CALM: Conditional Adversarial Latent Models for Directable Virtual Characters

CHEN TESSLER, NVIDIA, Israel
YONI KASTEN, NVIDIA, Israel
YUNRONG GUO, NVIDIA, Canada
SHIE MANNOR, NVIDIA, Israel and Technion Institute of Technology, Israel
GAL CHECHIK, NVIDIA, Israel and Bar-Ilan University, Israel
XUE BIN PENG, NVIDIA, Canada and Simon Fraser University, Canada

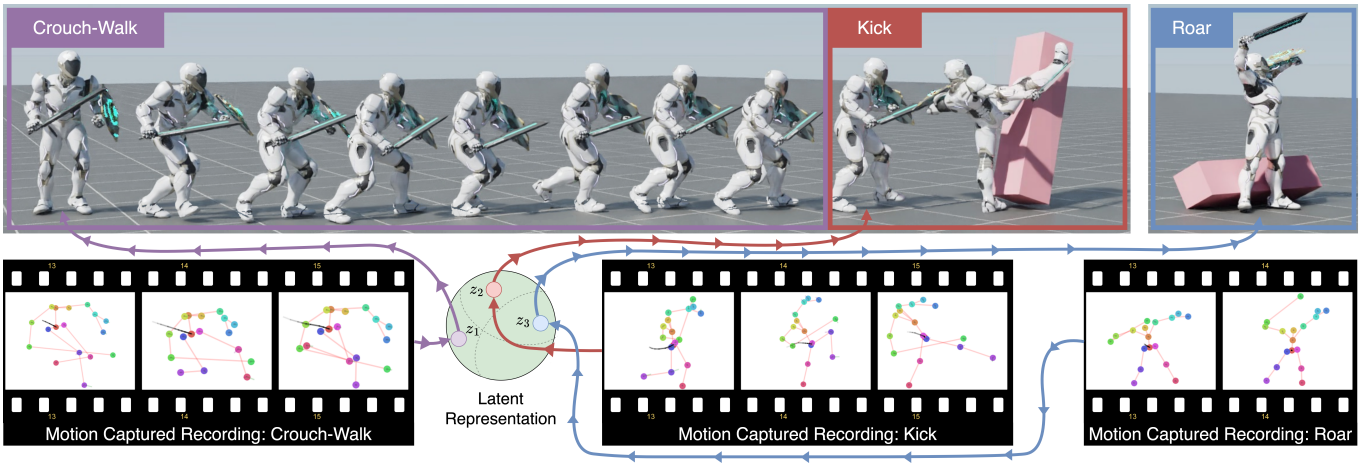


Fig. 1. Our framework enables users to direct the behavior of a physically simulated character using demonstrations encoded in the form of low-dimensional latent embeddings of motion capture data. In this example, the character is instructed to crouch-walk towards a target, kick when within range, and finally raise its arms and celebrate.

In this work, we present Conditional Adversarial Latent Models (CALM), an approach for generating diverse and directable behaviors for user-controlled interactive virtual characters. Using imitation learning, CALM learns a representation of movement that captures the complexity and diversity of human motion, and enables direct control over character movements. The approach jointly learns a control policy and a motion encoder that reconstructs key characteristics of a given motion without merely replicating it. The results show that CALM learns a semantic motion representation, enabling control over the generated motions and style-conditioning for higher-level task training. Once trained, the character can be controlled using intuitive interfaces, akin to those found in video games.

1 INTRODUCTION

Virtual environments and interactive characters have become more prevalent and user-friendly, but creating realistic and diverse behaviors for these virtual agents remains a challenge due to the complexity of human motion. To create interactive and immersive experiences, virtual agents must adapt to different environments and user inputs in a life-like manner, and this requires the ability to perform a wide range of behaviors on demand. To that end, we need to develop control models that can generate complex and realistic behaviors, while taking into account the properties of the environment. For example, in virtual reality games, players that interact with virtual characters and objects expect them to behave realistically. This includes responding to user commands and navigating through virtual environments. When virtual agents fail to respond naturally to user input, it can disrupt the immersive experience.

Recent advancements in machine learning and access to high-quality human motion capture data have led to the development of control policies that can replicate human behavior. Early studies in this field, such as [Peng et al. 2018], focused on imitating single motion clips. However, as each motion is learned using an independent controller, it does not effectively scale. Later research [Peng et al. 2022, ASE] aimed to improve the diversity of generated motion by learning a latent-conditioned controller and maximizing a mutual information objective. When trained on a dataset of diverse motions, distinct behaviors emerged, however at the cost of losing the ability to control the generated motion.

Building on these previous works, we present Conditional Adversarial Latent Models (CALM), a method for learning a representation of movement that captures the complexity and diversity of human motion, while also providing a directable interface for controlling a character's movements. Given raw motion capture recordings,

Authors' addresses: Chen Tessler, NVIDIA, Israel, ctessler@nvidia.com; Yoni Kasten, NVIDIA, Israel; Yunrong Guo, NVIDIA, Canada; Shie Mannor, NVIDIA, Israel and Technion Institute of Technology, Israel; Gal Chechik, NVIDIA, Israel and Bar-Ilan University, Israel; Xue Bin Peng, NVIDIA, Canada and Simon Fraser University, Canada.

illustrated in Figure 1, CALM encodes them into a latent representation. CALM further decodes a given latent vector into a skill for a physically simulated character, enabling it to perform high-level tasks while being conditioned on a desired motion. As can be seen in Figure 1, without any further training, using only predefined raw motion data, the agent can solve challenging tasks using a set of desired skills. A key benefit of our framework is that the policy does not need to precisely replicate the original reference motion. Instead, it has the flexibility to produce diverse movements, as long as they resemble the distributional characteristics of the particular motion clip. This enables the policy to deviate from the motion data and generate new and diverse behaviors that appear natural even though they were not explicitly depicted in the original dataset.

Compared to prior work, we focus on unsupervised techniques. The data is unlabeled and we do not assume prior knowledge of semantic connections between motions. We present an end-to-end method that jointly learns a meaningful semantic representation of skills and a control policy capable of producing the selected skills. While previous work used language to derive semantic connections [Juravsky et al. 2022], we demonstrate that semantic meaning can be directly inferred from the motions and similarity is determined in the context of the policy reproducing the motions. This ability to direct the character's motions then enables tasks to be solved using human-like control and without further re-training, demonstrating CALM's ability to generate interactive and directable virtual characters. To conclude, our contributions are as follows:

(1) We present a method for jointly training a generative motion controller and a motion encoder, from unlabeled motion capture data. The resulting policy can be directed to generate a motion M via its encoding $z = E(M)$.
(2) We introduce precision training, a way to reuse the pre-trained policy and leverage similarity within the learned latent space to enable control over the produced motion when solving high-level tasks, such as locomotion.
(3) Finally, we show that combining steps 1 and 2 enables the design of simple FSMs to solve tasks without further training or meticulous design of reward functions or termination conditions.

2 RELATED WORK

We aim to learn rich and reusable skill representations, from diverse unlabeled data, for character control. The field of data-driven motion generation can be broadly divided into kinematic models and physics-based models. Kinematic models directly generate pose trajectories without explicitly considering physical constraints, while physics-based models often use a physically simulated environment to enforce realistic dynamics during motion generation.

We differentiate between two methods of control for parametric models: direct prediction and latent/language-based control. In direct prediction, the model learns to generate motions for a predefined task, while latent and language-based control involves learning a generative model for motions. As our focus is on creating interactive, controllable characters, we build upon the literature of physics-constrained generation and focus on latent-based control.

2.1 Physics-constrained motion generation

In physics-constrained generation, the model predicts motor actuations. These are fed into the simulator, which then produces the next state by emulating the character's motion while adhering to the various governing laws (such as gravity and friction). Recent advances in deep learning, specifically deep reinforcement learning, have enabled a leap forward in generation quality. Such an example is DeepMimic [Peng et al. 2018], in which they defined a motion-tracking reward that is used for training a policy to imitate specific motions. However, these schemes focused on imitating single, pre-defined, motion clips.

Direct prediction. In direct prediction, the goal is to directly learn to solve a downstream task. For instance, the agent may be tasked with reaching a goal location. Here, AMP [Peng et al. 2021] combines adversarial imitation learning [Ho and Ermon 2016, GAIL] with classic RL. The agent attempts to balance between maximizing the task reward, whilst successfully fooling a discriminator. However, when the demonstration data distribution does not fit the task, the resulting behavior is unsatisfactory, and the resulting model is unable to generalize to new tasks – requiring re-optimizing the provided data and re-training per each task.

Latent-based control. Here the task is to learn a behavior manifold that can be sampled from. Park et al. [2019] learn to predict future states and then a controller to track that behavior. Won et al. [2022] learn a conditional variational auto-encoder (VAE), conditioned on the next state. Finally, closest to our work, ASE [Peng et al. 2022] presents an unsupervised discriminative learning procedure. By maximizing the mutual information between the latent space and the produced next state, in addition to optimizing the imitation learning objective, the agent learns to generate diverse motions.

These methods share a common theme – the resulting latent space is complex to control. In this work, we learn a dense representation of human motion jointly with a directable latent-conditioned policy.

2.2 Representation Learning

Representation learning, the task of capturing the underlying structure of data, has been a significant area of focus in the field of machine learning, particularly in the representation of images and videos. There are different ways to define similarity between data points, but one approach that is closest to our method is by directly utilizing a downstream task. For instance, generative adversarial networks [Goodfellow et al. 2020, GAN] and VAEs [Kingma and Welling 2013] learn a low-dimensional latent representation in order to recreate data from the reference distribution.

In the context of learning representations for motions, VAE and adversarial-based training can be differentiated. Motion VAEs [Ling et al. 2020; Won et al. 2022] enable jointly learning to represent and generate motion. While VAE-based methods are typically easier to train, their pitfall is that they are driven by a reconstruction loss, preventing the ability to deviate from the data distribution.

The benefit of adversarial methods is in their ability to generate new motions that are likely under the reference data distribution. This is important in the context of controllable characters. The

character must transition naturally between every motion pair, even when it is not provided with an explicit demonstration.

As shown in ASE [Peng et al. 2022], the policy learns to generate diverse behaviors, generalizing beyond motion reconstruction. However, their resulting latent space lacks global semantic structure. As a result, ASE tends to mode-collapse and does not provide an easy way to map motions to the latent space, an important requirement for directability and control. Alternative work such as PADL [Juravsky et al. 2022] utilized supervision from natural language.These methods are able to train language-aligned motion representations that can then generate behaviors based on natural language commands. However, they require access to labeled data.

In this work, the data is unlabeled and the representation is learned end-to-end with the imitation learning policy. Hence, the semantic connections between motions are determined in the context of the policy reproducing the motions.

2.3 Hierarchical Reinforcement Learning

Latent generative models can be seen as part of the options/skills framework [Sutton et al. 1999]. The options framework differentiates between a high and low-level controller. The low-level controller produces micro-actions, which are high-frequency actions capable of generating diverse behaviors. On the other hand, the high-level controller often plans at a lower frequency. At each time step, it selects which skill to play. Skills are temporally extended actions, or, in our context, long-term motions.

Prior efforts in hierarchical reinforcement learning (HRL) have shown that utilizing meaningful skills can not only speed up training [Tessler et al. 2017], but also enforce the solution to reside within the support of the data [Peng et al. 2022].

3 REINFORCEMENT LEARNING BACKGROUND

In this work, both the pre-training and the downstream tasks are modeled as reinforcement learning problems, where an agent interacts with an environment according to a policy π. At each step t, the agent observes a state s_t and samples an action a_t from the policy $a_t \sim \pi(a_t|s_t)$. The environment then transitions to the next state s_{t+1} based on the transition probability $p(s_{t+1}|s_t, a_t)$. The goal is to maximize the discounted cumulative reward, defined as

$$J = \mathbb{E}_{p(\tau|\pi)}\left[\sum_{t=0}^{T} \gamma^t r_t \middle| s_0 = s\right], \tag{1}$$

where $p(\tau|\pi) = p(s_0)\Pi_{t=0}^{T-1} p(s_{t+1}|s_t, a_t)\pi(a_t|s_t)$ is the likelihood of a trajectory $\tau = [s_0, a_0, r_0, \ldots, s_{T-1}, a_{T-1}, r_{T-1}, s_T]$, and $\gamma \in [0, 1)$ is the discount factor that determines whether the agent is short-sighted or considers longer-term outcomes.

4 OVERVIEW

In this paper, we introduce Conditional Adversarial Latent Models (CALM), a scalable, data-driven approach for creating directable controllers for physically simulated characters. These characters can be controlled in a similar way to how players control virtual characters in games, by providing a sequence of instructions for movement and actions. Figure 2 outlines our framework. It consists of three steps. (1) Low-level training of a motion encoder and motion generator. (2) Precision training using a high-level policy. (3) Inference, during which tasks are solved without further training.

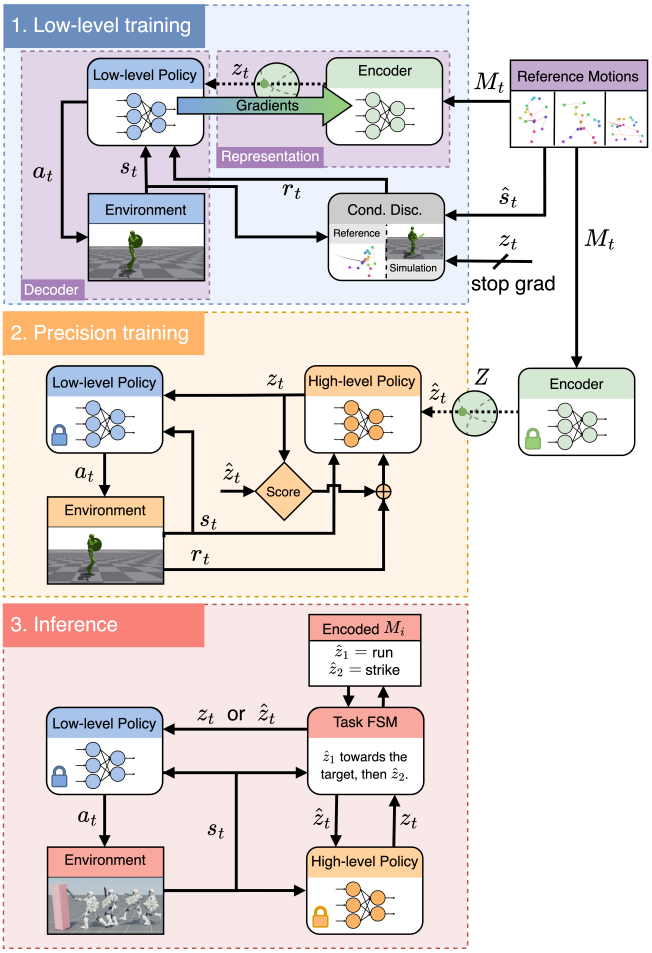


Fig. 2. **The CALM framework** consists of three phases. **1)** In the initial training phase, both the encoder and low-level policy are developed by utilizing feedback from a conditional discriminator. The encoder learns to create a condensed representation that encapsulates the core of the motion, while the low-level policy, which interacts with the environment, serves as a decoder. **2)** Once the pre-training phase has concluded, the encoder and low-level policy are frozen. The next step is **precision training**. Here, a high-level policy is trained to control the low-level policy. **3)** In the last phase, **inference**, the high-level policy is also frozen. Solutions to complex tasks are then described using finite state machines (FSM), rule-based systems that do not require training, instead of rewards. Based on the state of the task, the FSM either provides a command to the high-level policy or provides a latent directly to the low-level policy.

During low-level training, CALM learns an encoder E. It takes a motion M from a reference dataset of motions $\mathcal{M}$, a time-series of joint locations, and maps it into to a low-dimensional latent representation $z \in \mathcal{Z}$. Additionally, CALM also jointly learns a decoder. The decoder is a low-level policy $\pi(a|s, z)$ that interacts with the simulator and generates motions similar to the reference

dataset. This policy produces a variety of behaviors on demand, but is not conditioned on the directionality of the motion. For example, it can be instructed to walk, but does not enable intuitive control over the direction of walking.

Next, to control motion direction, we train a high-level task-driven policy to select latent variables z_t. These latents are provided to the low-level policy which generates the requested motion. Here, the motion encoder is used to constrain the latents z_t to be close to pre-specified motions $\hat{z}_t$, thus guiding the high-level policy to adopt a desired behavioral style. For example, move in a given direction while performing a crouch-walking motion.

Finally, the previously trained models are combined to compose complex movements without additional training. To do so, the user produces a finite-state machine (FSM) containing standard rules and commands. These determine which motion to perform, similar to how a user controls a video game character. For example, they determine whether the character should perform a simple motion, performed directly using the low-level policy, or a directed motion requiring high-level control. As an example, one may construct an FSM like (a) "crouch-walk towards the target, until distance < 1m", then (b) "kick", and finally (c) "celebrate" (Figure 1).

5 CONDITIONAL ADVERSARIAL LATENT MODELS

Learning a rich reusable skill representation enables characters to generate a wide variety of motions on demand, opening a range of potential applications such as games and visual effects. In CALM , these skills are modeled using a motion-conditioned policy $\pi(a|s,z)$, where motions M are encoded into a latent variable $z \in \mathcal{Z}$. This mapping is modeled using a motion encoder $z = E(M)$, and learned by solving a conditional imitation learning objective over motions M sampled uniformly from a reference dataset $\mathcal{M}$:

$$\max_{\pi} -\mathbb{E}_{M\in\mathcal{M}}\left[D_{\text{JS}}\left(d^{\pi}\left(s,s'|z\right)\big|_{z=E(M)}\middle\|d^{M}\left(\hat{s},\hat{s}'\right)\right)\right], \tag{2}$$

where D_{JS} is the Jensen-Shannon divergence, and $d^{\pi}\left(s,s'|z\right)\big|_{z=E(M)}$ and $d^{M}\left(\hat{s},\hat{s}'\right)$ are respectively the state transition distribution of the policy and reference motions.

This objective is related to prior work in imitation learning literature, namely learning from observations (*LfO*) [Torabi et al. 2018]. In this setup, the agent is provided with a demonstration dataset that consists only of state transitions/observations, without the underlying actions that produced those transitions. Generative Adversarial Imitation from Observation [Torabi et al. 2018, GAIfO] jointly trains a policy and a discriminator. The policy generates state transitions (s,s'), while the discriminator tries to distinguish them from demonstrations sampled from the data $(\hat{s},\hat{s}')$. However, provided a variety of motions, GAIfO is prone to mode-collapse, where the policy does not model all of the different behaviors from a large dataset, and the resulting model does not explicitly learn a skill embedding that can be used to direct the policy on downstream tasks.

In previous research, ASE [Peng et al. 2022] attempted to address these problems by introducing a latent variable and maximizing mutual information. They demonstrated that the model can produce diverse behavior when provided with randomly sampled variables. However, solely relying on mutual information loss is not enough. To avoid mode-collapse, ASE also employs a diversity loss, which states that similar latent variables should produce similar action distributions. Our study shows that this is particularly important in terms of directability. The encoder learns an imprecise mapping between motions and latents, resulting in a controller that does not produce similar motions when conditioned on the same latent, making it challenging to direct the policy to perform specific motions.

CALM mitigates the issues from prior methods by using a conditional discriminator that forces the policy to reproduce each motion in the dataset. At each iteration, a random motion M is sampled from the reference dataset. The encoder maps the motion to a latent encoding $z = E(M)$. Both the policy and the conditional discriminator are then conditioned on this latent z. Conditioning the discriminator on the latent helps to mitigate mode collapse, by forcing the policy to produce motions that are similar to the corresponding motion of a given latent. This leads to the following discriminator loss

$$\begin{aligned}\mathcal{L}_{\mathcal{D}} = -\mathbb{E}_{M\in\mathcal{M}}\Bigg[&\mathbb{E}_{d^{M}(\hat{s},\hat{s}')}\Big(\log\mathcal{D}(\hat{s},\hat{s}'|z)\Big) \\ &+\mathbb{E}_{d^{\pi(z)}(s,s')}\Big(\log(1-\mathcal{D}(s,s'|z))\Big)\Bigg|z=E(M)\Bigg],\end{aligned} \tag{3}$$

and policy objective

$$J = \mathbb{E}_{M\in\mathcal{M}}\left[\mathbb{E}_{p(\tau|\pi,z)}\left(\sum_t \gamma^t r(s_t,s_{t+1},z)\middle|s_0=s\right)\middle|z=E(M)\right], \tag{4}$$

with rewards $r(s_t,s_{t+1},z) = -\log\left(1-\mathcal{D}\left(s_t,s_{t+1}|z\right)\right)$.

5.1 Practical Considerations

In this section, we present design decisions needed for a practical instantiation of CALM, as well as in-depth implementation details.

5.1.1 Encoder. An ideal representation is one that is optimized for the control policy, rather than relying on auxiliary objectives to drive the structure of the latent representation, as in contrastive learning methods [Oord et al. 2018]. We show that an effective encoder can be trained in an end-to-end fashion using gradients from the policy when optimizing the pre-training objective Equation (4). Furthermore, this approach results in a latent space with a clear semantic structure, where similar motions are grouped closely, enabling interpolation in a semantically meaningful manner.

The encoder's output is projected onto the l_2 unit hypersphere. This constraint is inspired by prior work in the field [Bojanowski and Joulin 2017; Chen et al. 2020; Parkhi et al. 2015; Peng et al. 2022; Wang et al. 2017; Wang and Isola 2020; Xu and Durrett 2018], and has several benefits. For example, fixed-norm vectors are known to improve training stability in machine learning, where dot products are commonly used [Wang et al. 2017; Xu and Durrett 2018]. In the context of motion generation, the structure imposed on the latent space by the unit l_2 norm constraint reduces the likelihood of unnatural behaviors arising from sampling out-of-distribution latents during inference, as demonstrated by Peng et al. [2022].

As motion clips can be arbitrarily long, we split the data into overlapping sub-motions of 2 seconds. This results in motions that are long enough to present distinct and coherent characteristics. Additionally, sub-motions with close temporal proximity are likely to have similar characteristics. We leverage this understanding to

improve the latent space structure by applying an alignment and uniformity loss on the predicted embeddings [Wang and Isola 2020].

$$\mathcal{L}_{\text{align}} = \mathbb{E}_{(M,M')\overset{\text{overlapping}}{\sim}\mathcal{M}}\left[\left\|E(M) - E(M')\right\|_2^2\right], \tag{5}$$

$$\mathcal{L}_{\text{uniform}} = \log \mathbb{E}_{(M,M')\overset{\text{i.i.d}}{\sim}\mathcal{M}}\left[\exp\left(-2\left\|E(M) - E(M')\right\|_2^2\right)\right], \tag{6}$$

where *overlapping* corresponds to two-second sub-motions (M, M') from the same original motion sequence with non-zero overlap, and *i.i.d* to sub-motions randomly sampled from the data.

5.1.2 Skill Transitions. When performing new tasks, agents often need to sequentially transition between behaviors. Even in simple tasks, like reaching a location, the character needs to utilize a combination of skills like walking, turning, and standing. To enable smooth and robust transitions between motions, we explicitly train the model to transition between different motions by changing the conditional motion M at random timesteps. This teaches the policy to successfully transition between disparate motions.

5.1.3 Discriminative Loss. In adversarial imitation learning the discriminative reward provides the learning signal to the agent. To improve training dynamics, [Juravsky et al. 2022; Peng et al. 2022, 2021] propose to use a gradient penalty regularizer and negative sampling. This regularization helps mitigate discriminator overfitting and produces a smoother optimization landscape for the policy. The result is improved training stability and overall quality of the generated motion. In addition, as our goal is to learn a motion encoding optimal for the control task, we prevent gradients from flowing from the discriminator's objective into the encoder. This results in the following objective:

$$\begin{aligned}\mathcal{L}_{\mathcal{D}} = -\mathbb{E}_{M\in\mathcal{M}}\Big(&\mathbb{E}_{d^{\pi}(s,s'|z)}\left[\log\left(1-\mathcal{D}(s,s'|z)\right)\right] \\ &+\mathbb{E}_{d^{M}(\hat{s},\hat{s}')}\left[\log\mathcal{D}(\hat{s},\hat{s}'|z)+\log\left(1-\mathcal{D}(s,s'|z'\sim\mathcal{Z})\right)\right] \\ &+w_{\text{gp}}\mathbb{E}_{d^{M}(\hat{s},\hat{s}')}\left[||\nabla_{\theta}\mathcal{D}(\theta)|_{\theta=(\hat{s},\hat{s}'|z)}||^2\right]\Big|z=\text{stop grad}(E(M))\Big).\end{aligned} \tag{7}$$

Here, w_{gp} is the gradient penalty coefficient.

6 HIGH-LEVEL CONTROL

Once the low-level controller has been trained, it is used as a motion generator, grounding the motions of the character to those seen in the data. In this section, we present how to train a high-level policy to control the direction in which motions are performed, which can then be leveraged to solve complex tasks in varying forms without specifically training on them.

6.1 Precision training

Provided a motion encoding $\hat{z} = E(M)$, the low-level policy generates a matching motion, however, providing the low-level policy with the encoding for "run" does not control the running direction. To provide better control we train a high-level policy to generate motion in the requested form and direction.

Specifically, the high-level policy produces latent variables z_t. These are then provided to the low-level policy which controls the character. The character is tasked with moving in a specified direction $\mathbf{d}_t^*$ while crouch-walking M_1, or alternatively sprinting M_2. We achieve this by combining a task reward with a latent similarity loss. Specifically, given a motion encoding $\hat{z} = E(M)$, we train the high-level policy with the following reward

$$r_t^{\text{locomotion}} = \exp\left(-0.25\left\|\mathbf{d}_t^* - \frac{\dot{\mathbf{x}}_t^{\text{root}}}{||\dot{\mathbf{x}}_t^{\text{root}}||}\right\|^2\right) + \exp\left(-4||z_t - \hat{z}||^2\right), \tag{8}$$

where $\dot{\mathbf{x}}_t^{\text{root}}$ is the character's velocity.

6.2 Exemplar guidance

Given a pre-trained encoder and low-level policy (Section 5) and a pre-trained high-level policy (Section 6.1), the task designer describes how a task should be solved, in a natural and intuitive way, overcoming the fragility of reward design. Here, the task designer provides demonstrations for the various motions the agent should perform, enclosed within a finite-state machine (FSM) that determines when to transition between behaviors. For instance, the task of striking an object can be broken down into three phases "run towards the object", once within 0.5 meters then "perform an attack", and then "stand idle" until the next command.

Achieving such a level of control requires a combination of versatility, provided by the low-level policy, and precision, provided by a pre-trained high-level policy. During phases requiring precision, such as moving in a specified direction, the FSM provides the high-level policy with a requested motion embedding $\hat{z}$ and a direction in which this motion should be performed. When transitioning to isolated motions, such as a specific sword swipe, the FSM provides the motion encoding directly to the low-level policy.

This enables re-usability without re-training and resembles how a user would interact with the character given a game controller. A single combination of (a) low-level policy, (b) encoder, and (c) high-level policy, are used to solve unseen tasks in varying forms.

7 EXPERIMENTS

The data: To acquire diverse motion control capabilities, the low-level policy is trained using 160 motion clips totaling over 30 minutes in duration [Reallusion 2022]. Each motion clip is broken down into 2-second continuously overlapping sub-sequences, oblivious to transition boundaries. Hence, if a motion clip contains multiple skills, the 2-second clips may contain motions from several skills.. This includes basic movements such as various forms of walking, as well as more complex motions such as sword-strike combinations. This is explained in further detail in the supplementary material.

Training workflow: Throughout the pre-training process, the agent interacts with the environment for rollouts of K steps. At random timesteps, a random motion is sampled from the reference dataset, it is then encoded and provided to the low-level policy. The rollout then consists of the observed states resulting from the low-level policy, conditioned on the latent z, interacting with the environment. The low-level policy is then trained with respect to the collected rollout using PPO [Schulman et al. 2017].

We parallelize training over 4096 Isaac Gym [Makoviychuk et al. 2021] environments on a single A100 GPU, for a total of 5 billion steps. The low-level (high-level) policy takes decisions at a rate of 30 (6) Hz. The latent space $\mathcal{Z}$ is defined as a 64D hypersphere.

	Encoder quality ↓	Diversity ↑	Controllability ↑
CALM	**0.23**	**19.8±0.1**	**78%**
ASE	0.68	18.6±0.4	35%

Table 1. **Pre-training:** Quantitative evaluation of the learned encoder and low-level policy. We measure Fisher's concentration coeff. (Encoder quality), Inception score (Diversity), and Generation accuracy (Controllability).

Model architecture The encoder is a standard MLP, mapping $E(M) \mapsto z$, the policy and conditional discriminator each contain an additional input head $H(z)$ for latent parsing, followed by an MLP $\pi(s, H(z)) \mapsto a$, where $a \in \mathbb{R}^{31}$.

8 RESULTS

We tested the effectiveness of our method by using CALM to learn skill embeddings that allow a simulated humanoid to perform various motion control tasks. We demonstrate that CALM learns a semantically meaningful latent representation of diverse human motion and a directable policy. We then trained a high-level policy to control the direction in which these motions are performed. Finally, we show how these low and high-level policies can be re-used for solving unseen tasks without further training. See the motions produced by CALM in the provided supplementary video [1].

8.1 Controllable motion generation

We begin by analyzing three aspects of CALM: (1) the *encoder quality*, (2) *diversity* of the low-level controller, and (3) *controllability* of the combined system. Results are reported in Table 1. We focus our comparison on ASE [Peng et al. 2022], a latent generative model which learns to map arbitrary latent variables to motions.

Experiment 1: Encoder quality. Using Fisher's class separability metric [Bishop et al. 1995] over the representation learned by the encoder, we measure the separability between the motion classes within the latent space, where a motion class is defined as sub-motions within a single motion file. As shown in Table 1, CALM learns to encode motions into representations with much better separation.

Experiment 2: Diversity. We trained a classifier using the reference dataset from Section 5 to map a motion sequence to the originating motion index. We report the Inception Score [Salimans et al. 2016] over generated motions, when generated from randomly sampled latents $z \sim \mathcal{Z}$. As seen in Table 1, CALM significantly improves the diversity of generated motion.

Experiment 3: Controllability. Finally, we quantified how well CALM generates the requested motions, using a user study. Provided a reference motion and a textual description (taken from Juravsky et al. [2022]) raters were asked to classify the generated motion as similar or not. For each model, we presented raters with 40 reference motions and 3 generations per reference. We report the accuracy, measured as the percentage of accurate generations by the controller. The results show that CALM enables better control over the generated motions, compared to ASE, increasing the accuracy of perceived generation from 35% to 78%.

[1] See Section 9 for an explanation on rendering artifacts within the visualization process.

Motion	**Heading**		**Location**		**Strike**	
	Style	Score	Ending	Score	Ending	Score
Run	1	0.92		0.98		0.96
				0.99		1
				0.96		0.99
Walk	0.81	0.92		0.99		0.98
				1		1
				0.97		0.96
Crouch Walk	0.94	0.91		0.98		0.99
				0.99		1
				0.97		0.99

Table 2. Quantitative evaluation of **directional motion control** (heading) and **zero-shot task solution**. We consider three forms of locomotion: *run*, *walk*, and *crouch-walk*, each characterized by a different speed and style. For each task, we consider various finishing motions. For the location task, we consider (stand idle), (celebrate, arms-up), and (crouch idle). In the strike task, we consider: (kick), (shield charge), and (sword swipe).

This was enabled by performing end-to-end learning of both the representation (encoder) and the generative motion model (low-level policy) using a conditional discriminative objective. As a result, CALM learns to encode motion onto a semantically meaningful representation and a controller capable of generating motion with similar characteristics to the demonstration.

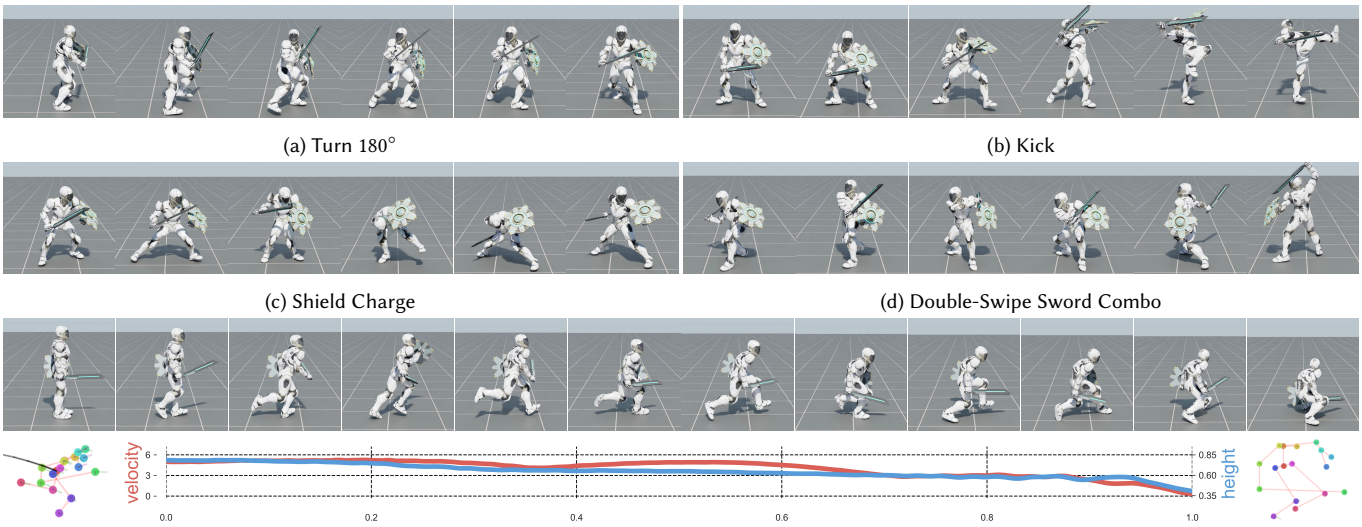
8.1.1 Qualitative analysis. In Figure 3, we show motions generated by CALM . Throughout a single episode, the conditional motions were changed, resulting in human-like transitions between the requested motions. Additionally, to illustrate the semantic structure of the latent space, we encode two semantically connected motions "sprint" and "crouching idle" and interpolate between their encodings over time. As shown in Figure 3e, CALM smoothly transitions between the two motions, decreasing both speed and height while continuously performing a form of walking motion.

8.2 Solving downstream tasks

Using the encoder and low-level policy from Section 8.1, we show how they can be used to compose motions for solving unseen tasks using commands akin to video game control.

8.2.1 Directional motion control. First, we show that provided a reference motion M and a direction d^*, a high-level policy can learn to control the low-level policy. We refer to this task as Heading. The character should produce motions with similar characteristics in the requested direction. We demonstrate the learned motions in Figure 4 and quantify them in Table 2 under the Heading column. Here, a high-level policy was conditioned on jointly learning "run", "walking with a raised shield", and "walk crouching".

During the evaluation, the high-level policy is conditioned on a fixed style and the direction is changed at random timesteps. We report the success in generating the requested style, measured using human raters, and the direction of motion, measured as the cosine distance between the requested and actual movement direction.



(a) Turn 180°

(b) Kick

(c) Shield Charge

(d) Double-Swipe Sword Combo

(e) Interpolation over time: from sprinting to crouching-idle

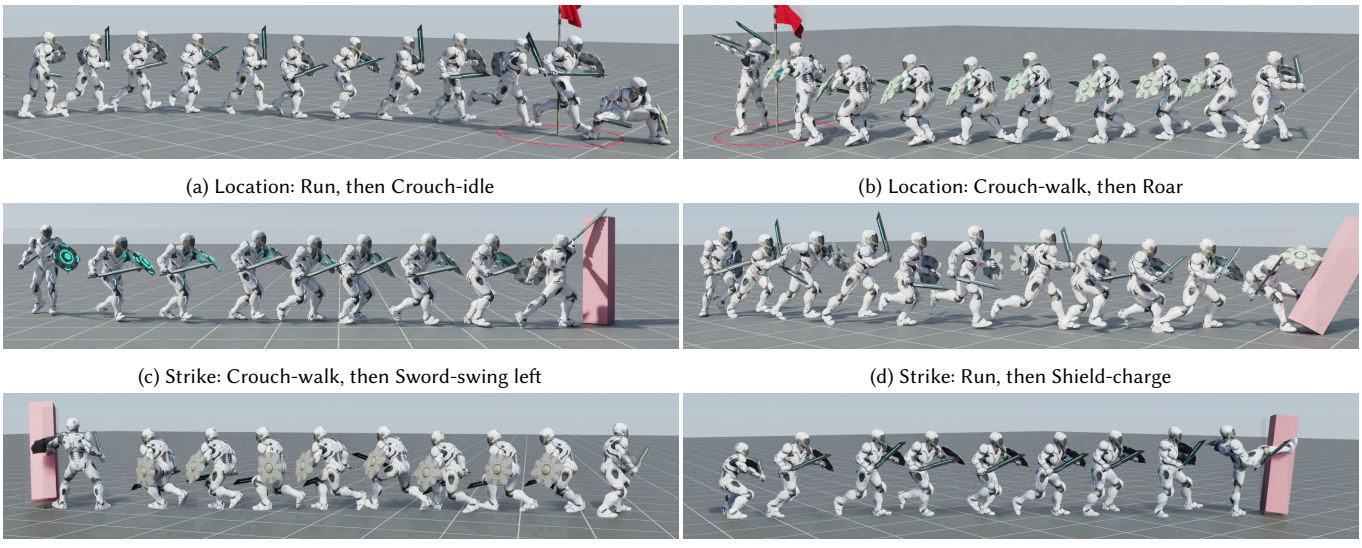
Fig. 3. **Low-level training:** Skills generated by a low-level controller conditioned on the encoding of a demonstrated motion.

(a) Heading: Run

(b) Heading: Walk, shield up

(c) Heading: Crouch-walk

Fig. 4. **Precision training:** A single high-level controller is trained to generate style-constrained locomotion via reward guidance.



(a) Location: Run, then Crouch-idle

(b) Location: Crouch-walk, then Roar

(c) Strike: Crouch-walk, then Sword-swing left

(d) Strike: Run, then Shield-charge

(e) Strike: Crouch-walk, then Shield-swipe

(f) Strike: Walk, then Kick

Fig. 5. **Inference:** Without any further training, CALM solves multiple different tasks and the same task in multiple different forms.

The results show that by constraining the latents to reside close to the reference motion encoding, the high-level policy is capable of generating motion in the specified style while ensuring it moves in the requested direction.

8.2.2 Solving tasks without further training. In our final experiment, we combine the directable low-level controller together with the high-level locomotion policy to provide zero-shot solutions to unseen tasks. We consider two tasks, location and strike. For the location task, the agent should reach and remain within the goal position – illustrated as a circle around the flagpost. Strike, a more complex task, requires the agent to reach the target and strike it down. In both cases, the character is controlled by conditioning a sequence of reference motions. To do so, the direction vector is provided to the target location, represented in the character's local coordinate frame. Once within range, the low-level policy is directly provided the latent corresponding to the requested action, e.g., kick, shield charge, or sword swipe.

As shown qualitatively in Figure 5 and quantitatively in Table 2, CALM can be used to solve tasks similarly to how a human, given a game controller, would solve them. Thanks to the controllability aspect of CALM, without any further training or task-specific reward design, the FSM sequentially orders the character to transition between motions. The result is a composition of human-like motion that solves the task.

9 LIMITATIONS

The results in Tables 1 and 2 show how CALM learns a diverse repertoire of motions without sacrificing controllability. This can then be leveraged to learn style-conditioned locomotion and finally to compose motions for solving multi-step, unseen, tasks. In this section, we highlight open challenges and questions that arise from this work.

Pre-training – mode collapse: We have shown that our algorithm CALM improves the controllability of generated motions compared to the existing approach, ASE, with a significant boost in performance from 35% up to 78%. However, mode collapse remains an open challenge. For instance, we have observed that conditioning the low-level controller on idle-motions can lead to unrealistic micro-motions that slowly move the character. Although our approach addresses the problem of mode collapse to a large extent, there remains room for further improvements.

Pre-training – unseen motions: Our work focuses on learning a latent generative motion controller for in-distribution motions. However, when conditioning the character on encodings from unseen motions, we cannot guarantee the quality of the generated motions. While we have observed that some unseen motions map to semantically similar motions from the data, such as tip-toe mapping to bounce-walk, we anticipate that the model may fail as the motions become increasingly out of distribution.

Precision-training – beyond locomotion: In Table 2 our approach is shown to leverage the learned latent space and achieve style-conditioned locomotion. However, controlling intricate movements such as the path of a sword or shield in an attack may require additional innovations in the pre-training phase, such as learning motions with a larger distributional discrepancy to the data.

FSM – robustness: Our approach has demonstrated the ability to solve tasks using classic tools from the gaming and animation industry, such as FSM and behavior trees, presenting a game-controller-like interface, as shown in Figure 5. However, we anticipate that the policy's robustness envelope may limit its ability to solve tasks with vastly different dynamics from those seen during training, such as climbing stairs or walking on uneven terrain. Therefore, further innovations and training may be required to solve such tasks.

Rendering – artifacts: To illustrate how our work can be integrated into the gaming industry, we visualize the motions using high-resolution characters, rendered within Omniverse (OV). Although physically accurate motion is recorded in IsaacGym (IG) where physical constraints are maintained without visual artifacts, the visualization character in OV is not subject to these constraints during rendering. Consequently, due to the difference in character geometry, the visualization character may exhibit penetrations and other visual artifacts that do not occur in IG. It is worth noting that the issue of rendering artifacts is not an inherent problem with our proposed algorithm CALM , but due to the differences in character geometry. One way to minimize these artifacts is by ensuring that the simulated character's geometry is closer to that of the visualization. Another solution is robustifying the training process to handle varying character morphologies to directly control the visualization character while enforcing physical constraints.

10 DISCUSSION AND FUTURE WORK

In this work, we presented CALM, a framework for learning reusable and directable motor skills for physics-based character animation. Our model enables the character's behaviors to be directed using motion clips. Given an unlabeled motion dataset, CALM learns both an encoder and a low-level controller. The encoder maps motions onto a semantically meaningful low-dimensional representation and a low-level controller takes the role of a decoder and produces motions with similar characteristics to those encoded within the learned representation. These reference motions can be used both for controlling low-level skills and to guide higher-level controllers and specify which motions the character should use when solving complex tasks. The ability to control the generated motion of the character enables zero-shot solutions to complex multi-step tasks, a step towards real integration of interactive virtual characters.

Our motion-constrained training enabled guiding the solution towards utilizing pre-specified motions. However, successfully learning to produce the requested motion in the specified direction required delicate tuning of the reward parameters. We are interested in exploring ways for disentangling the representation of direction from the representation of the content motion. Such disentanglement will enable high-level policies to learn with simplified rewards, while ensuring that they produce motion with the desired characteristics. Finally, some motions require coordinated interaction with the environment. For instance, aerobatic motions like handsprings and vaults can only be performed while interacting with a vaulting table/elevated box. We intend to investigate automation methods for understanding the motion-object pairs, and the integration of such objects throughout training for learning the respective motions.

REFERENCES

Christopher M Bishop et al. 1995. *Neural networks for pattern recognition*. Oxford university press.

Piotr Bojanowski and Armand Joulin. 2017. Unsupervised learning by predicting noise. In *International Conference on Machine Learning*. PMLR, 517–526.

Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. 2020. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*. PMLR, 1597–1607.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2020. Generative adversarial networks. *Commun. ACM* 63, 11 (2020), 139–144.

F Sebastian Grassia. 1998. Practical parameterization of rotations using the exponential map. *Journal of graphics tools* 3, 3 (1998), 29–48.

Jonathan Ho and Stefano Ermon. 2016. Generative adversarial imitation learning. *Advances in neural information processing systems* 29 (2016).

Jordan Juravsky, Yunrong Guo, Sanja Fidler, and Xue Bin Peng. 2022. PADL: Language-Directed Physics-Based Character Control. In *SIGGRAPH Asia 2022 Conference Papers* (Daegu, Republic of Korea) *(SA '22)*. Association for Computing Machinery, New York, NY, USA, Article 19, 9 pages. https://doi.org/10.1145/3550469.3555391

Diederik P Kingma and Max Welling. 2013. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114* (2013).

Hung Yu Ling, Fabio Zinno, George Cheng, and Michiel Van De Panne. 2020. Character controllers using motion vaes. *ACM Transactions on Graphics (TOG)* 39, 4 (2020), 40–1.

Viktor Makoviychuk, Lukasz Wawrzyniak, Yunrong Guo, Michelle Lu, Kier Storey, Miles Macklin, David Hoeller, Nikita Rudin, Arthur Allshire, Ankur Handa, and Gavriel State. 2021. Isaac Gym: High Performance GPU Based Physics Simulation For Robot Learning. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*. https://openreview.net/forum?id=fgFBtYgJQX_

Aaron van den Oord, Yazhe Li, and Oriol Vinyals. 2018. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748* (2018).

Soohwan Park, Hoseok Ryu, Seyoung Lee, Sunmin Lee, and Jehee Lee. 2019. Learning predict-and-simulate policies from unorganized human motion data. *ACM Transactions on Graphics (TOG)* 38, 6 (2019), 1–11.

Omkar M Parkhi, Andrea Vedaldi, and Andrew Zisserman. 2015. Deep face recognition. (2015).

Xue Bin Peng, Pieter Abbeel, Sergey Levine, and Michiel Van de Panne. 2018. Deepmimic: Example-guided deep reinforcement learning of physics-based character skills. *ACM Transactions On Graphics (TOG)* 37, 4 (2018), 1–14.

Xue Bin Peng, Yunrong Guo, Lina Halper, Sergey Levine, and Sanja Fidler. 2022. ASE: Large-scale Reusable Adversarial Skill Embeddings for Physically Simulated Characters. *ACM Trans. Graph.* 41, 4, Article 94 (July 2022).

Xue Bin Peng, Ze Ma, Pieter Abbeel, Sergey Levine, and Angjoo Kanazawa. 2021. Amp: Adversarial motion priors for stylized physics-based character control. *ACM Transactions on Graphics (TOG)* 40, 4 (2021), 1–20.

Reallusion. 2022. 3D Animation and 2D Cartoons Made Simple. (2022). http://www.reallusion.com

Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. 2016. Improved techniques for training gans. *Advances in neural information processing systems* 29 (2016).

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347* (2017).

Richard S Sutton, Doina Precup, and Satinder Singh. 1999. Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning. *Artificial intelligence* 112, 1-2 (1999), 181–211.

Chen Tessler, Shahar Givony, Tom Zahavy, Daniel Mankowitz, and Shie Mannor. 2017. A deep hierarchical approach to lifelong learning in minecraft. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 31.

Faraz Torabi, Garrett Warnell, and Peter Stone. 2018. Generative Adversarial Imitation from Observation. *CoRR* abs/1807.06158 (2018). http://arxiv.org/abs/1807.06158

Laurens Van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-SNE. *Journal of machine learning research* 9, 11 (2008).

Feng Wang, Xiang Xiang, Jian Cheng, and Alan Loddon Yuille. 2017. Normface: L2 hypersphere embedding for face verification. In *Proceedings of the 25th ACM international conference on Multimedia*. 1041–1049.

Tongzhou Wang and Phillip Isola. 2020. Understanding contrastive representation learning through alignment and uniformity on the hypersphere. In *International Conference on Machine Learning*. PMLR, 9929–9939.

Jungdam Won, Deepak Gopinath, and Jessica Hodgins. 2022. Physics-based character controllers using conditional VAEs. *ACM Transactions on Graphics (TOG)* 41, 4 (2022), 1–12.

Jiacheng Xu and Greg Durrett. 2018. Spherical Latent Spaces for Stable Variational Autoencoders. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*. 4503–4513.

A STATE AND ACTION SPACE

In this work, we consider a 3D physically-simulated humanoid character wielding a sword and a shield, with 37 degrees of freedom. A similar character was used in Peng et al. [2022]. To encode the state, we follow the same representation technique from Peng et al. [2021]. The agent observes:

- Root (character's pelvis) height.
- Root rotation with respect to the character's local coordinate frame.
- Local rotation of each joint.
- Local velocity of each joint.
- Positions of hands, feet, shield, and the tip of the sword, all in the character's local coordinate frame.

The character's local coordinate frame is defined with the origin located at the root, the x-axis oriented along the root link's facing direction, and the y-axis aligned with the global up vector. The 3D rotation of each joint is encoded using two 3D vectors, corresponding to the tangent $\mathbf{u}$ and the normal $\mathbf{v}$ of the link's local coordinate frame expressed in the link parent's coordinate frame [Peng et al. 2021]. In total, this results in a 120D state space.

A.1 Low-level policy

In addition, the low-level policy observes a 64D latent representation of motion.

To control the character, the agent provides an action a that represents the target rotations for PD controllers, which are positioned at each of the character's joints. Similar to Juravsky et al. [2022]; Peng et al. [2022, 2021], the target rotation for 3D spherical joints are encoded using a 3D exponential map [Grassia 1998], resulting in a 31D action space.

A.2 Encoder

The encoder learns to map fixed-length motions onto a low-dimensional representation. We consider 2-second motions. As the low-level controller operates at 30Hz, this results in 60 frames. During training, we randomly sample motion clips. If the motion clip is longer than 2 seconds, we randomly sample a continuous 2-second window and interpolate between the discrete supports of the motion. On the other hand, if the motion is shorter than 2 seconds, we apply zero padding.

A.3 Discriminator

The discriminator is trained similarly to ASE [Peng et al. 2022]. Given a 2-second motion clip M, the discriminator is conditioned on the corresponding latent encoding $z = E(M)$ and tasked with differentiating between $(\hat{s}_1, \ldots, \hat{s}_{10})$ randomly sampled from the reference motion $\mathcal{M}$ and the transition sequence $(s_1, \ldots, s_{10})$ generated by the policy, also conditioned on the same latent z.

A.4 High-level policy

The high-level policy observes additional task information and produces latent variables $z \in \mathcal{Z}$ that are provided to the low-level policy.

A.4.1 Block. In the block task, a projectile is launched toward the agent. The agent is required to block the projectile using its shield. In each episode, the projectile has two phases. During the warmup phase, it remains static, followed by a launch phase in which it travels towards a target area in the character's body.

The high-level policy observes

- The relative progress within this process.
- The location of the origin, in the character's local coordinate frame.
- The local of the projectile, in the character's local coordinate frame.
- The angle to the projectile.
- The projectile's velocity and its angular velocity.

this results in an additional 18D task-specific observation.

A.4.2 Reach. The goal of reach is to place the tip of the sword in a specified location and maintain this position. Here, the agent observes the 3D position of the goal location, in the character's local coordinate frame.

A.4.3 Locomotion. The locomotion task is a motion-conditioned task. Provided a set of motions $\tilde{\mathcal{M}}$, at time t, the high-level policy is tasked with moving in a specified direction, while utilizing motions with similar characteristics as $M_t \in \tilde{\mathcal{M}}$. To do so, it observes

- The target direction, in the character's local coordinate frame.
- A one-hot encoding $|\tilde{\mathcal{M}}|$ of the current specified motion M_t.

B ARCHITECTURE DETAILS

In this work, all networks are composed of fully connected layers. The encoder and high-level policies are standard MLP networks. Prior to concatenating the latent representation with the observation, the low-level policy and the conditional discriminator also incorporate a pre-processing MLP for the latent variable z.

C COMPARISON OF CALM TO OTHER PHYSICS-CONSTRAINED MOTION GENERATION FRAMEWORKS

Prior work has also considered the challenge of physical character animation. In this section, we compare our method CALM with prior work, highlighting the differences both in the method and the results.

C.1 Motion matching

Initial efforts in generating complex character motion focused on heuristics for motion matching. In DeepMimic [Peng et al. 2018], a reward is formulated for the deviation between the character's current pose and the corresponding pose in the data. This was later extended with motion VAEs enabled a certain degree of control by developing character controllers using the VAE paradigm. Notable efforts are Ling et al. [2020] and Won et al. [2022] that learned, respectively, an unconditional and conditional motion VAE for motion reconstruction.

Citing Won et al. [2022], we highlight the main challenge in motion-matching schemes:

"our controller can also have the sinking problem mentioned in [Ling et al. 2020], where it fails to transition among different behaviors if the input trajectories lack such transitions. Using datasets composed of many heterogeneous behaviors where individual recordings are relatively short might cause this problem. Augmenting datasets by constructing motion graphs, which we used in our experiments, might be a quick remedy, however, preparing datasets rich in transitions would create the most natural-looking motions."

As the low-level controller learns based on a state-reconstruction loss, it lacks the built-in ability to generalize beyond what was seen in the data. A core strength of adversarial techniques is that they force the behavior to be likely under the reference data distribution, enabling the emergence of new and novel movements such as transitioning between motion classes, as seen in Figure 7.

On the other hand, a core benefit of methods aiming to directly mimic demonstrations is that the per-trajectory loss is highly informative of the agent's ability to reconstruct a given motion. This, which may be non-trivial in adversarial methods, can easily be leveraged for adaptive sampling [Park et al. 2019], thus reducing mode-collapse and improving overall quality.

C.2 Adversarial techniques

More similar to our line of work are adversarial techniques. Here, the motion-matching reward is replaced with a discriminative signal. The discriminator produces a signal corresponding to how likely the generated motions are, given the data.

Initially, AMP [Peng et al. 2021] focused on solving single tasks using an unconditional policy and unconditional discriminator. The lack of conditioning prevents the ability to direct the behavior. Hence, optimizing the adversarial objective results in generating the entire reference data distribution. To force human-like solutions, the reference data distribution is carefully tuned to match the expected task statistics.

This effort was later extended by ASE [Peng et al. 2022] to split learning into two phases – low-level training, followed by high-level training. First, a low-level conditional policy is trained with an unconditional discriminator on a diverse dataset with an information maximization term. This enabled learning diverse behaviors that can be controlled by fixing the conditioning variable z. However, the lack of correspondence between motions and latents results in mode-collapse, requiring tricks such as a diversity loss in order to improve the quality. As a result, controlling the motions generated by ASE is not trivial.

Finally, closest to our work, PADL [Juravsky et al. 2022] introduced a scheme composing both a conditioned policy and a discriminator. The main differences between CALM and PADL are that (1) PADL requires labeled data, which is both costly to acquire and less expressive than demonstrations. We do not assume any supervision beyond the ability to obtain sequences of motions. (2) PADL learns the representation separately from the control. We jointly learn the control and representation. (3) PADL directly learns a task-driven controller whereas we learn a general low-level conditional skill generator that can be re-used without further re-training.

As a result, CALM learns without supervision (1) a well-structured latent representation that captures the semantic meaning of motions.

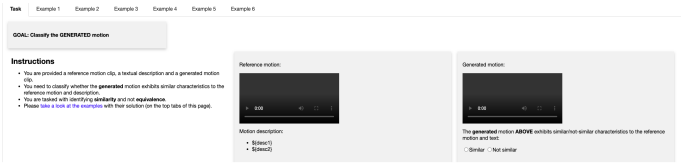


Fig. 6. Motion classification system

This is seen in the interpolation experiment Figure 3e where the character transitions smoothly through a semantically meaningful path. (2) CALM provides a more scalable approach, where the same low-level controller can be re-used for multiple tasks as shown in Figure 5.

D QUALITATIVE ANALYSIS OF THE LOW-LEVEL CONTROLLER

In this section, we present some additional information regarding the quantitative results from Section 8.1.

To analyze what CALM has learned, we performed two tests. The first aims to evaluate the encoder and what it has learned. We report the Fisher's distance, which measures the concentrability. Given a motion file M and 2-second sliding window portions of it M_i

$$coeff(M) = \frac{\sum_{M_i, M_j \in M} \frac{||E(M_i) - E(M_j)||}{|M|^2}}{\sum_{M_i \in M, M_k \in \mathcal{M}} \frac{||E(M_i) - E(M_k)||}{|\mathcal{M}| \cdot |M|}} \tag{9}$$

and the score is defined as the average concentrability coefficient over motion files.

Next, we tested the ability of the low-level controller to generate motions on demand. We randomly selected a subset of 40 motions from the dataset. We conditioned CALM and ASE, each with their own respective encoder, to generate motion with similar characteristics. The initial state was randomly sampled from the data. Human raters were tasked with classifying each generated motion, where we generated 3 motions per reference clip.

We report the mean accuracy over all generated motions. In addition, we present a screenshot from our questionnaire in Figure 6.

E ADDITIONAL EXPERIMENTS

Complementing the experiments in the main paper, we present some additional results. The motions are best seen in the supplementary video.

E.1 Ablation analysis

To analyze the importance of the various design decisions for the pre-training phase, we perform an ablation experiment. We compare the full model, with removing negative samples from the discriminator training, and with removing the latent space regularization term.

As shown in Table 3, adding each design element further improves the generation quality of the model.

E.2 Skill transitions

To test the ability of the agent to transition between skills in a natural manner and on demand, we run the low-level policy on a long episode. During the episode, we sample a motion clip from

	Concentration ↓	Generation ↑	Accuracy ↑
CALM	**0.23**	**19.8±0.11**	**78%**
w/o negative samples	0.24	15.7±0.07	62%
w/o negative samples, w/o regularization	0.35	12.8±0.05	61%

Table 3. **Pre-training ablation:** We perform ablation for various design choices for the pre-training phase.

the reference dataset and condition the agent on the corresponding latent.

As can be seen in Figure 7, the agent learns to transition between complex motions.

E.3 Downstream tasks

In the paper we focused on the unique benefits of our method, namely, enabling the solution of unseen tasks without any additional training. Despite zero-shot applications being of great interest, sometimes an automated solution is needed.

In Figure 8 we present scenarios where the high-level policy is trained directly to solve downstream tasks without style conditioning.

E.3.1 Block. In the block task, a projectile is thrown at the agent and it is required to block it with the shield. To solve the task, the high-level policy is provided with a positive reward when a projectile hits its shield

$$r_t^{\text{block}} = \mathbf{1}(\text{projectile blocked by shield}) . \tag{10}$$

E.3.2 Reach. Since the 'Block' task tests control over the shield-bearing arm, we also showcase the 'Reach' task, which requires precise sword motions. Here, the agent is tasked with bringing the tip of its sword to a specified x^* location. To achieve this, it is provided a reward

$$r_t^{\text{reach}} = \exp\left(-4||x_t^{\text{sword}} - x^*||_2^2\right) , \tag{11}$$

where x_t^{sword} is the location of the tip of the sword at time t.

E.3.3 Location. During 'Block' and 'Reach' the character remains rooted in the same spot, yet requires accurate low-level motions. To showcase the ability to compose motions directly from reward, as shown in Peng et al. [2022], we also train a 'Location' and 'Strike' task.

In the location task, the character needs to reach a target location. The high-level policy receives a reward

$$r_t^{\text{location}} = \exp\left(-0.5||x^* - x_t^{\text{root}}||^2\right) , \tag{12}$$

where x^* is the goal location and x_t^{root} is the location of the character's root. This reward urges the high-level controller to produce motions that reach the goal location as fast as possible. As seen in Figure 8c, the controller learns to select latent variables z_t that result in running.

E.3.4 Strike. The 'Strike' task tests the ability of the high-level controller to transition between motions within a complex multi-step task. Here, the agent is required to strike down a target. To do so, the high-level policy receives a reward

$$r_t^{\text{strike}} = 1 - \mathbf{u}^{\text{up}} \cdot \mathbf{u}_t^* , \tag{13}$$

where $\mathbf{u}^{\text{up}}$ is the global up vector, and $\mathbf{u}_t^*$ is the local up vector of the target object, expressed in the global coordinate frame. In addition, to prevent the character from just crashing into the target, Peng et al. [2022] provides a termination condition in which the environment terminates if the character touches the target with any element that isn't the sword.

The reward urges to character to quickly reach the target and the termination makes sure it only does so by using the sword. As seen in Figure 8d, the character runs towards the target and then performs a sword strike to hit it.

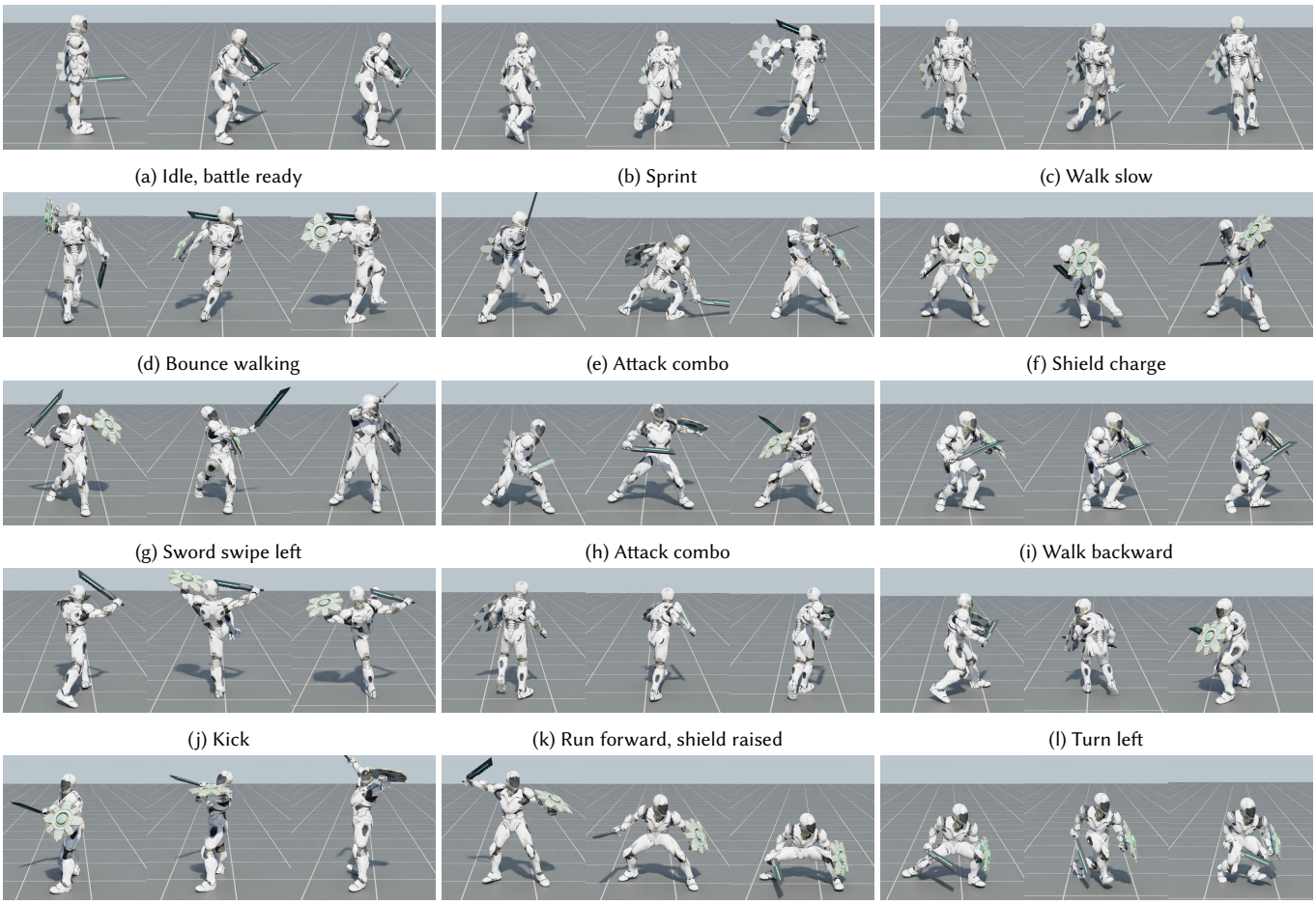
E.3.5 Tasks conclusions. Despite the low-level policy being trained on generating long-term behaviors, through iterative latent control, the high-level policy is capable of producing new motions to solve the task. For instance, to block the projectile it learns to control where the shield is aimed, and in the reach task, it learns to control the location of the sword and maintain a relatively static position around the goal location. These specific behaviors showcase the benefit of adversarial training schemes in their ability to generalize beyond what was observed in the reference dataset. Finally, when tasked with complex long horizon tasks, the character learns to utilize human-like motions while also transitioning naturally between them.

E.3.6 FSM versus Reward design. In the paper we presented a way for solving tasks, such as location and strike, without performing task-specific training. This was done by leveraging an FSM design and the fact that the low-level policy can generate specified motions on demand. The benefit compared to reward design is clear. Learning to crouch-walk towards the target and then kick it, followed by a celebrative roaring motion – requires a delicate reward and termination design. On the other hand, by utilizing the FSM approach, the character can be directed to perform specific motions in a specified direction, resulting in diverse solutions to tasks without reward design and without any task-specific training.

F LATENT SPACE ANALYSIS

To gain further insight into the learned representations, we analyze the structure of the latent space. We split the motions, roughly, into 5 categories:

- Walking motions
- Sword attacks
- Shield attacks
- Turning motions
- and Idle motions



(a) Idle, battle ready (b) Sprint (c) Walk slow

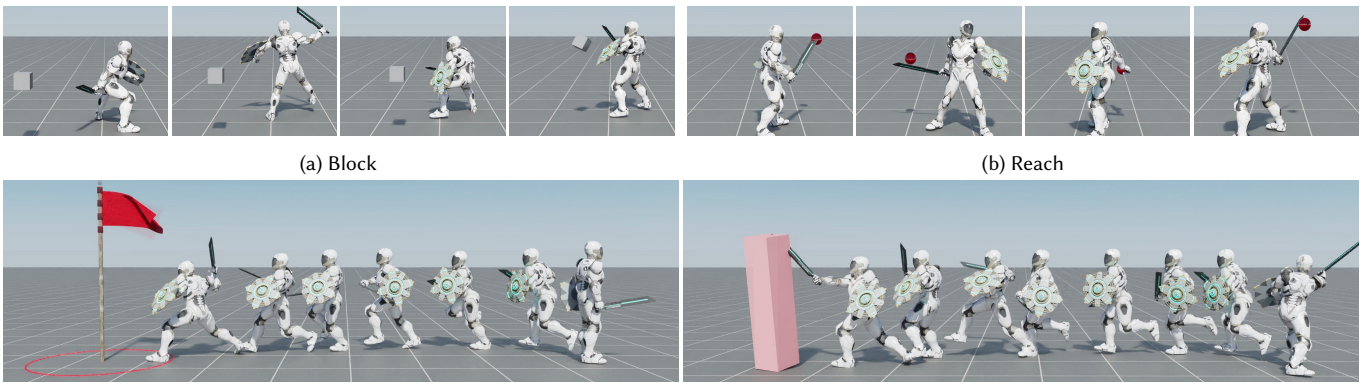
(d) Bounce walking (e) Attack combo (f) Shield charge

(g) Sword swipe left (h) Attack combo (i) Walk backward

(j) Kick (k) Run forward, shield raised (l) Turn left

(m) Roar, hands up (n) Crouch, idle (o) Crouch, walk

Fig. 7. Skills generated by a low-level controller conditioned on the encoding of a demonstrated motion. Every 3 seconds, a new motion is sampled and the low-level policy is conditioned on the corresponding latent representation. These figures are generated from a single, long, episode during which the motions were performed sequentially.



(a) Block (b) Reach

(c) Location (d) Strike

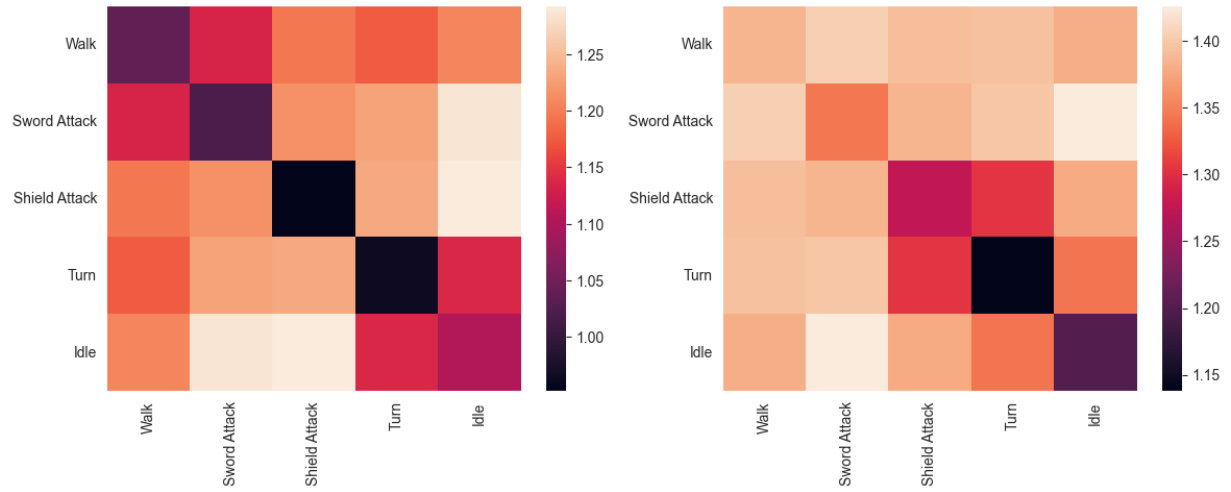
Fig. 8. Standard hierarchical task solving. A high-level policy directly controls the low-level policy and optimizes a task-specific objective.

For each motion in the reference dataset, we encode it using each method's respective encoder. We then calculate the average pairwise distance between groups. As seen, in Figure 9, CALM clusters motion groups closer in the latent space. This is observed by a lower distance along the diagonal. The meaning is that CALM learned a representation with stronger semantic meaning – similar motions are clustered closely within the latent space.

Moreover, analyzing the distances within the ASE encodings shows that it does not maintain semantic relations between motion classes. Specifically, motions corresponding to walking are dispersed over the latent space.

In addition, we present a TSNE plot [Van der Maaten and Hinton 2008] of the learned representation for CALM. While all the data is plotted, due to the vast number of motions (approximately 180), for clarity, we only label a subset.

While TSNE does not preserve the global structure, as seen in Figure 10, it does suggest that sub-sequences from the same motion clip do indeed receive a similar representation, in the sense that they reside in proximity in the latent space.



(a) CALM (b) ASE

Fig. 9. Distance between motion classes within the latent space. Darker colors represent higher pairwise-latent-space proximity between the two motion classes.

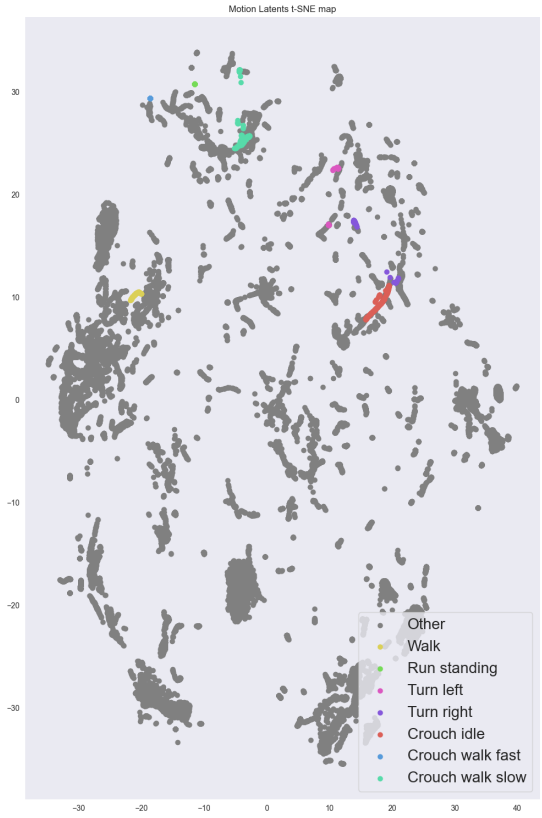


Fig. 10. CALM TSNE analysis