panne. Deepmimic: Example-guided deep re-
392 inforcement learning of physics-based character skills. *ACM Transactions on Graphics (TOG)*,
393 37(4):1–14, 2018.
- 394 [57] Unitree Robotics. <https://www.unitree.com/en/go2>.
- 395 [58] D. Kang, J. Cheng, M. Zamora, F. Zargarbashi, and S. Coros. Rl+ model-based control: Using
396 on-demand optimal control to learn versatile legged locomotion. *IEEE Robotics and Automa-*
397 *tion Letters*, 2023.

398 A Appendix

399 A.1 Observation and Action Space

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

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

404 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
 405 each time step t , each keyframe \mathbf{k}^i is transformed spatially and temporally into a robot-centric view.
 406 Then, the goal observation is prepared by calculating the remaining time to goal $\hat{t}^i - t$ and the error
 407 to target goals ($\Delta \mathbf{g}_t^i$),

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

408 Here, $\Delta \mathbf{p}_b^i$ denotes the error between robot base position and keyframe position in the base coordi-
 409 nate 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
 410 and yaw angles, respectively, which are wrapped to $(-\pi, \pi]$.

411 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
 412 $\mathbf{x}_t^0 = (\mathbf{s}_t, \mathbf{0}, 0)$, 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
 413 keyframe encoding, the extra tokens can be masked to enable arbitrary number of goals. In addition,
 414 keyframes with a time over one second past the current time are also masked to avoid any long-term
 415 influence on reaching the future goals.

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

418 A.2 Reward Terms

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

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

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

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

424 To generate natural motion between the keyframes, we use AMP proposed by Peng et al. [12], which
 425 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	$\mathcal{K}(\max(\boldsymbol{\theta}_j - \boldsymbol{\theta}_{j,min}, \boldsymbol{\theta}_{j,max} - \boldsymbol{\theta}), 0.1, 0)$

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

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

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

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

429 and only activated when the corresponding timestep for that goal \hat{t}^i is reached in the episode. The
 430 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	$\Phi^i(\mathcal{K}(\ \boldsymbol{\theta}_j - \hat{\boldsymbol{\theta}}_j^i\ , 0.2, 0))$

431 A.3 Dataset Preparation

432 We use a database of motion capture from dogs introduced by Zhang et al. [44]. The motions are
 433 retargeted to the robot skeleton using inverse kinematics for the end-effectors' positions with some
 434 local offsets to compensate for the different proportions of the robot and dog. A subset of around
 435 20 minutes of data was used, removing the undesired motions such as smelling the ground, walking
 436 on slopes, etc. We augment this dataset with other motion clips animated by artists to include more
 437 diversity in the dataset. The frame rate is adjusted to that of the simulation, i.e. 50 frames per
 438 second.

439 A.4 Training Procedure

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

453 The meticulous sampling of target keyframes is critical for ensuring their feasibility and preventing
 454 them from impeding effective policy learning. We train the policy to handle a maximum number of
 455 keyframes, randomly selecting the actual number of keyframes for each episode. To avoid negative
 456 impacts on training, unused goals are masked when input into the transformer encoder. For stability,
 457 the episode does not terminate immediately after the last goal is reached; instead, it terminates
 458 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.

461 **A.5 Hardware Implementation Details**

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

470 **A.6 Future goal anticipation**

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

Table A3: Details of Keyframe Scenarios

Scenario	First Goal		Second Gaol	
	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)

472 **A.7 Training Hyperparameters**

473 Table A4 provides details of hyperparameters used for training.

Table A4: Summery 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