

Bi-Level Motion Imitation for Humanoid Robots

Anonymous Author(s)

Affiliation

Address

email

Abstract: Imitation learning from human motion capture (MoCap) data provides a promising way to train humanoid robots. However, due to differences in morphology, such as varying degrees of joint freedom and force limits, exact replication of human behaviors may not be feasible for humanoid robots. Consequently, incorporating physically infeasible MoCap data in training datasets can adversely affect the performance of the robot policy. To address this issue, we propose a bi-level optimization-based imitation learning framework that alternates between optimizing both the robot policy and the target MoCap data. Specifically, we first develop a generative latent dynamics model using a novel self-consistent auto-encoder, which learns sparse and structured motion representations while capturing desired motion patterns in the dataset. The dynamics model is then utilized to generate reference motions while the latent representation regularizes the bi-level motion imitation process. Experiments conducted on a simulated realistic humanoid robot demonstrate that our proposed method enhances the robot policy by modifying reference motions to be physically consistent.

Keywords: Humanoid Robots, Imitation Learning, Latent Dynamics Model

1 Introduction

The use of human motion capture (MoCap) data as reference trajectories offers a promising way to design powerful humanoid robot controllers [1, 2, 3, 4]. After appropriate motion retargeting these close-expert reference trajectories can be directly imitated by robots, reducing the need for extensive reward engineering typically required in reinforcement learning [5, 3]. Existing motion imitation works either learn the motion styles in a generative adversarial way [6, 2, 7, 3] or directly learn to track the provided motion trajectories [1, 8]. While the former method, based on generative adversarial imitation learning (GAIL) [9], avoids the exact definition of similarity between reference motions and robot trajectories, its min-max computational formulation usually suffers from unstable learning and sample inefficiency [10, 11]. The latter method, however, can also be problematic because the reference motion is often noisy and physically infeasible for realistic humanoid robots due to embodiment differences such as different force and joint limits between humans and robots [4]. Consequently, including such data may degenerate the policy learning of the robot [4].

The aforementioned issues arising from noisy and physically infeasible reference motion have been mainly studied in the field of motion retargeting [12, 13, 14]. For example, in order to create natural motions for various animated characters, researchers pursue retargeting the human MoCap motions into physically consistent motions of new characters, which in our case corresponds to humanoid robots. The common approach used in physics-based retargeting hinges on trajectory optimization with known dynamics of the target robot and constraints that arise from the reference trajectories [14, 15]. However, the resulting optimization problem is often complex and includes specific domain knowledge. There is therefore an emergent need for a learning-based method that does not rely on an explicit dynamics model while guaranteeing physical consistency at the same time. We address this need by proposing the Bi-Level Motion Imitation (BMI) framework.

Our method shares a similar bi-level optimization idea with differential optimal control [15] but does not need a prior dynamics model and human-specified constraints. Specifically, BMI first learns a generative latent dynamics model based on a novel self-consistent generative auto-encoder (SCAE) from the reference motions. SCAE regularizes normal auto-encoder training with a latent reconstruction error and captures the essential motion patterns with sparse and well-structured latent representations. This enables us to sample latent parameters and reconstruct new motions, which are used to train the humanoid robot policy (*pre-training* step). After pre-training, BMI further finetunes both the decoder and the robot policy as a bi-level optimization problem. In this way, the decoder learns to return reference motions that are physically consistent. At the same time, the robot further improves its policy by imitating updated reference motions. We constrain the decoder updates to ensure that the reconstructed motions stay close to the original motions in the latent space, which prevents the decoder from degenerating into trivial motions that are far from the desired motion patterns in the human MoCap data.

We evaluate BMI on the MIT Humanoid Robot [16] in simulation, where we imitate motions from human MoCap data. The experiments first show that the proposed SCAE-based latent dynamics model learns structured motion representations. In the subsequent pre-training, the improved latent representation learned by SCAE also enhances policy learning compared to the baseline latent dynamics model. Finally, our bi-level fine-tuning with latent space regularization updates the decoder to construct reference motions that are physically consistent for the robot and retain the original patterns at the same time. Our experiments show that the robot policy can be further improved by imitating the updated motions.

The key contributions of this paper can be summarized as follows: (i) We propose a self-consistent latent dynamics model that is able to learn sparse and structured representations for human motions. (ii) We propose a bi-level motion imitation framework to update the decoder and the robot policy at the same time, which enhances the generated motions with physical consistency and closeness to the original human MoCap trajectories. (iii) We evaluate our method on a humanoid robot and imitate up to 13 different motions with a single policy. The experiments highlight improved policy learning with the proposed latent dynamics model and bi-level motion imitation framework.

2 Related Work

We first discuss existing reference-based humanoid imitation learning methods. Methods addressing the problem of physically inconsistent reference motions are discussed subsequently.

Humanoid Motion Imitation Imitating from human MoCap data is an efficient way for humanoid robots to learn agile and natural-looking skills [1]. Recent works [7, 17] based on generative adversarial imitation learning (GAIL) [9, 2] in animation have succeeded in training humanoid robots to track various human motions using a large MoCap dataset such as AMASS [18]. Nonetheless, the success may be partially attributed to the unrealistic humanoid robot that is used. With up to 69 DoFs, unlimited force, and even assistive external forces [19], the simulated robot is massively overactuated and can, in principle, perfectly track the given reference motions. It is therefore unclear whether the approaches in animation [7, 17] can be transferred to more realistic robots. As the reference motions can be physically infeasible for robots, including them in the training dataset can result in sub-optimal mimicking behaviors or even complete failure in imitation [13]. Therefore, it is important to distinguish which motions are applicable to realistic humanoid robots. The authors from [20] train whole-body humanoid controllers that only replicate upper-body movements while the lower body is restricted to track a given forward velocity for the base. An alternation has been proposed in [4] where the infeasible motions are explicitly removed by a privileged simulated imitator. Fourier Latent Dynamics (FLD) [8] employs a fallback mechanism to replace the given reference motions with default motions when the reference is far from the training motions.

Physically Consistent Motion Retargeting Motion retargeting describes the process of mapping the human MoCap data to target robot configurations such that downstream motion imitation can be

performed. While common motion retargeting methods [21, 13] such as inverse kinematics-based methods can generate visually convincing motions, these motions could be physically infeasible for humanoid robots. In order to obtain physically consistent motion retargeting, existing methods are usually formulated as trajectory optimization problems constrained by robot dynamics [22, 12, 14, 15]. For instance, differential optimal control [15] alternatively optimizes the retargeting parameters with manually defined contact constraints and the robot trajectories based on the retargeting as a bi-level optimization problem. However, it is often tedious to model the complex robot dynamics and these methods are therefore hard to generalize across different robots. In contrast, our method is purely data-driven.

3 Preliminaries

Our method involves modifying a latent dynamics model, which maps the motions through an auto-encoder [23] into latent space and back, in order to generate motions for the robot that are physically consistent and at the same time close to the desired motion patterns in the original MoCap dataset. However, measuring the closeness between the original trajectory and the generated physically-consistent reference motion for the robot, is challenging [24]. We address this problem by introducing a structured motion representation and incentivizing closeness in the latent space. Our proposed latent dynamics model is inspired by FLD [8], a structured motion representation method that explicitly enforces the periodicity of motions in the latent space by transforming the learned latent representation into the frequency domain [25].

The structure of FLD is illustrated in Figure 7 in the appendix. We denote a given trajectory segment of length H in d -dimensional state space by $\tau_t = (s_{t-H+1}, \dots, s_t) \in \mathbb{R}^{d \times H}$, where t denotes time and s_t the state at time t . The trajectory segment τ_t represents the input to the auto-encoder, where the encoder embeds the original motion trajectory into a latent space with c channels, denoted by $z_t \in \mathbb{R}^{c \times H}$. In order to explicitly account for the periodicity of the motions, FLD builds on earlier work on Periodic Autoencoders (PAEs) [25] and includes a differentiable Fast Fourier Transform (FFT) layer. The FFT layer returns the frequency f_t , amplitude a_t , and offset b_t of the latent motion embeddings, while a separate phase ϕ_t is computed by an additional fully connected (FC) layer and an atan2 operation. This transformation is denoted as p :

$$z_t = \text{enc}(\tau_t), \quad (\phi_t, f_t, a_t, b_t) = p(z_t), \quad (1)$$

where $\phi_t, f_t, a_t, b_t \in \mathbb{R}^c$ and enc is the encoder. Particularly, FLD improves PAE with a multi-step forward prediction to approximate the subsequent latent vectors by unrolling the latent phase. For a local range of N subsequent trajectory segments $\{\tau_t, \tau_{t+1}, \dots, \tau_{t+N}\}$, we assume that the segments share the same latent parameters f_t, a_t, b_t while differing only in their phases ϕ_{t+i} . Furthermore, ϕ_{t+i} can be approximated by $\phi_{t+i} \approx \phi_t + i f_t \Delta_t$, where Δ_t denotes the step time. This results in,

$$\hat{z}'_{t+i} = \hat{p}(\phi_t + i f_t \Delta_t, f_t, a_t, b_t), \quad \hat{\tau}'_{t+i} = \text{dec}(\hat{z}'_{t+i}), \quad (2)$$

where dec is the decoder. We denote \hat{p} the embedding reconstruction process from the frequency domain,

$$\hat{z}_t = \hat{p}(\phi_t, f_t, a_t, b_t) = a_t \sin(2\pi(f_t \mathcal{T} + \phi_t)) + b_t, \quad (3)$$

where \mathcal{T} represents a known time window with H evenly spaced samples [25]. We note that $\hat{z}'_{t+i}, \hat{s}'_{t+i}$ are different from \hat{z}_t, \hat{s}_t as they are approximated by the multi-step forward prediction from the trajectory τ_t . This motivates the following loss function that is used in FLD,

$$L_{\text{FLD}}^N = \sum_{i=0}^N \alpha^i |\hat{\tau}'_{t+i} - \tau_{t+i}|^2, \quad (4)$$

where α is a decay factor and $|\cdot|$ denotes the Euclidean distance.

4 Method

The proposed method involves a three-stage training procedure. (i) In the first stage, we learn a generative latent dynamics model from the original MoCap data that has been kinematically retargeted

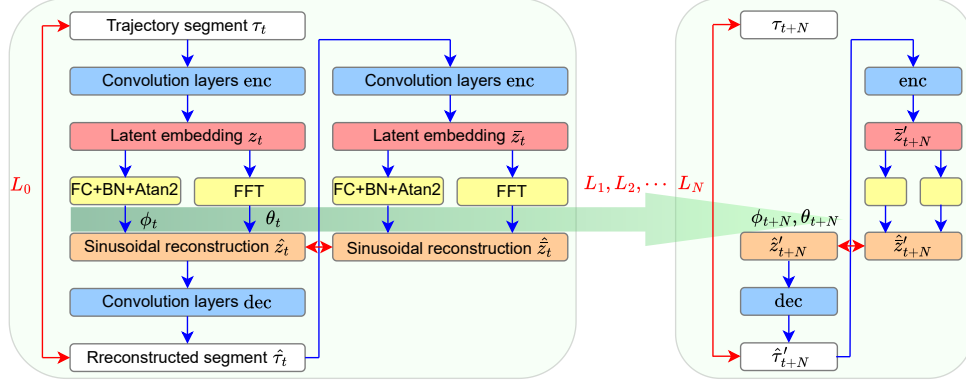


Figure 1: Structure of the proposed self-consistent auto-encoder (SCAE)

131 to the humanoid. We introduce a self-consistent auto-encoder trained using both reconstruction error
 132 and latent regularization, to capture the desired patterns embedded in the noisy kinematic motions
 133 more effectively. (ii) The second stage samples latent parameters encoded by the self-consistent
 134 dynamics model and then decodes these latent samples into the state space. The decoded states are
 135 used as the reference motions to pre-train the robot policy. (iii) We perform bi-level imitation by
 136 fine-tuning the policy and updating the decoder at the same time. Crucially, this bi-level optimiza-
 137 tion is constrained within the latent space, ensuring that the decoder generates motions that closely
 138 adhere to physics-based robot trajectories while preserving the original motion patterns intended for
 139 imitation. The following paragraphs explain the three-step procedure in detail.

140 4.1 Self-Consistent Latent Dynamics

141 Although FLD learns structured latent representations and shows accurate reconstruction, we find
 142 that the decoded motions with small reconstruction errors are not guaranteed to stay close to the
 143 original motions in the latent space. This means that the learned latent representation overfits to
 144 current data and is not robust to noise in the motions. In contrast, with our bi-level motion imitation
 145 framework, we introduce a latent representation that focuses on the general motion patterns instead
 146 of nuances and noise. This is important, since the nuances are likely to change when converted to
 147 be physically consistent in the fine-tuning step.

148 We address the above gap by a Self-Consistent Auto-Encoder (SCAE). Specifically, we propose
 149 to regularize FLD learning with a latent reconstruction error. A similar idea has been applied to
 150 VAE [26] but has not been investigated in deterministic auto-encoders for motion generation. Fig-
 151 ure 1 shows the structure of SCAE, where the reconstructed trajectory $\hat{\tau}_t$ is fed into the encoder
 152 again in order to obtain a reconstructed latent representation \hat{z}_t from the decoded motion $\hat{\tau}_t$. We
 153 retain the multi-step prediction in FLD and thus our SCAE training loss is

$$L_{\text{SCAE}}^N = \sum_{i=0}^N \alpha^i (|\hat{\tau}'_{t+i} - \tau_{t+i}|^2 + \beta |\hat{z}'_{t+i} - \hat{z}_t|^2), \quad (5)$$

154 where β is the coefficient of the latent reconstruction error and where we evaluate the loss on the
 155 entire dataset. The reconstructed latent representation \hat{z}'_{t+i} is computed by feeding the reconstructed
 156 trajectory $\hat{\tau}'_{t+i}$ into the encoder, the Fourier transform layer and the sinusoidal reconstruction layer.
 157 Note that $\hat{\tau}'_{t+i}$ is obtained by the multi-step forward prediction in Equation 2.

158 With a perfect decoder, the reconstructed motion $\hat{\tau}_t$ is exactly the same as the original motion τ_t
 159 leading to zero *latent reconstruction error* $|\hat{z}'_{t+i} - \hat{z}_t|^2$. However, this is usually not achievable.
 160 Although $|\hat{z}'_{t+i} - \hat{z}_t|^2$ generally decreases as the decoder learns to reconstruct the trajectory, our
 161 experiments show that $|\hat{z}'_{t+i} - \hat{z}_t|^2$ is not minimized when only optimizing the *motion reconstruc-*
 162 *tion error* $|\hat{\tau}'_{t+i} - \tau_{t+i}|^2$. In contrast, due to the latent reconstruction regularization, SCAE enforces
 163 the learned latent representation to be consistent with its decoded motions.

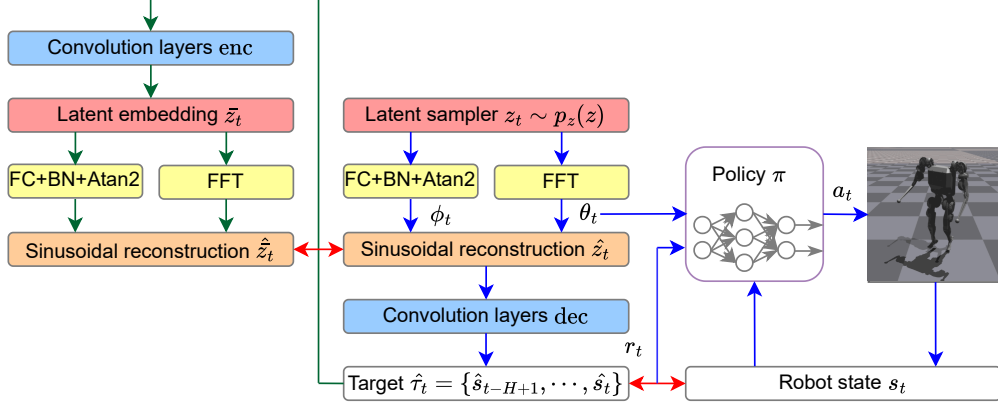


Figure 2: Structure of the proposed bi-level motion fine-tuning (BMI)

4.2 Pre-Training Policy

In this stage, we train our robot policy to track the given reference motions regardless of the feasibility of these motions as done in existing motion imitation works [1, 4]. In contrast to directly sampling trajectories from the original motion dataset to train the robot policy, we sample from the latent space of the SCAE and inform the robot policy with the sampled latent parameters as the target motion information. The self-consistent latent dynamics model provides two advantages compared to using the original datasets. (i) We can interpolate latent parameters to generate motion transitions and new motions, as discussed in FLD [8] and PAE [25]; (ii) We observe that a learned latent representation as the tracking goal for the robot is more concise with essential motion patterns and focuses less on motion nuances, which is beneficial for policy learning.

The policy pre-training procedure is illustrated in Figure 2 without the green arrow modules (these are only used in the next fine-tuning stage). For each episode, we sample a set of latent variables z_t from the pre-collected buffer $p_z(z)$ during SCAE training. We then obtain $(\phi_t, f_t, a_t, b_t) = p(z_t)$ by the following FC and FFT layers. Note that instead of taking the learned phase ϕ_t , we uniformly sample an initial phase variable $\phi_0 \in \mathbb{R}^c$ from a fixed range and update ϕ_t according to the latent dynamics in Equation 2,

$$\phi_t = \phi_{t-1} + f_{t-1}\Delta t, \quad \{f_t, a_t, b_t\} = \theta_t = \theta_{t-1}. \quad (6)$$

We maintain the same frequency f_t , amplitude a_t , and offset b_t for the episode. The latent variables are then used to reconstruct a motion trajectory

$$\{\hat{s}_{t-H+1}, \dots, \hat{s}_t\} = \hat{\tau}_t = \text{dec}(\hat{p}(f_t, a_t, b_t, \phi_t)), \quad (7)$$

where the most recent state \hat{s}_t serves as the target state to compute the robot tracking reward at the current timestep. The policy is learned using proximal policy optimization [27].

4.3 Bi-Level Fine-Tuning

This step ensures physical consistency of the reference motions generated by the decoder. Obtaining reference motions that are physically consistent is important as it facilitates policy learning and encourages the robot to learn a versatile set of skills, in particular when the humanoid robots are under-actuated and have restricted torque limits [7, 20, 4]. We propose to convert these unphysical motions into physically consistent ones by a bi-level fine-tuning to maximize the benefit of human MoCap data. This represents an important difference from recent works that address this problem by only tracking upper body movements [20] or filtering out the unlearnable motions [4].

Figure 2 shows the structure of our bi-level fine-tuning. In this stage, we alternatively optimize the policy π and the decoder dec while freezing the convolutional encoder enc and the FC, BN layers.

194 In this way, the decoder is encouraged to generate motions close to the robot trajectories, which are
 195 physically consistent by design. We further regularize the decoder optimization by constraining the
 196 generated motions to be close to the original motions in the latent space. This prevents the decoder
 197 from generating trivial motions by simply copying the robot, failing to improve the robot policy
 198 further. The bi-level optimization problem is formulated as,

$$\begin{aligned} \min_{\theta_{\text{dec}}} \mathbb{E}_{z_t \sim p_z(z), s_t \sim \pi_{\theta_{\pi}}} [|\hat{s}_t - s_t|^2 + \beta |\hat{z}_t - \hat{z}_t|^2], \\ \theta_{\pi}^* \in \arg \min_{\theta_{\pi}} \mathbb{E}_{z_t \sim p_z(z), s_t \sim \pi_{\theta_{\pi}}} [|\hat{s}_t - s_t|^2], \end{aligned} \quad (8)$$

199 where θ_{dec} denotes the parameters of the decoder and $\pi_{\theta_{\pi}}$ the robot policy with parameters θ_{π} .
 200 With the proposed regularized bi-level motion imitation, the decoder is updated to generate motions
 201 physically consistent with the robot while retaining the desired motion patterns in the dataset. As a
 202 result, we observed that the robot further improves the policy during this fine-tuning step.

203 5 Experiments

204 We evaluate BMI on the MIT humanoid robot [16] in Isaac Gym [28] while keeping the joint and
 205 force limits unchanged. We extend the dataset from FLD [8] by including four additional difficult
 206 motions, i.e., *jump*, *kick*, *spin-kick*, and *cross-over* [1]. In total, we have trajectories from 13 different
 207 motions in the dataset. In our experiments, we first show the reconstruction accuracy and analyze
 208 the motion representations learned by our latent dynamics model. We then compare the motion
 209 performance of the pre-trained policies based on SCAE and FLD, with the BMI fine-tuned policy.

210 5.1 Analysis of Learned Latent Dynamics Model

211 **Motion and Latent Reconstruction** Figure 3b shows that our method and FLD can reconstruct
 212 the original motions with comparable accuracy. However, our method with explicit self-consistency
 213 constraints achieves significantly lower latent reconstruction error, i.e., $|\hat{z}'_{t+i} - \hat{z}'_{t+i}|^2$, as shown in
 214 Figure 3a. Samples of reconstructed motions can be found in Figure 9 in the appendix, where both
 215 methods accurately reconstruct the original motions. The proposed self-consistent regularization
 improves the latent reconstruction without sacrificing the motion reconstruction accuracy.

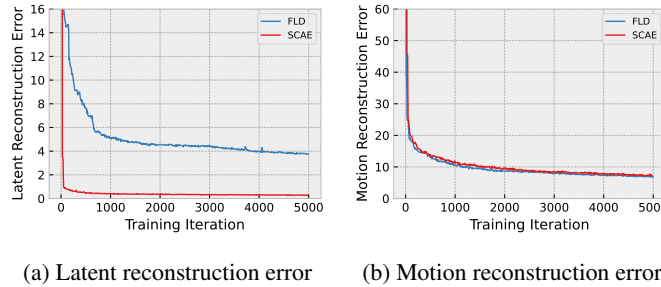


Figure 3: Reconstruction error during training: (a) The reconstruction error of latent embeddings.
 (b) The reconstruction error of the original motion states.

216

217 **Visualization of Learned Latent Manifold** We visualize the learned latent amplitude f_t and la-
 218 tent phase ϕ_t in eight latent channels, computed as in Equation 1, for four motions *run*, *jog*, *step fast*,
 219 *jump* in Figure 4, where each row denotes the same motion. Thanks to the latent regularization, our
 220 method learns a much sparser representation than FLD, as SCAE takes fewer frequency components
 221 to reconstruct the same motions with most channels' amplitudes around zero. This observation is
 222 consistent for all 13 different motions, as shown in Figure 8 in the appendix. Such a sparse repre-
 223 sentation helps our latent dynamics model capture the most relevant patterns while discarding the
 224 motion nuances, which is important for maintaining the desired motion patterns but modifying the
 225 motion details to be physically consistent for our robots.

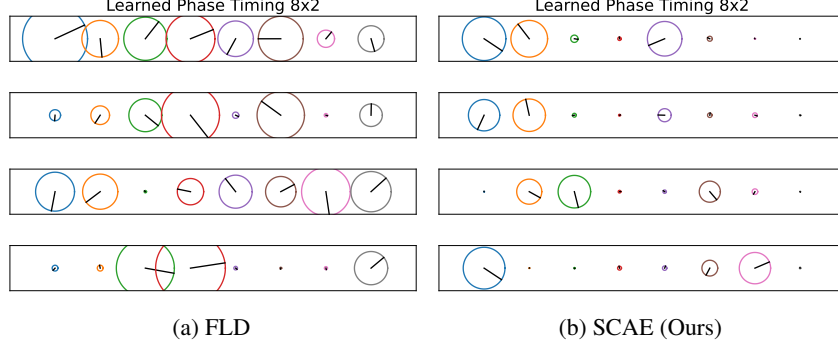


Figure 4: The figure displays the learned latent phases of four motions. Each row corresponds to one motion. Each circle represents a latent channel where the radius is the amplitude and the black bar is the phase timing. Compared to FLD, SCAE takes fewer frequency components and lower amplitudes to represent the same motion.

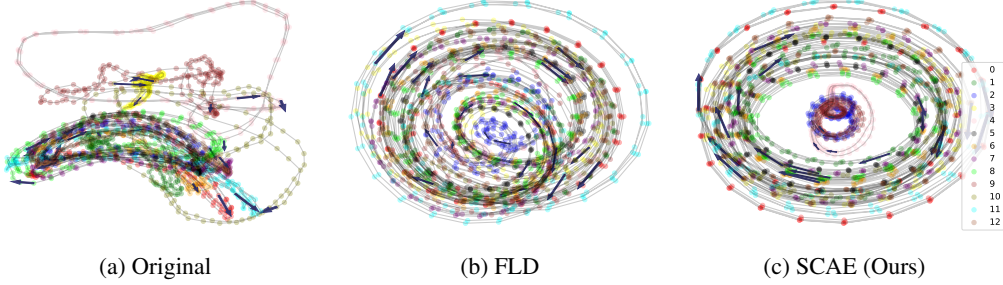


Figure 5: The figure shows the latent manifolds for 13 motions. Each color corresponds to a trajectory segment from a motion type. The arrows denote the motion evolution direction. The manifold induced by SCAE shows consistent structures across different motions.

In Figure 5, we further compare the latent structure induced by SCAE with that by FLD, where Figure 5a visualizes the principal components of the original motions for comprehensive analysis. Notably, SCAE demonstrates the most consistent structure across 13 different motions. The circles connecting points with the same color represent the primary period of individual motions and each point denotes a trajectory segment. The radius of a rough circle means that the high-level features throughout a motion can be constant, such as velocity, frequency, etc. The well-shaped latent manifolds learned by SCAE show that our method successfully captures essential motion patterns.

5.2 Performance of Pre-Trained Policy Based on SCAE

We find, perhaps surprisingly, that without further fine-tuning SCAE improves robot policy learning in the pre-training stage. Note that since the target reference motions are noisy and sometimes physically inconsistent for the robot, the commonly used mean square error from the reference motions is not an ideal performance metric anymore. Although both our SCAE-based and FLD-based pre-training policies can achieve similar tracking errors, we observe that the SCAE-based policy shows better capacity to do challenging motions such as *jump* and *kick*. In Table 1, we find that SCAE achieves the highest hang time percentage in 10 *jump* trajectories, which is computed as the proportion of time when both feet are off the ground. Note that each trajectory includes multiple *jump* trials. Figure 6b also shows that SCAE achieves better stability while kicking. We hypothesize that the policy improvement in pre-training is due to the change of latent parameterizations used to inform the policy. SCAE learns a sparser representation that makes policy learning easier. More results on the other motions can be found in the supplementary videos.

Table 1: Results on two selected challenging motions: *kick* and *jump*.

Motion (Metric)\Algo.	FLD	SCAE(Ours)	BMI(Ours)
Kick (Time (%))	64.4	61.2	71.3
Kick (Height (m))	0.157	0.152	0.164
Jump (Time (%))	32.5	36.2	35.2

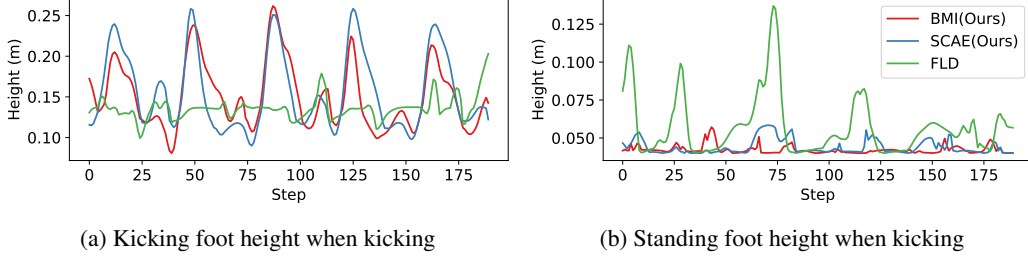


Figure 6: Comparison on the challenging *kick* task: The left figure shows the height of the kicking foot during one *kick* trajectory with multiple trials, where both SCAE and BMI outperform FLD in each kick (one mode of the curve). The right figure shows the height of the standing foot where BMI and SCAE are more stable with a lower height of the standing foot.

246 5.3 Bi-Level Fine-tuned Policy

247 Although the pre-trained policy demonstrates significant tracking ability on most feasible motions,
 248 as shown in the supplementary video, some physically inconsistent references can dominate the loss
 249 leading to a sub-optimal policy for certain motions. We apply the bi-level fine-tuning to alleviate
 250 this problem and further improve our policy. Table 1 shows that BMI achieves the longest kicking
 251 time, defined as the percentage of time the kicking foot is off the ground. Moreover, thanks to our
 252 bi-level optimization modifying the target motions, our robot learns to further stabilize its standing
 253 foot when performing kicks, as illustrated in Figure 6b. In contrast, the baseline policy tends to
 254 stagger the standing foot when jumping. The experiments (see also video attachment) therefore
 255 confirm our hypothesis that by updating the decoder the robot policy can be further improved.

256 6 Limitations

257 While the proposed bi-level motion imitation framework alleviates problems arising from physically
 258 inconsistent reference motions, the approach relies on a decent robot policy in the pre-training stage.
 259 When the human MoCap dataset consists of impractical motions for the humanoid robot and exhibits
 260 large perturbations, it is still under-investigated to what extent our method can recover physically
 261 consistent motions while maintaining the general motion patterns. Moreover, since the given refer-
 262 ences obtained by motion retargeting from human MoCap data are not the optimal targets for robot
 263 imitation, the choice of metric to quantify robot tracking performance is an open question.

264 7 Conclusion

265 This paper presents BMI, a novel bi-level motion imitation framework that minimizes the robot
 266 tracking error by alternatively optimizing the robot policy and the motion generation model while
 267 being regularized by latent space constraints. Our proposed self-consistent auto-encoder captures
 268 the essential motion patterns with sparse and well-structured latent representations, which provides
 269 a reliable anchor to regularize the decoder to stay close to the desired motion patterns in the dataset.
 270 In contrast to existing optimal control methods, BMI addresses the difficulty of including physically
 271 inconsistent reference motions in a purely data-driven way and is scalable to large-scale human
 272 MoCap datasets. Our experiments on the realistic MIT humanoid robot show that BMI not only
 273 improves the pre-trained policy on challenging tasks but also further stabilizes the learned motions.

References

- [1] X. B. Peng, P. Abbeel, S. Levine, and M. Van de Panne. DeepMimic: Example-guided deep reinforcement learning of physics-based character skills. *ACM Transactions On Graphics (TOG)*, 37(4):1–14, 2018.
- [2] X. B. Peng, Z. Ma, P. Abbeel, S. Levine, and A. Kanazawa. AMP: Adversarial motion priors for stylized physics-based character control. *ACM Transactions on Graphics (TOG)*, 40(4): 1–20, 2021.
- [3] A. Tang, T. Hiraoka, N. Hiraoka, F. Shi, K. Kawaharazuka, K. Kojima, K. Okada, and M. Inaba. HumanMimic: Learning natural locomotion and transitions for humanoid robot via Wasserstein adversarial imitation. *arXiv preprint arXiv:2309.14225*, 2023.
- [4] T. He, Z. Luo, W. Xiao, C. Zhang, K. Kitani, C. Liu, and G. Shi. Learning human-to-humanoid real-time whole-body teleoperation. *arXiv preprint arXiv:2403.04436*, 2024.
- [5] J. Koenemann, F. Burget, and M. Bennewitz. Real-time imitation of human whole-body motions by humanoids. In *Proceedings of the 2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2806–2812, 2014.
- [6] J. Merel, Y. Tassa, D. TB, S. Srinivasan, J. Lemmon, Z. Wang, G. Wayne, and N. Heess. Learning human behaviors from motion capture by adversarial imitation. *arXiv preprint arXiv:1707.02201*, 2017.
- [7] Z. Luo, J. Cao, K. Kitani, W. Xu, et al. Perpetual humanoid control for real-time simulated avatars. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 10895–10904, 2023.
- [8] C. Li, E. Stanger-Jones, S. Heim, and S. Kim. FLD: Fourier latent dynamics for structured motion representation and learning. *arXiv preprint arXiv:2402.13820*, 2024.
- [9] J. Ho and S. Ermon. Generative adversarial imitation learning. In *Proceedings of the Advances in Neural Information Processing Systems*, 2016.
- [10] M. Orsini, A. Raichuk, L. Hussenot, D. Vincent, R. Dadashi, S. Girgin, M. Geist, O. Bachem, O. Pietquin, and M. Andrychowicz. What matters for adversarial imitation learning? In *Proceedings of the Advances in Neural Information Processing Systems*, 2021.
- [11] D. Jung, H. Lee, and S. Yoon. Sample-efficient adversarial imitation learning. *Journal of Machine Learning Research*, 25(31):1–32, 2024.
- [12] G. Bin Hammam, P. M. Wensing, B. Dariush, and D. E. Orin. Kinodynamically consistent motion retargeting for humanoids. *International Journal of Humanoid Robotics*, 12(04), 2015.
- [13] T. Yoon, D. Kang, S. Kim, M. Ahn, S. Coros, and S. Choi. Spatio-temporal motion retargeting for quadruped robots. *arXiv preprint arXiv:2404.11557*, 2024.
- [14] M. Al Borno, L. Righetti, M. J. Black, S. L. Delp, E. Fiume, and J. Romero. Robust physics-based motion retargeting with realistic body shapes. *Computer Graphics Forum*, 37:81–92, 2018.
- [15] R. Grandia, F. Farshidian, E. Knoop, C. Schumacher, M. Hutter, and M. Bächer. DOC: Differentiable optimal control for retargeting motions onto legged robots. *ACM Transactions on Graphics (TOG)*, 42(4):1–14, 2023.
- [16] M. Chignoli, D. Kim, E. Stanger-Jones, and S. Kim. The MIT humanoid robot: Design, motion planning, and control for acrobatic behaviors. In *Proceedings of the 2020 IEEE-RAS 20th International Conference on Humanoid Robots (Humanoids)*, 2021.

- 317 [17] Z. Luo, J. Cao, J. Merel, A. Winkler, J. Huang, K. Kitani, and W. Xu. Universal humanoid
318 motion representations for physics-based control. *arXiv preprint arXiv:2310.04582*, 2023.
- 319 [18] N. Mahmood, N. Ghorbani, N. F. Troje, G. Pons-Moll, and M. J. Black. AMASS: Archive of
320 motion capture as surface shapes. In *Proceedings of the IEEE/CVF international conference*
321 *on computer vision*, pages 5442–5451, 2019.
- 322 [19] Y. Yuan and K. Kitani. Residual force control for agile human behavior imitation and extended
323 motion synthesis. In *Proceedings of the Advances in Neural Information Processing Systems*,
324 2020.
- 325 [20] X. Cheng, Y. Ji, J. Chen, R. Yang, G. Yang, and X. Wang. Expressive whole-body control for
326 humanoid robots. *arXiv preprint arXiv:2402.16796*, 2024.
- 327 [21] K.-J. Choi and H.-S. Ko. Online motion retargeting. *The Journal of Visualization and Com-*
328 *puter Animation*, 11(5):223–235, 2000.
- 329 [22] S. Tak and H.-S. Ko. A physically-based motion retargeting filter. *ACM Transactions on*
330 *Graphics (TOG)*, 24(1):98–117, 2005.
- 331 [23] G. Alain and Y. Bengio. What regularized auto-encoders learn from the data-generating distri-
332 bution. *The Journal of Machine Learning Research*, 15(1):3563–3593, 2014.
- 333 [24] C. Li, M. Vlastelica, S. Blaes, J. Frey, F. Grimmering, and G. Martius. Learning agile skills
334 via adversarial imitation of rough partial demonstrations. In *Proceedings of the Conference on*
335 *Robot Learning*, pages 342–352, 2023.
- 336 [25] S. Starke, I. Mason, and T. Komura. DeepPhase: Periodic autoencoders for learning motion
337 phase manifolds. *ACM Transactions on Graphics (TOG)*, 41(4):1–13, 2022.
- 338 [26] T. Cemgil, S. Ghaisas, K. Dvijotham, S. Goyal, and P. Kohli. The autoencoding variational
339 autoencoder. In *Proceedings of the Advances in Neural Information Processing Systems*, 2020.
- 340 [27] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov. Proximal policy optimization
341 algorithms. *arXiv preprint arXiv:1707.06347*, 2017.
- 342 [28] N. Rudin, D. Hoeller, P. Reist, and M. Hutter. Learning to walk in minutes using massively
343 parallel deep reinforcement learning. In *Proceedings of the Conference on Robot Learning*,
344 pages 91–100, 2022.
- 345 [29] J. Tan, K. Liu, and G. Turk. Stable proportional-derivative controllers. *IEEE Computer Graph-*
346 *ics and Applications*, 31(4):34–44, 2011.

A Appendix

Contents

A.1	Structure of FLD	11
A.2	Pseudo Code of BMI	11
A.3	Experiment Settings	12
A.3.1	Dataset	12
A.3.2	State and Action Spaces	13
A.3.3	SCAE Training	13
A.3.4	Policy Training	14
A.3.5	BMI Training	15
A.4	More Experiment Results	16
A.4.1	More Results on Latent Dynamics Model Learning	16
A.4.2	Visualization of Learned Policy	17

A.1 Structure of FLD

The structure of FLD follows PAE [25] using an auto-encoder to learn a generative dynamics model, where the encoder and the decoder are composed of 1D convolutional layers. In order to enforce the periodicity in the latent manifolds, PAE parameterized each latent channel as a sinusoidal function where the amplitude, frequency, and offset are computed by a differentiable Fast Fourier Transform layer while the phase is determined with a fully connected layer followed by an Atan2 operation. Inspired by the observation that the learned latent frequency, amplitude, and offset by PAE stay nearly constant along the trajectories, FLD improves PAE by combining the structure with a multi-step prediction step as in Equation 4.

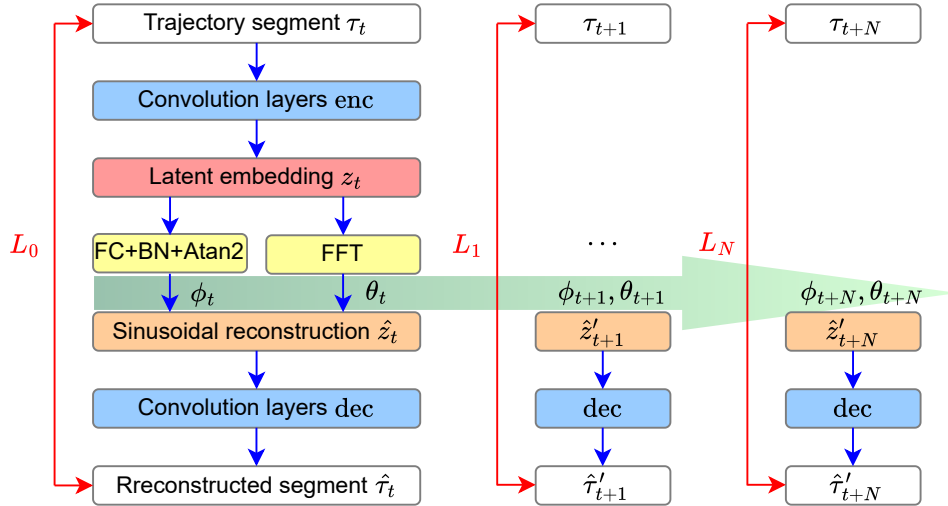


Figure 7: Multi-step forward prediction structure of FLD.

A.2 Pseudo Code of BMI

Algorithm 1 shows the details of BMI training.

Algorithm 1 Bi-Level Motion Imitation (BMI)

Input: SCAE encoder enc and decoder dec , latent parameters of the original motions $p_z(z)$,
Pre-trained policy π_{θ_π} based on SCAE, initial buffer \mathcal{D}
for $k = 1$ **to** K **do**
 Policy Learning:
 for $i = 1$ **to** M_1 **do**
 Sample latent targets $z_i \sim p_z(z)$
 Extract the target states $\{\hat{s}_{t-H+1}, \dots, \hat{s}_t\} = \hat{\tau}_t = \text{dec}(\hat{p}(f_t, a_t, b_t, \phi_t))$
 Rollout robot trajectory $\tau_i \sim p(\pi_{\theta_\pi}, \text{dec}, p_z)$
 Collect the trajectory and latent parameter pairs in the buffer $\mathcal{D} = \{(z_i, \tau_i) | i \sim M_1\}$
 Update robot policy π_{θ_π} with PPO or another RL algorithm according to the bottom ob-
 jective in Equation 8
 end for
 Decoder dec Update:
 for $i = 1$ **to** M_2 **do**
 Sample latent parameters and robot trajectories from \mathcal{D}
 Update the decoder dec according to the upper objective in Equation 8
 end for
end for

A.3 Experiment Settings

In this section, we provide more detailed experiment settings. We first introduce our dataset and then explain the state and action spaces used in SCAE and the policy. Finally, we list the architectures and hyper-parameters used in both the dynamics model learning and policy learning.

A.3.1 Dataset

We use the same dataset as FLD [8] which was originally released in DeepMimic [1]. The human MoCap data were manually processed and retargeted to the humanoid robot. Note that even with careful kinematic retargeting the reference motions can be physically inconsistent to the robot dynamics. Our dataset consists of 13 different motions: *run, jog, step fast, jump, spin-kick, back, side left, jog slow, side right, cross-over, kick, stride, step* and each motion has 10 trajectories collected from different demonstrations. In each trajectory of length 240 steps, the demonstrator performs multiple trials of the same action. For example, in one *kick* trajectory, the demonstrator may continuously kick 5 times as shown in Figure 6a. In total, we have $13 \times 10 \times 240 = 31200$ data points. Each data point corresponds to a state vector of length 52, where the elements are listed in the following Table 2.

Table 2: Elements of one data point (step) in the dataset

Entry	Symbol	Dimensions
base position	p_b	0:3
base rotation	q_b	3:7
base linear velocity	v	7:10
base angular velocity	w	10:13
projected gravity	g	13:16
joint positions	q	16:34
joint velocity	\dot{q}	34:52

Note that FLD experiments only on nine motions: *run, jog, step fast, back, side left, jog slow, side right, stride, step*, referred to as *normal motions*, which present a mild difficulty for the robot to track. However, our experiments include an additional four motions, *jump, spin-kick, cross-over, kick*, that are significantly more challenging. FLD fails to learn these complex motions satisfactorily without a specifically designed reward function tailored to each individual motion, while our methods show improved performance on the challenging *kick* and *jump* with unchanged reward design.

A.3.2 State and Action Spaces

In this section, we introduce the state space used in the latent dynamics model and the observation and action spaces for the robot policy.

State Space of Latent Dynamics Model The state space used in the latent dynamics model is composed of the linear and angular velocities of the robot base v, w in the robot frame, measurement of the gravity vector g in the robot frame, and joint positions q as in Table 3. Note that we use the same setting for both FLD and SCAE.

Table 3: Elements of the state space for latent dynamics model

Entry	Symbol	Dimensions
base linear velocity	v	0:3
base angular velocity	w	3:6
projected gravity	g	6:9
joint positions	q	9:27

Observation Space of Robot Policy In addition to the state information used in the latent dynamics model, the robot observes extra information such as joint velocities \dot{q} and its last action a' . Moreover, we provide the latent parameters to the robot as the target motion information. Therefore, the observation space is shown as Table 4. Note that we apply domain randomization to the policy training including the observation noises, mass noises, and pushing noises as used in FLD [8].

Table 4: Elements of the observation space for robot policy

Entry	Symbol	Dimensions	Noise level
base linear velocity	v	0:3	0.2
base angular velocity	w	3:6	0.05
projected gravity	g	6:9	0.05
joint positions	q	9:27	0.01
joint velocities	\dot{q}	27:45	0.75
last actions	a'	45:63	0.0
latent phase	$\sin \phi$	63:71	0.0
latent phase	$\cos \phi$	71:79	0.0
latent frequency	f	79:87	0.0
latent amplitude	a	87:95	0.0
latent offset	b	95:103	0.0

Action Space of Robot Policy The action space of our robot is of 18 dimensions which represent the target positions of 18 joints in the robot. An underlying PD controller [29] is used to compute the torques to drive each joint. The PD gains are set to (30.0, 5.0) for lower body joints and (40.0, 5.0) for upper body joints, respectively.

A.3.3 SCAE Training

We introduce first the architecture of neural networks used in SCAE, which is the same as FLD. Then we list the hyper-parameters for training the latent dynamics model.

Architecture of SCAE SCAE shares the same architecture as FLD. The architectures of the encoder enc and decoder dec are shown in Table 5. BN denotes batch normalization and ELU represents the exponential linear unit.

Hyper-Parameters for SCAE Training SCAE uses the same hyper-parameters for training FLD as in Table 6. The extra coefficient of the latent reconstruction regularization used in SCAE, i.e., β in Equation 5, is set to 1. Adam is used as the optimizer for training the latent dynamics model.

Table 5: Architecture of the neural networks used in SCAE

Network	Layer	Output size	Kernel size	Normalization	Activation
encoder	Conv1d	64x51	51	BN	ELU
	Conv1d	64x51	51	BN	ELU
	Conv1d	8x51	51	BN	ELU
phase encoder	Linear	8x2	–	BN	Atan2
decoder	Conv1d	64x51	51	BN	ELU
	Conv1d	64x51	51	BN	ELU
	Conv1d	27x51	51	BN	ELU

Table 6: Hyper-parameters of SCAE training

Parameter	Symbol	Value
step time seconds	Δt	0.02
max training iterations	–	5000
learning rate	–	0.0001
weight decay	–	0.0005
learning epochs	–	5
mini-batches	–	4
latent channels	c	8
trajectory segment length	H	51
multi-step prediction length	N	50
propagation decay	α	1.0

417 A.3.4 Policy Training

418 **Architecture of Policy & Value function** The neural network architectures of the learning policy
 419 π and the value function V used in PPO are shown in Table 7.

Table 7: Architecture of the neural networks used in policy training

Network	Type	Hidden	Output size	Activation
policy π	MLP	128, 128, 128	18	ELU
value function V	MLP	128, 128, 128	1	ELU

420 **Hyper-Parameters for Policy Training** We use Adam as the optimizer for the policy and value
 421 function with an adaptive learning rate with a KL divergence target of 0.01. The policy runs at 50
 422 Hz. We parallize 4096 environments in Isaac Gym to collect samples. The summary of the policy
 training hyper-parameters can be found in Table 8.

Table 8: Hyper-parameters of policy training

Parameter	Symbol	Value
step time seconds	Δt	0.02
max training iterations	–	3000
max episode time seconds	–	20
learning rate	–	0.001
steps per iteration	–	24
learning epochs	–	5
mini-batches	–	4
KL divergence target	–	0.01
discount factor	γ	0.99
clip range	ϵ	0.2
entropy coefficient	–	0.01
parallel training environments	–	4096

424 **Reward Function for Policy Training** The reward function used to train the robot policy consists
 425 of two categories $r = r^T + r^R$, where r^T denotes the tracking rewards and r^R represents the
 426 regularization rewards. The tracking reward calculates the weighted sum of individual rewards on
 427 each dimension bounded in $[0, 1]$ with their weights in Table 9,

$$r^T = w_v r_v + w_w r_w + w_g r_g + w_{q_{\text{leg}}} r_{q_{\text{leg}}} + w_{q_{\text{arm}}} r_{q_{\text{arm}}}. \quad (9)$$

428 The reward of each dimension is generally formulated as,

$$r_i = e^{-\sigma_i |\hat{d}_i - d_i|^2}, \quad (10)$$

429 where i denotes the i_{th} dimension. d_i denotes the target value of this dimension while \hat{d}_i represents
 430 the reconstructed value. σ_i is a temperature factor for each reward and can be found in Table 10.

Table 9: Weights of the tracking rewards

Weight	w_v	w_w	w_g	$w_{q_{\text{leg}}}$	$w_{q_{\text{arm}}}$
Value	1.0	1.0	1.0	1.0	1.0

Table 10: Temperature factors of the tracking rewards

Weight	σ_v	σ_w	σ_g	$\sigma_{q_{\text{leg}}}$	$\sigma_{q_{\text{arm}}}$
Value	0.2	0.2	1.0	1.0	1.0

431 The regularization reward is formulated as Equation 11, where the weights can be found in Table 11
 432 and each term is detailed as follows.

$$r^R = w_{\text{ar}} r_{\text{ar}} + w_{\text{qa}} r_{\text{qa}} + w_{\text{qT}} r_{\text{qT}} \quad (11)$$

433 Action rate

$$r_{\text{ar}} = |a' - a|^2, \quad (12)$$

434 where a' and a denote the previous and current actions.

435 Joint acceleration

$$r_{\text{qa}} = \left| \frac{\dot{q}' - \dot{q}}{\Delta t} \right|^2, \quad (13)$$

436 where \dot{q}' and \dot{q} denote the previous and current joint velocity. Δt represents the step time interval.

437 Joint torque

$$r_{\text{qT}} = |T|^2, \quad (14)$$

438 where T denotes the joint torques.

Table 11: Weights of the regularization rewards

Weight	w_{ar}	w_{qa}	w_{qT}
Value	-0.01	-2.5×10^{-7}	-1.0×10^{-5}

439 A.3.5 BMI Training

440 In the bi-level fine-tuning process, we retain most of the hyperparameters from the pre-training stage.
 441 Notable exceptions include the following: (i) We set a lower learning rate for the decoder update
 442 compared to the rate used in SCAE training. (ii) To align the magnitudes of the latent reconstruction
 443 loss and the motion reconstruction loss in Equation 8, we increase the coefficient β to 200. The key
 444 hyper-parameters used in BMI are summarized in Table 12. These hyper-parameters may be further
 445 tuned for improved results. As this is an initial study of bi-level fine-tuning, we tested only a limited
 446 number of hyper-parameter configurations in our experiments.

Table 12: Hyper-parameters of BMI fine-tuning

Parameter	Symbol	Value
coefficient of latent reconstruction loss	β	200
learning rate for decoder	–	0.00001
number of mini-batch for decoder	–	2
max training iteration	–	50
epochs for decoder	–	1
steps per iteration	–	24
parallel training environments	–	4096

447 A.4 More Experiment Results

448 We show more experiment results in this section, including experiments for both the latent dynamics
 449 model learning and the policy learning.

450 A.4.1 More Results on Latent Dynamics Model Learning

451 Figure 8 compares the learned latent phases across all the 13 motions with different methods. We
 452 observe that our method SCAE consistently achieves sparser representations than FLD with fewer
 453 frequency components and lower amplitudes.

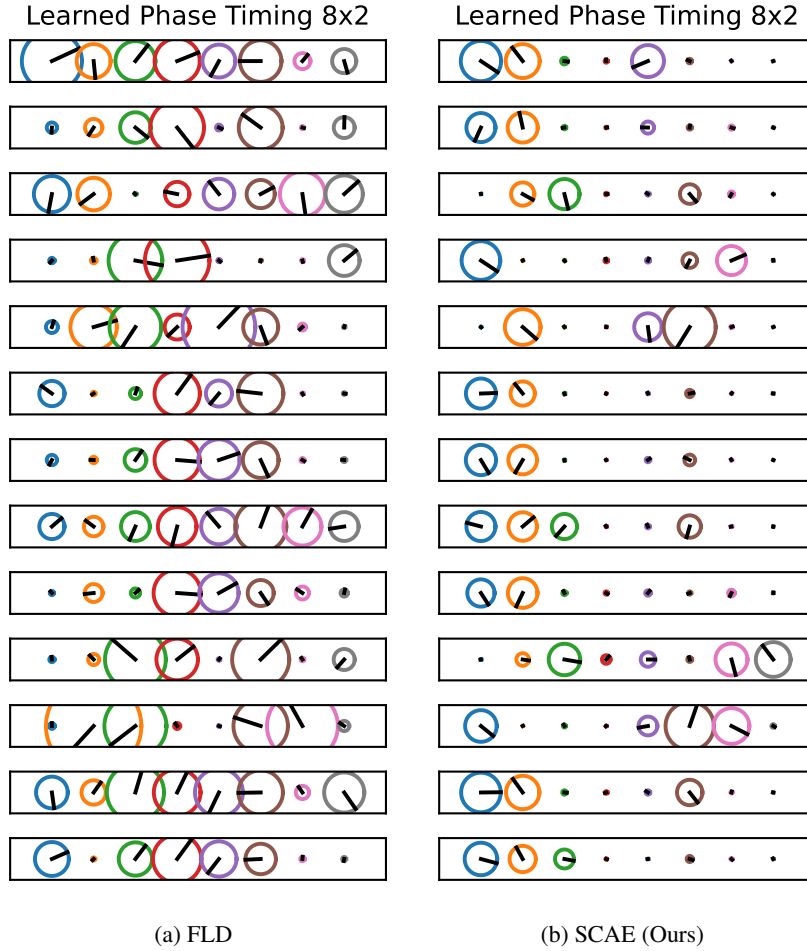


Figure 8: Learned latent phases of 13 different motions. From top to bottom, the motions are: *run*, *jog*, *step fast*, *jump*, *spin-kick*, *back*, *side left*, *jog slow*, *side right*, *cross-over*, *kick*, *stride*, *step*.

SCAE learns sparse and well-shaped latent representations. Nonetheless, it retains accurate motion reconstruction as FLD. As shown in Figure 9, both FLD and SCAE accurately reconstruct the motions, which is also validated by the training loss in Figure 3b.

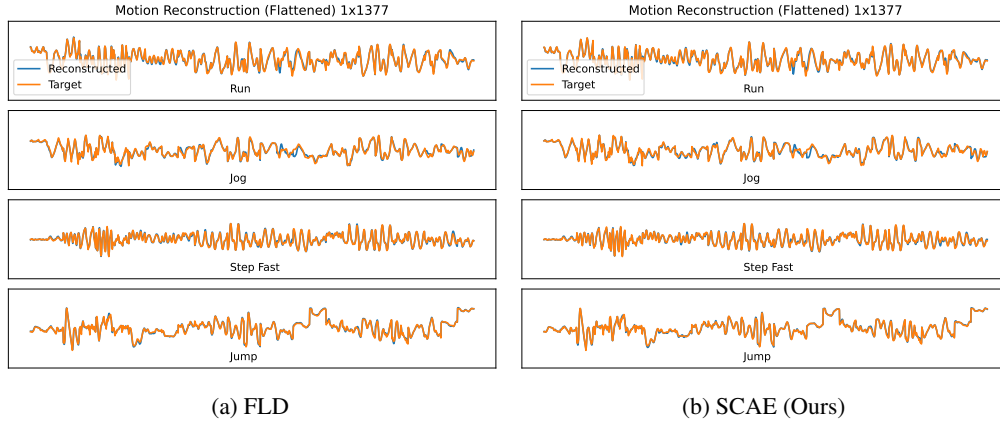


Figure 9: Motion reconstruction performance.

A.4.2 Visualization of Learned Policy

We visualize the motions learned by BMI. In addition to normal motions, such as *stride* in Figure 10a, which can be effectively learned by FLD, BMI successfully acquires two challenging motions *kick* and *jump* in which FLD fails. Figure 11a shows that BMI policy can naturally lift the kicking foot while maintaining the stability of the standing foot. Similarly, Figure 11b illustrates that the robot successfully jumps, with both feet leaving the ground.

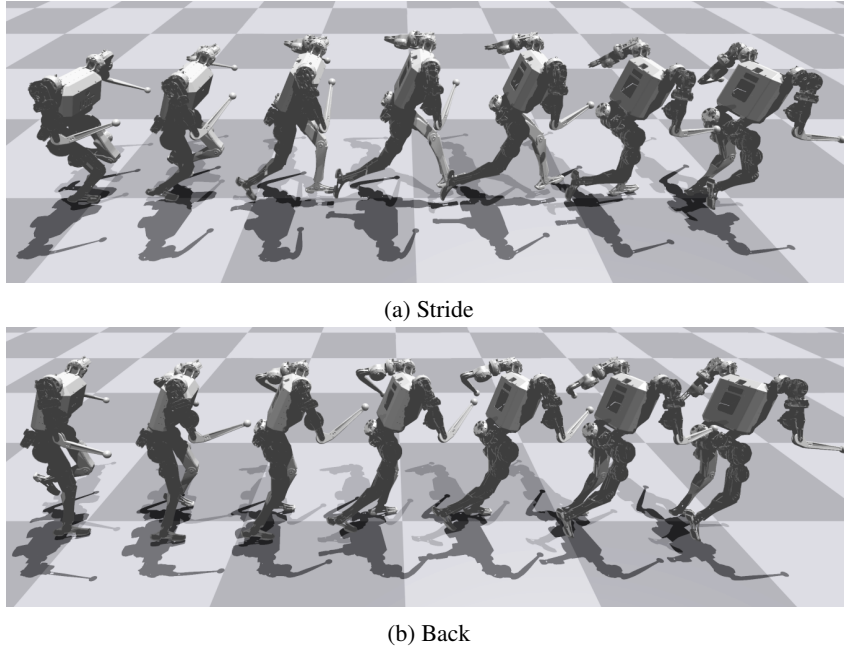
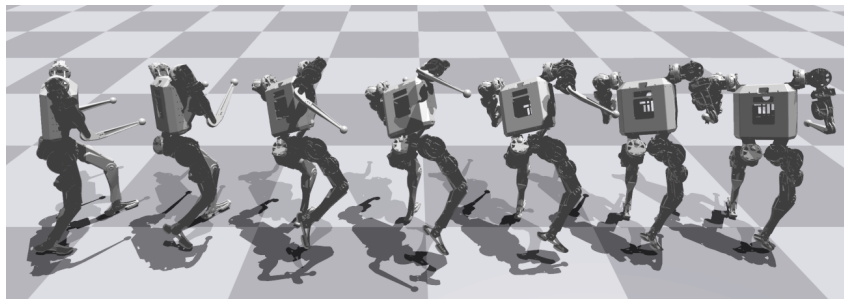
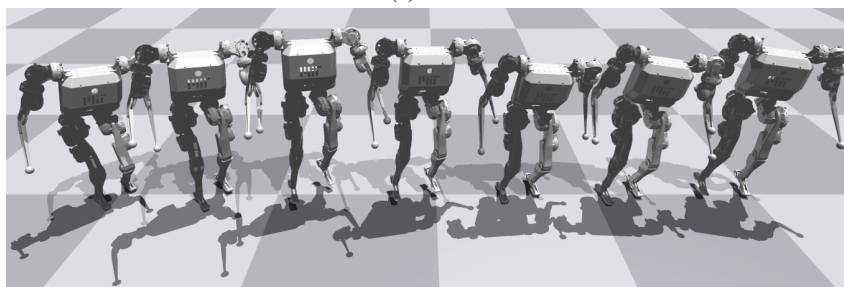


Figure 10: Normal motions learned by BMI.

However, we note that our policy still struggles with the difficult *spin-kick* and *cross-over* motions which are highly dynamic and can significantly influence the robot balance. Consequently, the robot prioritizes maintaining balance over replicating these motion patterns. For example, the robot rarely lifts its kicking foot in *spin-kick*, and the legs do not fully cross in *cross-over*, as shown in Figure 12.



(a) Kick



(b) Jump

Figure 11: Challenging motions learned by BMI.



(a) Spin-Kick



(b) Cross-Over

Figure 12: Unsatisfying motions learned by BMI.