Policy-space Interpolation for Physics-based Characters

MICHELE ROCCA, Univeristy of Copenhagen, Denmark
SHELDON ANDREWS*, École de technologie supérieure, Canada
KENNY ERLEBEN*, University of Copenhagen, Denmark

Fig. 1. One hundred characters ($c = 100$) are each controlled by a unique interpolated policy, generated by randomly combining three trained policies ($n = 3$). This approach introduces meaningful variability without the need for retraining or fine-tuning. In contrast, achieving the same effect with action interpolation would require nc policy evaluations per step, whereas our method requires only c evaluations per step, independent of the number of policies included in the interpolation.

Controllers for physics-based humanoids often rely on reference animations to synthesize plausible motion trajectories for task-driven control. When the controller is a deep reinforcement learning policy, the output of several controllers can be combined to enhance controller robustness and synthesize new motion variations. However, this requires evaluating several policies at each timestep and combining their outputs, which can be computationally costly. In this work, we propose an alternative approach that combines individual controllers using linear interpolation of network parameters, thereby requiring only a single policy evaluation per timestep. Our method employs a graph-based weight regularization strategy to ensure that similar motions generate similar policy weights during training. We show that this technique produces visually indistinguishable outcomes compared to blending controller outputs, and that the approach easily integrates new control policies without retraining existing ones. We further demonstrate that interpolating or perturbing individual layers results in novel variations of the internal motion pattern that cannot be easily achieved by operating on the actions. This opens a path toward improved variability in controller networks by manipulating their weights. Several compelling use cases demonstrate the benefits of our approach, including interactive control and synthesizing motion variations.

CCS Concepts: • **Computing methodologies** → **Physical simulation**; **Motion processing**.

Additional Key Words and Phrases: Control Policies, Deep Reinforcement Learning, Interactive Animation, Weight-space

*Equal contribution to supervision of the project.

Authors' Contact Information: Michele Rocca, Computer Science, Univeristy of Copenhagen, Copenhagen, Denmark, miro@di.ku.dk; Sheldon Andrews, LOGTI, École de technologie supérieure, Montreal, Canada, sheldon.andrews@etsmtl.ca; Kenny Erleben, Computer Science, University of Copenhagen, Copenhagen, Denmark, kenny@di.ku.dk.

ACM 2577-6193/2025/8-ART61
https://doi.org/10.1145/3747863

ACM Reference Format:
Michele Rocca, Sheldon Andrews, and Kenny Erleben. 2025. Policy-space Interpolation for Physics-based Characters. *Proc. ACM Comput. Graph. Interact. Tech.* 8, 4, Article 61 (August 2025), 22 pages. https://doi.org/10.1145/3747863

Project page: michelerocca.github.io/projects/policy-space_linterp

1 Introduction

Physics-based character animation plays a crucial role in modern computer graphics by enabling the generation of realistic and physically plausible motion through the simulation of forces, contacts, and interactions with dynamic environments. While early approaches relied on manually specified controllers [Faloutsos et al. 2001; Hodgins et al. 1995; Yin et al. 2007] or numerical optimization [Liu et al. 2005; Sharon and van de Panne 2005] to synthesize physically plausible motions that meet desired objectives, recently deep reinforcement learning (DRL) has emerged as a central approach for learning dynamic controllers for physics-based characters [Bergamin et al. 2019; Kwiatkowski et al. 2022; Peng et al. 2018]. DRL often integrates task-specific rewards with imitation rewards that track reference motions, thus allowing the controller to learn complex motor skills from plausible human motion data. Each imitation controller exhibits a single motion style, learning to reproduce trajectories similar to those found in the reference motion. However, various motion styles for a diverse set of skills can be achieved by dynamically transitioning and blending individual imitation controllers. A simple technique for blending controllers is action interpolation, where the outputs of controllers representing various styles are combined. This technique requires the evaluation of multiple controllers and then composing their output.

Our work proposes a completely different approach that rethinks typical approaches for blending the behaviors of multiple DRL control policies by operating directly in the space of the network weights. By training imitation policies using a policy regularization strategy [Rocca et al. 2025], we find that policies producing similar motions also have similar weights. This allows simple linear interpolation of policy weights for computing new controllers from a set of pre-trained policies. This shifts the paradigm from controller composition to dynamic manipulation of policy weights.

The main contributions of this work are:

- A novel policy network design that leverages linear interpolation in weight space, reducing the need for multiple evaluations per timestep.
- A graph-based weight regularization strategy that ensures similar motions yield similar policy weights, resulting in smooth transitions.
- Obtaining motion variations and smooth transitions through linear interpolation of two or more policies, with partial or total perturbation of their weights.

We showcase our method in an interactive control setting, using a gamepad to drive complex, real-time, responsive motions. We demonstrate that motion and style variation may be efficiently realized by a single policy that replaces the behavior of a mixture of policies. This highlights the practicality of our approach for resource-limited settings such video games, virtual reality, and crowd simulation, where computational efficiency and seamless interactivity are paramount.

2 Related Work

Generating lifelike natural motions for virtual characters in physics-based simulations has been a long-standing goal in computer graphics and related fields. This section reviews existing literature relevant to our work, which focuses on policy controllers for physics-based characters. We categorize related work into several themes: Physics-Based Character Control, Motion Imitation, Blending of Motions and Controllers.

2.1 Physics-Based Character Control and Motion Imitation

Early efforts in physics-based character control involved the design of manual controllers for specific motor skills, often requiring tedious parameter tuning. These early works, such as those by Hodgins et al. [1995], demonstrated key insights into achieving robust locomotion gaits by decomposing control into components like hopping height, torso pitch, and speed. Auslander et al. [1995] further showcased the potential by simulating a variety of human motor skills. Later, researchers aimed to simplify the design process by developing more general control principles, resulting in more robust locomotion controllers [Coros et al. 2009]. The combination of PD servos and balance control strategies proved effective for walking in diverse environments.

Trajectory optimization methods emerged as a promising way to synthesize animation using keyframes and physical principles defined in an objective function [Al Borno et al. 2013; Mordatch et al. 2012]. These methods were extended to human body animation using simplified physical models, motion capture data, and reduced dimension subspaces [da Silva et al. 2008; Safonova and Hodgins 2007]. Unlike trajectory optimization, our approach computes motions adapted to new situations by adapting control policy parameters rather than motion parameters, thereby facilitating the modeling of unanticipated changes and interpolation between related motions.

Imitating human motion data has become a popular approach for creating realistic character animations. Early methods often involved kinematic techniques that leveraged datasets of motion clips [Arikan et al. 2003; Kovar et al. 2002]. Reinforcement learning techniques have also been employed to train physics-based characters to mimic reference motions [Chentanez et al. 2018; Peng et al. 2018]. Peng et al. [2018] presented a framework combining goal-directed reinforcement learning with motion capture data, achieving high-quality and robust imitation of various skills by incorporating a phase-aware policy and techniques like reference state initialization and early termination. However, this method requires a phase variable synchronized with the reference motion, limiting timing adjustments.

Peng et al. [2019] approaches controller composition by factorizing an agent's skills into a collection of primitives that can be activated simultaneously. This approach learns reusable motor primitives during pre-training and composes them multiplicatively to produce a wider range of behaviors without requiring explicit phase labels. Peng et al. [2021] introduced Adversarial Motion Prior (AMP), which used a learned discriminator as a style-reward to encourage characters to adopt general characteristics from a motion dataset without exactly matching any specific motion, allowing characters to perform other tasks using the training motion as a prior. Peng et al. [2022] hierarchically structures this framework by learning reusable motor skills from large, unstructured motion datasets by training a low-level policy together with a latent space representation. A high-level policy is then learned for goal-specific tasks. Dou et al. [2023] built upon this framework by learning a conditional controller, advocating a divide-and-conquer strategy to learn individual skills conditioned on labels, offering explicit control.

Recent works have explored improvements and extensions to motion imitation. Wagener et al. [2022] distilled single-clip experts into a unified policy in the MoCapAct dataset. Tessler et al. [2024] introduced a unified physics-based controller capable of inpainting motions conditioned on multi-modal partial objectives, such as head tracking, text commands, and object interaction, using PD control.

2.2 Blending of Motions and Controllers

Early work in motion blending [Feng et al. 2012] typically interpolates between motion trajectories. Controlling characters to perform a diverse set of skills and seamlessly transition between them is

crucial for interactive applications. An approach more common in physics-based settings involves the interpolation of actions produced by different controllers.

For example, da Silva et al. [2009] introduced Linear Bellman Combination, which produces a new controller by linearly combining the value functions of multiple existing optimal controllers. By leveraging the linearity of a transformed Hamilton–Jacobi–Bellman equation, each expert is weighted according to its relevance to the current state and goal, allowing seamless transitions between distinct skills. Building on the idea of combining experts, Zhang et al. [2018] uses a gating network for blending expert controllers corresponding to different quadruped gaits. Their end-to-end training on unstructured motion-capture data eliminates manual gait labeling and yields responsive, multi-modal quadruped animations. In a similar vein, Won et al. [2020] presented a scalable mixture-of-experts framework, where reference motions are first clustered into specialized controllers, and a learned gating network merges their outputs into a unified policy.

Beyond imitating specific motions, controlling the style of animation is important for creating diverse and engaging characters. Layering full or partial full-body animations from different reference motions is a common blending technique, and Starke et al. [2021] proposed a way of training a gating neural network to blend the output of multiple controllers in a kinematic animation setting.

AMP decouples task specification from style specification by combining a task-specific reward with a task-agnostic style-reward learned through a generative adversarial network (GAN) framework. Li et al. [2022] similarly proposed to use a hierarchical GAN framework to learn motion style transfer from a single reference motion for kinematic characters.

Won et al. [2022] explores physics-based character controllers using conditional VAEs for motion generation. Recent advancements have explored the use of natural language to direct physics-based character animation. Juravsky et al. [2022] presented a system that trains control policies to map from high-level language commands to low-level motor commands, enabling characters to reproduce corresponding skills. PADL uses adversarial imitation learning to reproduce diverse skills and ground them in language. Ren et al. [2023] further advanced this by using a hierarchical diffusion model conditioned on language instructions to generate physics-based character animations, demonstrating robustness to environmental perturbations. While our work does not directly address language-based control, the ability to manipulate policy weights could potentially be integrated with such systems to provide fine-grained control over the generated motions based on language input.

Holden et al. [2017] increased the expressiveness of neural networks for locomotion by dynamically changing network weights as a function of the motion phase, enabling the generation of appropriate locomotion over rough terrain and obstacle avoidance.

Recently Rocca et al. [2025] explored the weight-space of control policies as a representation of motion. By introducing common neighbor policy (CNP) regularization, DRL training is encouraged to produce policies with similar network parameters. A diffusion model trained on a set of policies is used to generate new policies, adapting to novel character morphologies unseen by the model.

Our work builds upon the idea of using the control policy as a representation of motion and introduces a simplified approach to operate directly in the weight-space to efficiently manage and combine diverse skills. We elaborate on the concept of CNP regularization to design a graph-based regularization strategy. This enables simple linear interpolation of policy parameters as a tool for obtaining meaningful variations from pretrained policies without retraining or fine-tuning.

3 Methodology

This work aims to leverage heuristics that enable interpolation within the weight space of motion policies to generate motion variations. We use linear interpolation to produce new policies where

the coefficients of the interpolation determine how much is the contribution of each trained policy. To test the capabilities of the method we sample these coefficients randomly. We assign success and failure criteria for assessing the quality of interpolated policies. In this section, we outline the methodology of our approach by describing our graph-based regularization strategy, as well as interpolation, coefficient sampling, and evaluation methods.

3.1 Policy Regularization Strategy

Rocca et al. [2025] show that using CNP regularization when training with a policy gradient algorithm as Proximal Policy Optimization (PPO) and a Multi-Layer Perceptron (MLP) with ReLU activation can confer local continuity of the weight space. Under the assumption of a smooth manifold, we hypothesize that for sufficiently similar motions, linear interpolation can be a good approximation of the manifold. To facilitate this approximation we extend the CNP to a graph-based regularization choosing the edges according to motion similarity.

We build on the AMP framework [Peng et al. 2021], training a discriminator and an actor network for motion imitation. To apply the same motion prior across different tasks, we incorporate a task-specific goal by defining appropriate rewards. Initially, we train a primary policy, referred to as main policy, starting from random weights without regularization. Subsequent policies are trained with regularization by applying a Gaussian prior to the network parameters, encouraging similarity to the weights of policies trained for related tasks. The overall loss function is defined as:

$$\mathcal{L} = c_{\text{Disc}}\,\mathcal{L}_{\text{Disc}} + c_{\text{Task}}\,\mathcal{L}_{\text{Task}} + \gamma\,\|w - w_{\text{Reg}}\|^2,$$

where:

- w_{Reg} denotes the reference parameters (i.e., the weights and biases) of a selected pretrained policy, used as a prior for regularization. The selection of this policy is determined by the regularization graph. During training, the network parameters w are encouraged to converge toward the reference parameters.
- As for the AMP framework, c_{Disc} is the hyperparameter assigned to the imitation loss $\mathcal{L}_{\text{Disc}}$, and c_{Task} is the hyperparameter assigned to the high-level task loss $\mathcal{L}_{\text{Task}}$, with the constraint $c_{\text{Disc}} + c_{\text{Task}} = 1$.
- γ controls the strength of the penalty for the deviation of the current parameters w from the reference parameters w_{Reg}

The regularization strategy follows a hierarchical graph structure, starting from the main policy and branching into derived policies based on state space overlap, as illustrated in Figure 2.

Policies are trained for motion imitation alongside a target location task. In this task, a target is placed on the x-y plane, and the character is rewarded for moving toward the target and stopping upon reaching it. By maintaining a consistent input structure across policies, linear interpolation becomes feasible while preserving the character's controllability. The reward function can be specialized according to the learned skill, for example, the Idle policy is rewarded for maintaining a stationary stance, regardless of the target location. Get-up is a policy trained to recover from a fall to an upright position; it imitates some MoCap examples and is rewarded for reaching a target Z position of the root bone. Roll is a policy that performs a roll on the floor from an upright position.

3.2 Regularization Graph and Linear interpolation

The regularization graph in figure 2 is based on the idea that similar motions should have similar policies. We define the similarity between two motion clips as the amount of overlap in state distribution. This means that the graph for the regularization strategy can be built in many ways, and we chose one of them. To build the graph 2a, we first train a policy for the Walk motion from

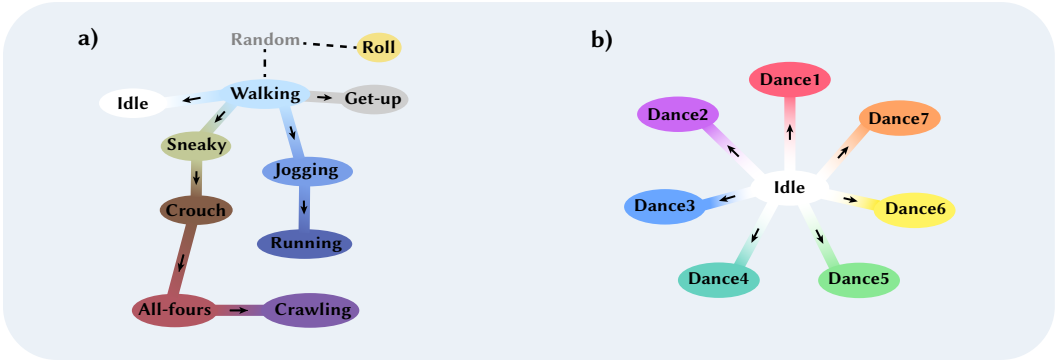

Fig. 2. Graph-based structure for the regularization strategy. a) Graph describing the regularization of policies for typical locomotion. Here, Walking is the main policy, from which the regularization originates. "Random" represents random weight initialization, which differs for Walking and for Rolling in each case., which is different in each case for Walking and Rolling. The dashed line indicates that the training has not been regularized. The arrows represent the direction of the regularization, pointing from the pre-trained to the newly trained policy. For Get-up, the policy has been regularized from Walking, but linear interpolation is not meaningful. b) The Idle policy is a regularizer for seven other different dance policies. Here, similar color does not necessarily mean similar dance.

random weights and without regularization as it is considered our main policy. We train all the other policies following the arrows in the graph. The regularization hinders the training of the Roll motion, however, since it will not be interpolated, we chose not to regularize its training. To build the graph 2b, we assume that Idle can be a regularization policy for all the different dances.

For calculating the linear piecewise interpolation between any policy in the graph, we first need to find a path that connects the two nodes on the graph. Assume we have a path connecting the starting policy π_0 and the target policy π_n through intermediate policies $\pi_1, \pi_2, \ldots, \pi_{n-1}$. The overall interpolation parameter is $s \in [0, 1]$. We split the interval $[0, 1]$ into n equal segments so that each segment corresponds to a link between consecutive policies.

Define the segment index k such that

$$\frac{k}{n} \leq s < \frac{k+1}{n}, \quad k = 0, 1, \ldots, n-1.$$

Then, within the kth segment, we introduce a local interpolation parameter

$$s' = ns - k, \quad s' \in [0, 1].$$

This local parameter s' enables linear blending between successive policies π_k and π_{k+1}. Thus, the interpolated policy $\pi(s)$ is given by:

$$\pi(s) = (1 - s')\pi_k + s'\pi_{k+1}.$$

Expressed in classic piecewise notation, the formula becomes:

$$\pi(s) = \begin{cases} (1 - ns)\pi_0 + (ns)\pi_1, & 0 \leq s < \frac{1}{n}, \\ (1 - (ns - 1))\,\pi_1 + (ns - 1)\pi_2, & \frac{1}{n} \leq s < \frac{2}{n}, \\ \quad \vdots & \\ \big(1 - \big(ns - (n-1)\big)\big)\,\pi_{n-1} + \big(ns - (n-1)\big)\pi_n, & \frac{n-1}{n} \leq s \leq 1. \end{cases}$$

The piecewise interpolation is constructed to approximate the policies on the true weight manifold under the assumption that the local curvature is small.

Interpolation works also for blending three or more policies. This can be useful to create realistic policy variations. This can be achieved by using generalized barycentric coordinates. Assume we have m policies, denoted by $\pi_1, \pi_2, \ldots, \pi_m$, which are the vertices of an $(m-1)$-simplex. Any point inside the simplex can be represented as a convex combination of the vertices:

$$\pi = \lambda_1\pi_1 + \lambda_2\pi_2 + \cdots + \lambda_m\pi_m,$$

where the barycentric coordinates satisfy

$$\lambda_i \geq 0 \quad \text{for } i = 1, 2, \ldots, m, \quad \text{and} \quad \lambda_1 + \lambda_2 + \cdots + \lambda_m = 1.$$

These coefficients can be chosen or randomly sampled from the simplex.

3.3 Sampling Interpolation Coefficients

To extensively test the effect of interpolation across the portion of weight space enclosed by the trained polices, we sample the interpolation coefficients randomly. Uniformly sampling from a multivariate distribution of dimension $m > 3$ would quickly lead to the curse of dimensionality. We are interested in testing the compatibility between interpolated policies and the effect of perturbations in the weight space. For these reasons, we use rejection-based sampling for the coefficients λ. Specifically, we sample the interpolation coefficients uniformly, normalizing samples drawn from a multivariate exponential distribution of dimension m. We retain only those samples that satisfy at least one of the following conditions:

- one policy has a coefficient $\lambda > 0.80$
- two policies have coefficients $\lambda > 0.40$
- three policies have coefficients $\lambda > 0.25$

This ensures that the interpolated policy is either a perturbation of a trained policy or a perturbed combination of at most three policies, simplifying the analysis of the results.

When visualizing the general success and failure of the interpolated policies, we use a different sampling criterion; only in figure 16. We sample triplets of interpolation coefficient $\lambda_1, \lambda_2, \lambda_3$ such that $\lambda_1 + \lambda_2 > 0.8$; which represents a perturbed interpolation along a line. We define criteria for the success and failure of interpolated policies depending on the motion we expect them to represent.

3.4 Evaluation of Policy Quality and Success

Assessing motion quality and overall success is essential when manipulating the parameters of a motion policy. While visual inspection of the motions is often regarded as the definitive evaluation method in computer graphics, we complement this with a quantitative analysis. To facilitate interpretation and provide a concise summary of our findings, we also define specific success criteria that reflect the performance and robustness of the resulting behaviors.

We use Fréchet Motion Distance (FMD) [Maiorca et al. 2022] to quantify the similarity between motions produced by the different policies. For comparing two motion clips, we extract overlapping windows of 34 poses, described as a sequence of directional vectors for every joint. Each patch is encoded into a latent space by means of a pre-trained autoencoder, which we fine-tuned on the same dataset used to train the policies. he patches from the same clip represent a distribution of points in the latent space that is approximated by a Gaussian distribution. The Wasserstein metric is used to calculate the distance between the two Gaussians.

When deciding if an interpolated policy is a success or a failure, we need to take into account the original purpose of the policy. Failure criteria differ based on the motion type: for standing-based

motions, failure is defined as falling, whereas for Crawling or All-fours motions, failure occurs if the final x-y position of the character is less than 2 meters from the starting point. Crawling+Idle and All-fours+Idle represent a particular case; the character is both expected to fall and not move, making their success/failure criteria ambiguous.

4 Experimental Set-up

In designing the experiments to evaluate our method, we define the architectures for both the policy and discriminator networks, select specific hyperparameters for training, and implement a gamepad control logic to translate interactive input into the simulation. In this section, we describe the setup used during the training phases, along with the heuristics and design choices that contributed to the results presented in the following section.

4.1 Policy architecture

The policies are implemented as an MLP with two fully connected hidden layers of sizes [128, 32], using ReLU activations. The input is a 225-dimensional state observation that comprises the root-bone height, local body positions, rotations, velocities, angular velocities, and the relative x-y coordinates of the target location with respect to the character's root bone. The output is a 28-dimensional action vector for the Humanoid character, as described in [Peng et al. 2021]. This output represents the mean of a multivariate Gaussian distribution, with the standard deviation fixed across all dimensions. The discriminator network, also modeled as an MLP with hidden layers of sizes [1024, 512], is used during training to evaluate the imitation quality and during replay to automate the termination of non-interruptible animation policies, such as Get-up or Roll motions.

4.2 Training

To effectively learn motions it is found to be beneficial to divide the training into phases with different combinations of the hyperparameters c_{Disc} and c_{Task}. At the beginning of the training, it is convenient to set $c_{\text{Disc}} = 1$ and consequently $c_{\text{Task}} = 0$, to allow the policy to learn motions that faithfully reproduce the reference data. In this phase, it is important to tune the gradient penalty to enable the learning of complex motions. Once the motion is learned, c_{Task} can be increased to allow the high-level task to be learned. In the final phase, the policy is fine-tuned on a wavy terrain to improve its robustness to falling. The regularization hyperparameter γ is set on a range [0.02,0.05]. All training was executed on Ubuntu 20.04, utilizing an RTX A6000 GPU.

4.3 Gamepad Control

Character control is achieved by leveraging the target location task. Placing the x-y target in the desired direction relative to the character determines the heading direction, and the motion speed is determined by the target's distance. To run our control experiments, we implemented a gamepad control logic to control the character's behavior using Isaac Sim. We map the controller input from the left stick to control the character's direction, and we assign the left and right analog triggers to linear piecewise interpolation paths to control the motion style. Other buttons are programmed to trigger non-interruptible policy switches, such as Roll and Get-up. For these motions, we have designed an automatic termination method by evaluating the discriminator of the Walk policy on a window containing a buffer of previous states. The discriminator outputs a value between -1 and 1, with more positive values indicating that the states are similar to the Walk motion clip, denoting an upright pose. When the output is above an empirical threshold, the policy is switched back to a locomotion policy.

5 Results

The results demonstrate that our control strategy enables smooth transitions between policies, effectively bridging the gap between imitation policies. Figures 3 and 4 illustrate how motion is dynamically interpolated across different styles and how non-interruptible animation policies are seamlessly integrated into the control framework. The method outputs motions that are visually equivalent to motion interpolation. Motion variations can be achieved by interpolating more than two policies according to their compatibility. In this section, we present qualitative and quantitative examples of our experiments.

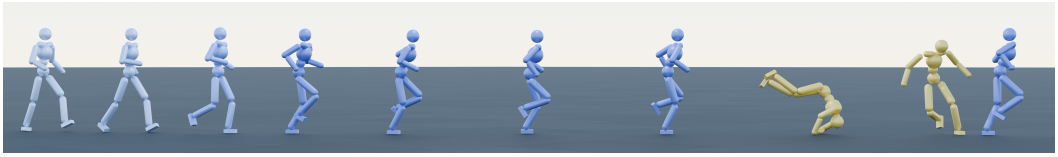

Fig. 3. Linear piecewise interpolation along the graph path from Walk to Running using our method. A roll motion is integrated as a non-interruptible policy and automatically terminated by using the Walk discriminator to detect when the character is back in an upright position.

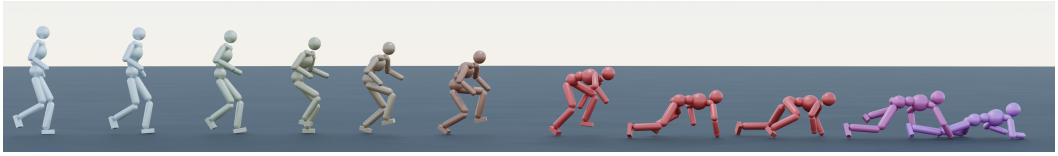

Fig. 4. Linear piecewise interpolation along the graph path from Walk to Crawling using our method. The interpolation between Crouch and All-fours is a stable transition since it does not introduce invalid control policies that lead to unexpected motions. However, it is not smooth nor reversible because of gravity.

5.1 Comparison with Action Interpolation

The linear piecewise interpolation that follows a path on the policy graph shown in Figure 2 generates motions of equivalent visual quality to action interpolation. Interpolation is also possible between policies that are not directly connected in the graph and, in many cases, generalizes to mixing n policies to produce meaningful motion variations. The interpolation of weights can be precomputed at a computational cost of $O(n\,p)$, where p represents the number of network parameters. During inference, evaluating a single policy for each of the c characters incurs a cost of $O(c\,p)$ per timestep, independent of the number of trained policies n in the mix. If continuous variation is required during simulation, interpolation can be computed at each timestep, with its feasibility depending on the trade-off between c and n.

Using the AMP algorithm, our regularization is effective when the state spaces of the motions are similar. For instance, training a chain of policies from Walk to Crawling requires approximately 2,000 epochs with 4,096 environments running in parallel for each regularized training session. These numbers can vary according to the trained motions.

We compare our interpolated policy with the naive interpolation of actions output by the two former networks The visual quality of the motion generated by our method is equivalent to that produced by interpolating the actions. Figure 5 shows descriptive frames from the supplementary video highlighting how similar the two motions are. The table in figure 6 gives a quantitative

comparison between the two methods using FMD. The midpoint interpolation representing an edge of the graph provides low FMD when policy interpolation is compared with action interpolation for the same pair of policies and a high FMD for all the other edges. This quantifies the specific similarity of policy interpolation to action interpolation for the same policy pair.

During inference, evaluating n networks and interpolating the actions costs $O(c\,n\,p)$ per timestep. This shows that our method is more efficient during simulation, with remarkable advantages when simulating a large, diverse crowd or when we want to blend many controllers. Figure 7 shows the increase in computation time when evaluating a policy network on a single thread. One interpolated policy per character can be used to simulate the same number of characters with fewer resources, or more characters before breaking the constraints of a real-time simulation. Depending on the level of optimization of the simulation implementation, the overall time for computing a frame can increase resulting in breaking the real-time constraints with fewer characters. These results describe the uttermost speedup this method can give. In section 6.3, we enter details about the computation times contextual to our instance of implementation.

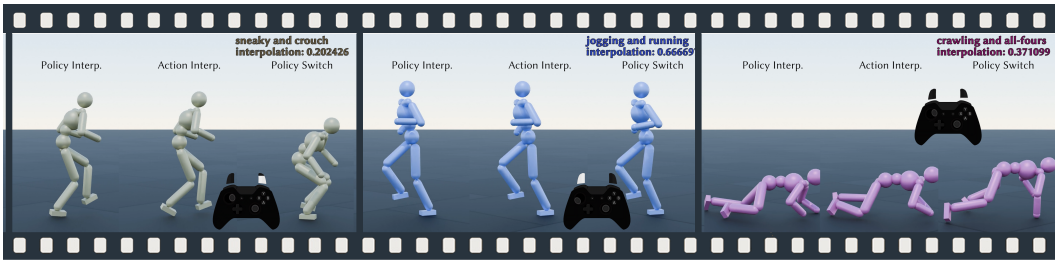

Fig. 5. Illustrative frames from the supplementary video. The gamepad analog triggers are used to interpolate the motion in real-time. Our method produces poses equivalent to direct action interpolation, while direct policy switching fails to ensure smooth transitions.

FMD comparing motions of interpolated policies VS interpolated actions

	a(All-fours, Crawling)	a(Crouch, All-fours)	a(Jogging, Running)	a(Sneaky, Crouch)	a(Walk, Idle)	a(Walk, Jogging)	a(Walk, Sneaky)
p(All-fours, Crawling)	0.1	2.4	2.1	1.9	2.2	2	1.9
p(Crouch, All-fours)	3.2	0.61	2.7	2.5	2.5	2.4	2.3
p(Jogging, Running)	2.4	2.4	0.017	0.45	0.78	0.19	0.26
p(Sneaky, Crouch)	2.1	2.2	0.39	0.02	0.78	0.46	0.25
p(Walk, Idle)	2.5	2.2	0.9	0.95	0.021	0.55	0.5
p(Walk, Jogging)	2.3	2.1	0.2	0.51	0.46	0.0062	0.13
p(Walk, Sneaky)	2.2	2.1	0.25	0.3	0.39	0.1	0.011

Fig. 6. The FMD between motions produced from interpolated policies and interpolated actions shows quantitatively the equivalence in visual quality of the two methods. Here, p(A, B) represents a motion (set of consecutive poses) produced by an interpolated policy with coefficients 0.5 and 0.5. In contrast, a(A, B) is the motion obtained by action interpolation with the same coefficients. Note that (Crouch, All-fours) interpolation does not produce a meaningful policy or actions due to gravity making the two motions not compatible in simulation.

The supplementary video demonstrates that our method achieves visual equivalence to direct action interpolation. Figure 5 highlights how the same gamepad input produces visually consistent poses. This behavior may be attributed to the structure of the policy networks, which are composed of linear layers with ReLU activations. Even though ReLU introduces localized non-linearity by

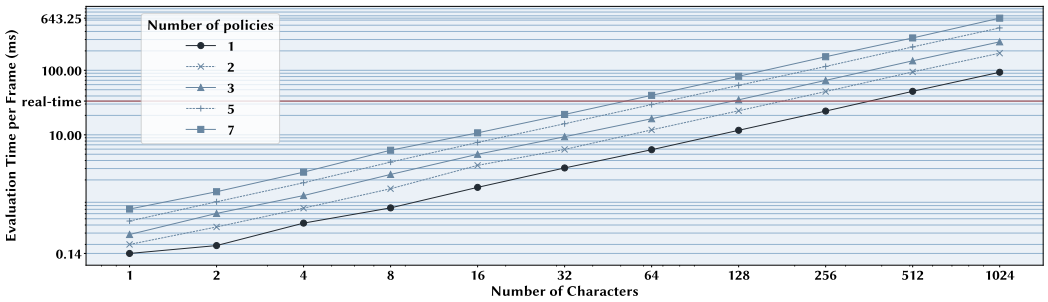

Fig. 7. Log–log plot of the computational time required to evaluate policy networks on a single thread. We compare the performance of a single interpolated policy with that of interpolated actions computed from multiple policies. Precomputing the interpolated policy parameters enables the simulation of a significantly larger number of characters with distinct behaviors, offering a clear advantage over action interpolation in terms of scalability. This plot illustrates the best-case scenario for animation speedup, as it includes only the time required to obtain the action from the controller. In practice, additional operations within the simulation step will introduce further computational overhead, and the actual performance gain will depend on the degree of implementation optimization.

zeroing out negative values, the interpolation remains piecewise linear within regions where the same set of neurons is active. Thinking of the network as function composition, this linear behavior implies that weight-space interpolation operates within regions where linearity is effectively preserved. Supporting this hypothesis, we have observed that alternative architectures with inherently stronger non-linear characteristics, such as Gated Recurrent Units (GRUs) [Xu and Karamouzas 2021; Xu et al. 2023], do not yield meaningful behaviors when their parameters are linearly interpolated. Furthermore, the learned policy manifold may be structured such that intermediate states resulting from interpolation still correspond to valid actions. This could be a consequence of the PPO training process and the regularizing effect of data constraints.

5.2 Variations through interpolation

We test the interpolation between regularized policies by presenting three examples. Interpolating across three, five, and seven policies, respectively.

Tree-Policy Locomotion: Graph a); Walk, Crouch, and Running. These three policies, even if not directly connected in the graph, appear to be compatible policies. We sample uniformly within the triangle of barycentric coordinates, without sample rejection. Figure 1 shows the motion of 100 randomly sampled policies that are all valid and meaningful variations of the three main policies. In Figure 8, we show a selection of motions from the interpolated policies. The quantitative similarity between the motion generated by the interpolated policies and the respective trained counterparts is evaluated in Figure 9, where the FMD score mostly reflects the interpolation coefficients, indicating the proportion of each policy present in the mix.

In this example, the policies appear to be compatible, where every combination within the simplex yields valid and meaningful policies. This is one of the main results that satisfies all the requirements for this technique to be successful. We use this example as a paradigm to analyze failure cases. Inherently, these three policies are trained using motions that have similar pose distribution, have a similar dynamic robustness because of the training of rough terrain, and are trained for the same general task of target location. Realistically, any one of them could serve as

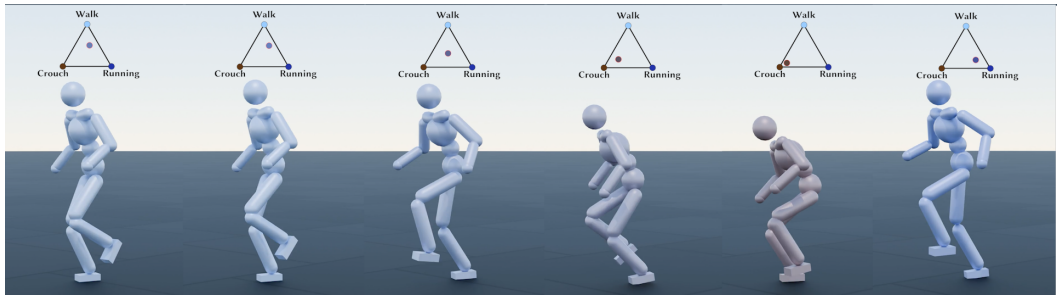

Fig. 8. Motions from interpolated policies within the weight space (Walk, Crouch, Running). Every interpolation among these policies gives valid and meaningful policies as shown in Figure 1 where the interpolation parameters are randomized.

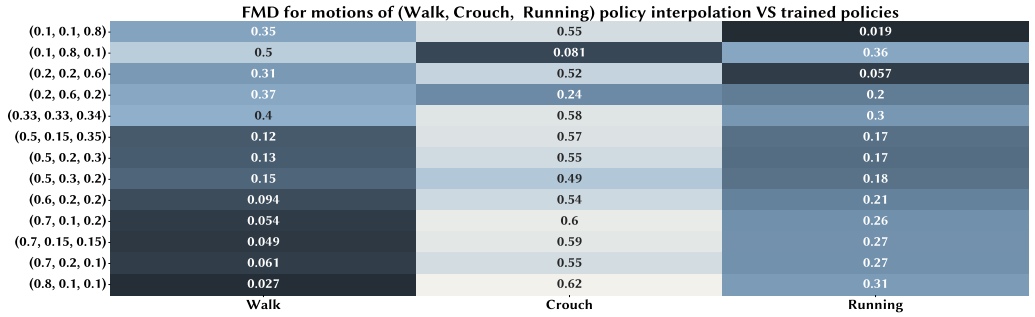

Fig. 9. The FMD of the interpolated policies is compared with the trained policies from which they have been computed. A bigger interpolation coefficient for one trained policy translates to lower FMD for that policy. The few cases in which this correlation does not subsist display that there is not an exact equivalence due to the nature of policy networks.

a regularization policy for the others, leading to successful training. In the other examples, we explore the effect when these characteristics are missing.

Five-Policy Locomotion: Graph a); Walk, Crouch, Run, Crawling, and Idle. The inclusion of Crawling and Idle allows us to explore additional interpolation tests. We have sampled 355 interpolated policies with rejection, of which 74.6% are considered valid. In total, 27.6% had Crawling as a prevalent component. However, 77.5% of policies with Crawling in the mix failed. A selection of perturbed policies is shown in Figure 10, available in the supplementary video too.

In this example, Crawling represents a distinctively different type of motion. Meaningful variations occur only when Crawling is the dominant policy in the interpolation or when it contributes as a minor perturbation alongside the other four policies. This observation suggests that Crawling is not compatible with all possible sets of interpolation coefficients. Figure11 provides a more in-depth understanding of this experiment's statistics. Crawling appears predominant in the failing mix with the other policies and rarely appears alone, confirming that perturbations of it are valid policies. On the other hand, Idle in the interpolation naturally leads to a reduction in motion speed when combined with other upright policies.

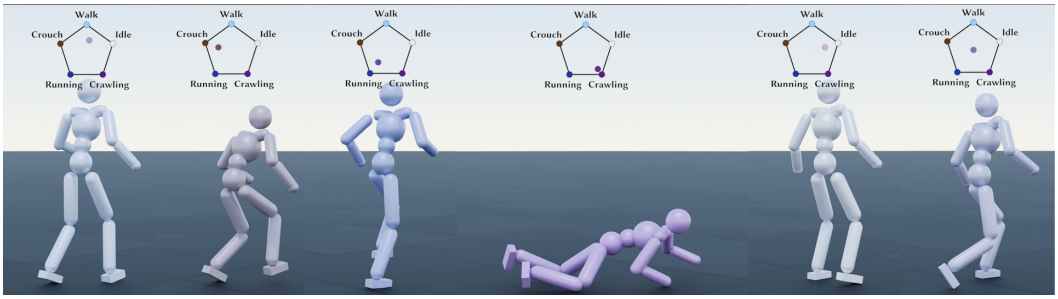

Fig. 10. Valid policies for Walk, Crouch, Running, Crawling, Idle motion variations generated perturbing the weights of the trained policies, through interpolation.

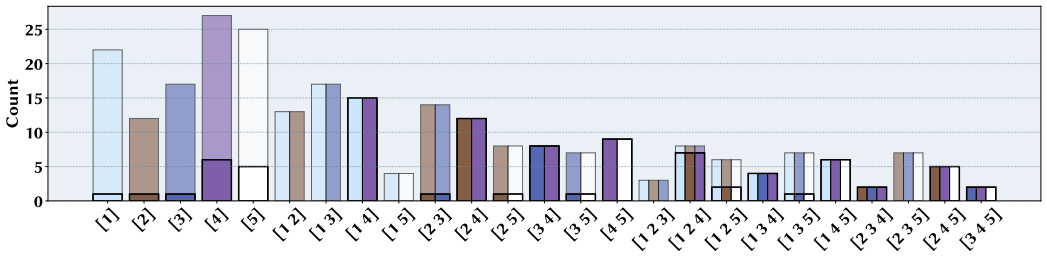

Fig. 11. This picture illustrates the count of policies with a similar configuration of interpolation coefficients among the 355 sampled policies, with the thick line representing the proportion of failed policies. The first group consists of perturbed policies, the second group (represented by two bars) consists of perturbed mixes of two dances, and the last group consists of perturbed mixes of three policies. These groups are described by the sampling criteria in section 5.2. Here, the numbers indicate which of the failing policies are prevalent in the mix in the order 1:Walk, 2:Crouch, 3:Running, 4:Crawling, 5:Idle. A relatively low failure when Crawling is the single prevalent policy shows that compatibility between the motions is a likely cause of failure.

Seven-Policy Dances: Graph b); Dance1 to Dance7. This example investigates the interpolation of different dance styles to generate motion variations. To simplify imitation training, we set $c_{\text{Task}} = 0$, making the policy largely agnostic to the input location. We observe that, for slow-moving dance policies, modifying the target location input has minimal impact on the motion. This effect helps mitigate freezing phenomena, which are characteristic of some very slow motions [Peng et al. 2021]. In this example, we sample either perturbed policies or a perturbed random pair, with rejection. We generated 150 interpolated policies, of which 65.34 % were deemed valid. A total of 66.7% were a mix of two random dances. Notably, only 49% of the mixed dances failed, indicating that half of the interpolations successfully produced stable motions. Here, failure is defined as falling. A selection of perturbed dances is shown in figure 12, and available in the supplementary video.

In this example, the dances have been regularized from a common Idle policy, as shown in figure 2b. The perturbation of a single Dance policy is mostly successful, while not all the mixed dances are. The robustness to small weight perturbations has been shown to be a general property of these types of policies. In figure 13, a more in-depth description of these statistics is given. Dance2 appears in the majority of the failed examples. We attribute this to the dance style relying mostly on one foot, while the other dances rely on two. In general, some dances work better together than others when interpolated, however, no clear rule can be delineated from these results. Among the different explanations for the failure cases, we identify that in contrast to the previous examples,

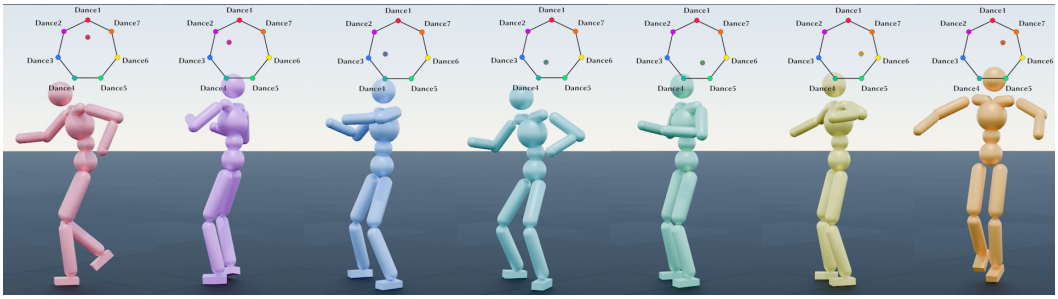

Fig. 12. Motion variations from seven different Dance policies obtained by interpolation.

there is no glaring progression from one type of motion to the other, making it more challenging to design a regularization strategy. The strategy of regularizing the different policies from Idle, although it sounds reasonable, appears to be unable to structure the weight space adequately for interpolation. We speculate that a different choice of motion capture clips and a more progressive regularization strategy can improve these results.

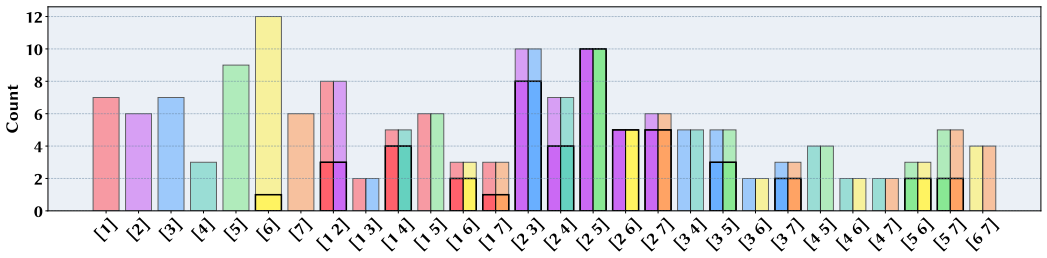

Fig. 13. This picture illustrates the count of policies with a similar configuration of interpolation coefficients among the 150 sampled policies, with the thick line representing the proportion of failed policies. The first group are perturbed policies, and the second one with two bars are perturbed mixes of dances. These groups are described by the sampling criteria in section 5.2. Here, the numbers indicate which of the failing policies Dances 1-7 are prevalent in the mix.

6 Discussion

Our method demonstrates that building a regularization strategy can effectively train similar policies for similar motions, enabling interpolation in the weight space. The success of regularized training and weight interpolation is an intricate matter, as the prerequisites are several, and their description cannot be exhaustive or comprehensive just with this work, opening an interesting investigation path for the future. In this section, we discuss the main aspects that led us to achieve our results, suggestions for replicating or improving the method, and limitations of the technique.

6.1 Considerations on requisites for regularization and interpolation

The requirements for a regularized training to converge successfully are not the same as those that allow meaningful interpolation. The Get-up motion is successfully trained using Walk as a regularization policy; however, interpolating between the two is not meaningful because some of the Get-up states are out-of-distribution for Walk. On the other hand, Walk has been trained on rough terrain, and during simulation, it is robust to perturbations. In this sense, using it as a

regularization for the Get-up policy confers some of these robustness properties upon the character after it stood up.

One of the characteristics of successful regularization and interpolation is the similarity between the visited states, where the input state is non-out-of-distribution for any of the policies involved. Here, we use the term non-out-of-distribution because it would be technically incorrect to classify them as in-distribution. In fact, running a Crouch policy from a standing pose leads the character to crouch instantly (policy switch in figure 5), even if, in principle, it has never seen a fully upright state. Hence, there is a tolerance to out-of-distribution states, which we refer to as compatibility between motions. This means that the set of visited states must exhibit sufficient similarity between the motions in the two training sets for successful regularization and weight interpolation. This is shown in figure 14 where the PCA representation of the state trajectory is similar for policies that interpolate successfully, and figure 15 where regularized policies with similar state distributions cluster closely in PCA representation, while non-regularized policies or those with highly dissimilar state distributions are farther apart. Walk, Sneaky, Crouch, Jogging and Running are compatible policies, as well as All-fours and Crawling. A more intuitive understanding of the effect of compatibility and robustness of policy space interpolation is offered by figure 16. The plot shows the success and failure of a larger sample of 2628 interpolated random triplets of policies. Two of the policies serve as the main interpolation path, consisting of more than 80% of the interpolation blend, while the third one serves as a perturbation. Success and failure were decided according to criteria described in section 3.4.

It was possible to train a regularized policy from Crouch to All-fours, however, it is a special case because the converged All-fours appear to be far from Crouch in the PCA policy weight representation, confirming that the two motions are fundamentally different. The dynamic interpolation between Crouch and All-fours has a critical transition when the character unbalances forward, which is recovered the more the interpolation moves towards All-fours. Also, this dynamic interpolation is not reversible, suggesting that the interpolation might be subject to a hysteresis phenomenon because of the asymmetry introduced by gravity. The geodesic on the weight manifold can take a different path if we want to pass from a crouch to an all-fours type of motion or vice versa; we can speculate that they are two different types of motor skills.

Having the target location as a general task seemed to improve our results, as having a clear common goal, different from pure imitation, helped the policies to take actions that remain compatible during interpolation. This is clear from the examples, especially from *Seven-policy Dances*, where, in opposition to the other two examples, the policies are not rewarded for the target location task to avoid hindering the imitation training. This leads the policies to take very different actions from the same state input, and sometimes incompatible actions that make the simulation fail. In principle, the seven dances are not incompatible, however, these results can be largely improved with a better understanding of regularized policies.

6.2 Choice of regularization strategy

There is flexibility in the choice of the main policy which can be justified under different perspectives. The regularization graph could start from Crouch, considering it to be between Walking and Crawling. Naturally, every regularization strategy introduces pros and cons that, for now, cannot be foreseen without an in-depth study. The regularization graphs in figure 2 are our choices for regularization strategy. Although Idle might appear to represent a natural choice for the main policy, we have chosen Walk because its training involved movement on rough terrain, and this confers dynamic balance properties to the policies that Idle would not have learned. As a consequence, training Idle as a regularization from Walk conserves this balance robustness. Further investigation revealed that Idle and Walk have very similar weight parameters, and their output layer is where

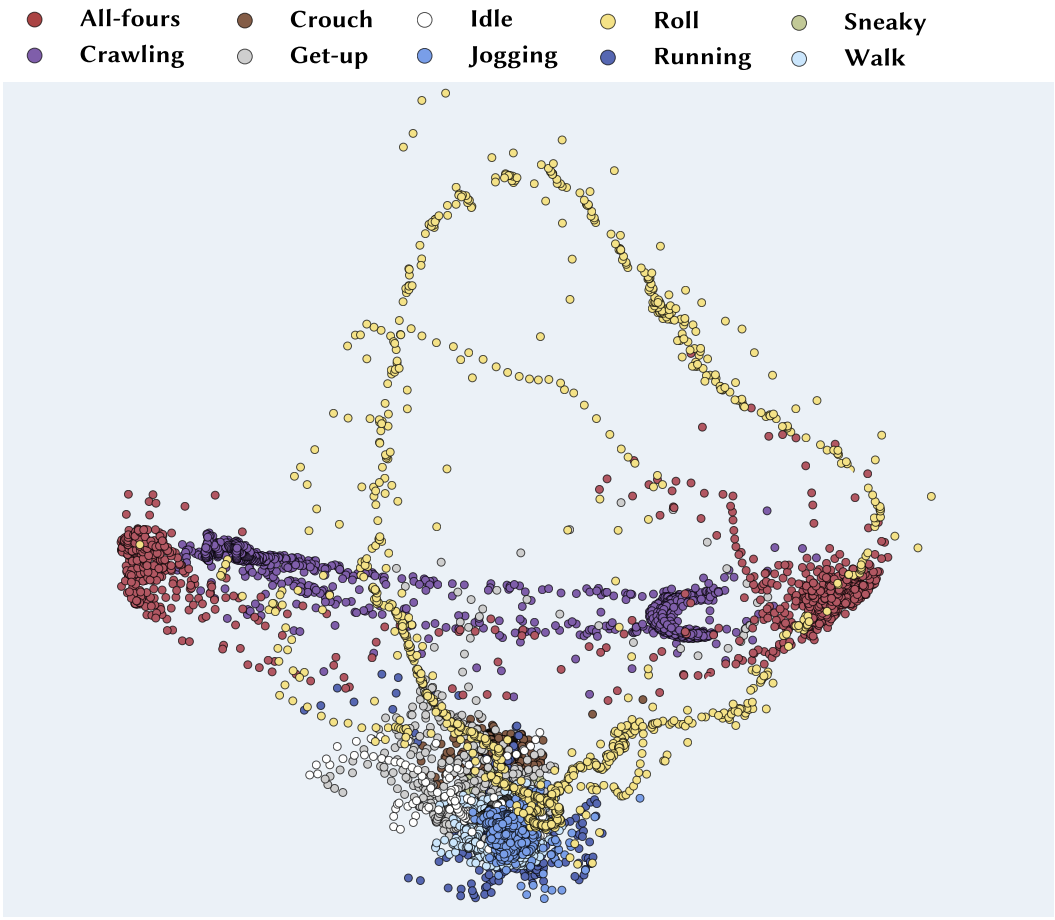

Fig. 14. 3D PCA representation of states visited by the policies. We correlate successful regularized training and interpolated motions with the amount of overlap in the state distribution of each pair of motions. This must be considered when building the regularization strategy as a graph. A policy for the Roll motion cannot be learned effectively using the other policies as a regularizer.

they differ significantly. Our interpretation of this phenomenon is that they process the input similarly until the last layer, where they take different actions, deciding if they are moving or standing still. This is an interesting result that might be worth exploring in future research.

Regularization can sometimes slow or limit the training of specific skills. Regularizing a jump motion from Idle or another locomotion policy can hinder training, as the RL algorithm focuses on matching the jump's starting and landing states, which are already present in the regularizer policy. We attribute this limitation to jumps requiring significantly higher torque than locomotion and the regularization holds it back from moving with such energy. Encouraging jumps by rewarding states with higher Z-positions of the root bone did not help to execute a jump. While other "reward engineering" could potentially enable the character to jump, we did not find such a suitable reward function, so we did not include it in this work. There was an attempt to train Roll, regularizing it from the Walk policy, but penalizing the weights moving further from Walk hindered learning how

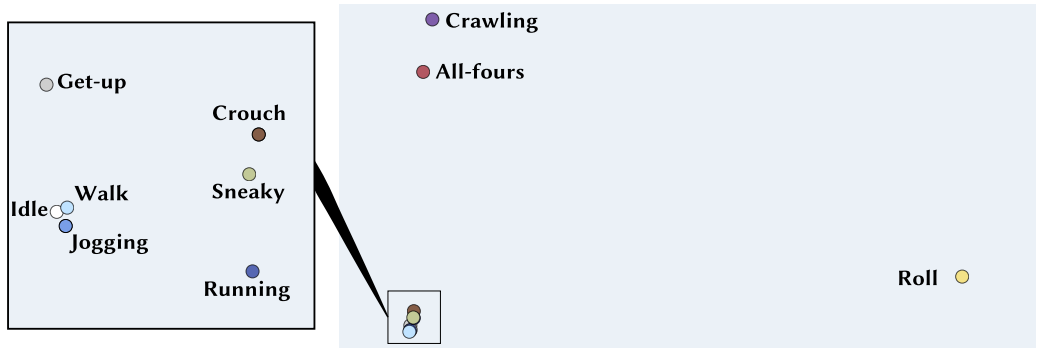

Fig. 15. A 2D PCA representation of policy weights preserving the aspect ratio. Policies close in this reduced space yield realistic results during interpolation. Comparison with figure 14 shows the correlation between weight distances and state distribution overlap. A close-up 3D PCA plot of the cluster of policies around Walk is shown in the framed plot on the right.

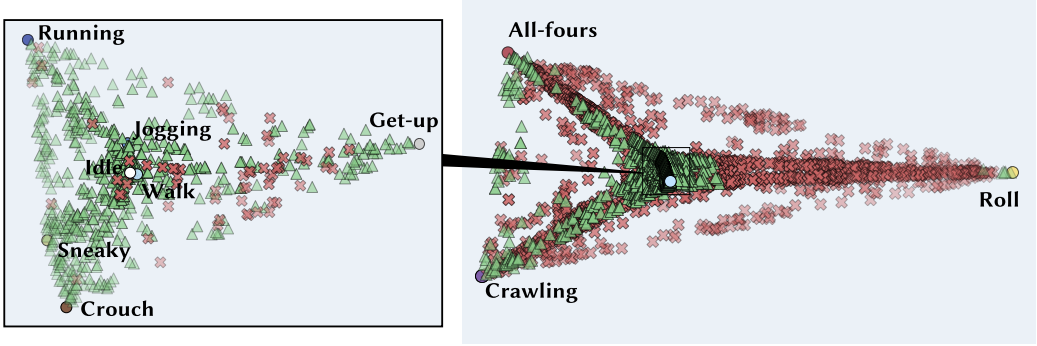

Fig. 16. A 3D PCA plot showing success (triangle in green) and failure (cross in red) of 2.6K randomly interpolated triplets of policies. Two of the policies serve as main path interpolation, and the third one serves as a perturbation. A close-up 3D PCA plot of the cluster of policies around Walk is shown in the framed plot on the left, showing a majority of successful interpolation.

to roll. This is the motivation behind training Roll as an unregularized policy and integrating it as a non-interruptible motion.

6.3 Performance

In the results, we have shown the ideal speedup that could asymptotically be achieved with our method. For completeness, we discuss the performance details of our implementation. In our implementation, a single frame takes an average of 7.3 ms to be computed. A detailed breakdown reveals that the majority of time is not spent on neural network inference. Specifically, computing the observation, which involves gathering simulation data from GPU, calculating the heading, formatting the state to match the policy input, and calculating the target location from the gamepad input rotated contextual to the camera point of view, takes approximately 5.0 ms (about 68% of the total time). Determining the appropriate policy combination based on the control input accounts for around 2.0 ms (27%), and some other operations take another 8% of the computation time. This breakdown highlights that more than 90% of the computation time is consumed by

operations outside of the core policy inference. As a result, although our approach reduces the number of forward passes needed when using interpolated policies, the overall improvement in simulation speed is limited. For instance, when doubling the number of policies per character, the total computation time only improves by about 6%, rather than the theoretical 2× speedup, due to these other dominant costs. Optimizing the measurement of observations and input processing would drastically increase the advantage this technique can offer.

The strengths of weight-space interpolation are evident when the interpolated weights are computed before the simulation begins. Uniquely for the visual comparison with action interpolation, we perform weight interpolation at each timestep, demonstrating that real-time policy switching is feasible, though computationally less convenient. Interpolating the weights at runtime adds approximately 1.2 ms, followed by 0.3 ms for loading the new weights into the model. Importantly, this runtime interpolation is not intrinsic to our method, as all other results rely on precomputed interpolated policies; an approach that is not feasible with action interpolation. Should future research identify a need to adjust some or all of the weights in real time, our work demonstrates that the network can be treated as a dynamic object, thereby making this approach a viable option.

6.4 Variations through extrapolation

We test extrapolation capabilities by selecting points outside the Walk, Crouch, and Running triangle. Figure 17 shows that small extrapolation can still generate motions similar to those generated by trained policies. However, the variations do not appear to be meaningful as they do not introduce recognizable patterns belonging to the policy with a negative coefficient. A stronger extrapolation of about 10% does not produce a valid policy. Remarkably, extrapolation also did not work in the case of Action interpolation. The supplementary video shows this test in detail.

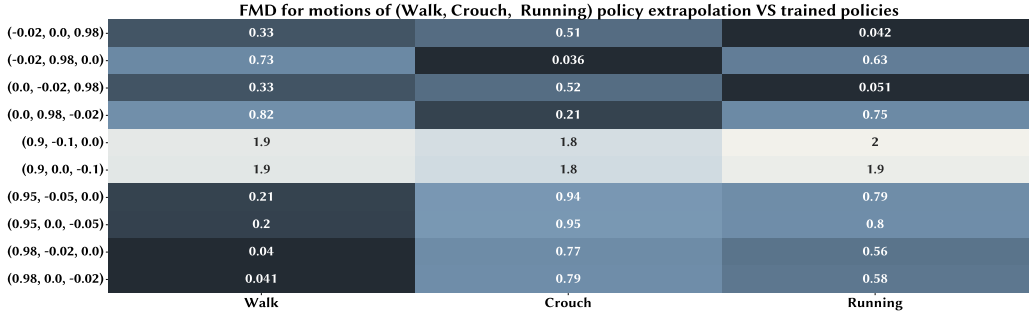

Fig. 17. The FMD of the extrapolated policies is compared with the trained policies they have been computed from. A bigger interpolation coefficient for one trained policy is translated in lower FMD to that policy. Extrapolation works only for small deviations that depend on the policies. For Walk, an extrapolation with coefficient −0.1 in either direction, Crouch or Running, makes the character fall, resulting in a high FMD.

6.5 Interpolation of single layers

The results in the supplementary video and figure 18 show that the interpolation of a single hidden layer or the output layer results in motion variations that do not directly translate into an interpolation of the final action. Instead, these variations indicate that weight interpolation affects the internal representations in a manner that alters the motion trajectory, producing behaviors that deviate from simple action blending. This suggests that the intermediate layers of the network

encode complex, nonlinear transformations that capture the temporal and spatial dynamics of the motion.

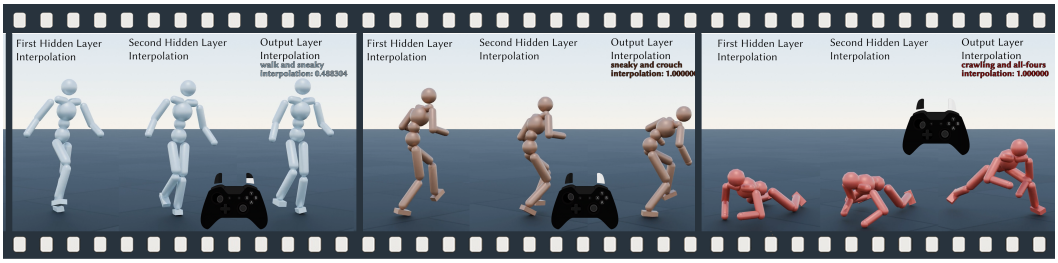

Fig. 18. Interpolation of single layers of the policy network create variations that cannot easily be obtained with action interpolation. The output layer seems to be more sensitive to weight variations. This shows the potentiality of policy weights to represent and manipulate internal motion patterns.

The observed phenomenon highlights the role of the network's architecture in shaping the interpolation outcomes. Interpolating weights in hidden layers influences the latent structure of the policy, which can lead to transitions that are not easily predictable solely from the final action. These results underline the potential of weight manipulation as a tool for generating diverse and expressive motion patterns beyond what is achievable through action interpolation.

The weights w of a control policy can be interpreted as a latent motion encoding, capturing information about the training dataset's visited states, gravity's influence, and robustness to perturbations. During inference, this information is consolidated into a single action, representing the instantaneous policy output. This behavior reflects the realization of a trajectory within a high-dimensional policy manifold. Working at the weight level modifies the internal patterns of the motion, as demonstrated by the interpolation of single layers.

The observed linear characteristics during weight interpolation suggest that the policy manifold has an underlying structure that could support dimensionality reduction. This would enable more efficient and interpretable policy representations. Further analysis is required to understand how this structure generalizes across tasks and architectures and how it can be exploited for practical applications.

7 Conclusion

We propose a novel approach to interpolate motions for physics-based characters. Our results demonstrate that our graph-based regularization strategy conditions the policy weights during training to a localized solution. This enables simple operations in the weight space to vary motion. The comparison with naive action interpolation demonstrates that our method is successful in similar circumstances, and the produced motions are indistinguishable from it, suggesting that the two methods might be approximately equivalent from a mathematical perspective. The regularization strategy seems to organize the weights to preserve linearity from the parameters to the output action. Our approach is advantageous when the simulation of a large number of diverse characters with several interpolated actions is required. Furthermore, interpolating individual layers within the policies generates motion variations, showing that weight manipulation is more expressive than simply blending output motions. This reveals the ability to uncover novel motion trajectories that are inaccessible through naive action interpolation. By representing policies as dynamic objects capable of altering their behavior through weight manipulation, we enable variations across the full motion representation rather than restricting them to contextual actions.

Thus far, this approach has been effectively applied to motions with similar state distributions, consistent with the conditions under which action interpolation achieves realistic results. However, we see promising opportunities to extend this method to a broader range of motions with more diverse state distributions. Different regularization graphs could be investigated in-depth to find useful strategies for advantageously influencing the weight space. This presents a chance to explore new policy representations that enhance the effectiveness and interpretability of weight manipulation, transitioning from ”black-box” systems to a more explainable and versatile representation of motion.

Although action interpolation requires fewer training adjustments, the resulting policies are often dissimilar, preventing an effective study of weights as a continuous representation of motion. We believe that the value of this work lies in improving the understanding of these objects as a continuous variation, leading to a more aware and explainable policy training.

Future research should focus on developing techniques to generalize this approach to more diverse motion datasets while ensuring smooth transitions between motions with dissimilar dynamics. Incorporating domain knowledge could further refine weight manipulation strategies and improve guidance for the process. Additionally, a systematic analysis of how specific groups of weights contribute to motion variations could lead to more precise control over motion style and behavior. Understanding the structure of the policy manifold could also support dimensionality reduction, optimizing storage, computation, and policy interpretation.

Another avenue for exploration is incorporating learning objectives that encourage interpretable latent encodings, fostering a deeper understanding of the relationship between weights, dynamics, and the resulting motion. Finally, integrating these techniques into interactive or real-time systems could unlock applications in animation, robotics, and virtual environments, where adaptability and explainability are critical.

Acknowledgments

The authors would like to thank Victor Zordan for providing valuable feedback and discussions.

References

Mazen Al Borno, Martin de Lasa, and Aaron Hertzmann. 2013. Trajectory Optimization for Full-Body Movements with Complex Contacts. *IEEE Transactions on Visualization and Computer Graphics* 19, 8 (Aug. 2013), 1405–1414. doi:10.1109/TVCG.2012.325

Okan Arikan, David A. Forsyth, and James F. O'Brien. 2003. Motion synthesis from annotations. In *ACM SIGGRAPH 2003 Papers* (San Diego, California) *(SIGGRAPH '03)*. Association for Computing Machinery, New York, NY, USA, 402–408. doi:10.1145/1201775.882284

Joel Auslander, Alex Fukunaga, Hadi Partovi, Jon Christensen, Lloyd Hsu, Peter Reiss, Andrew Shuman, Joe Marks, and J. Thomas Ngo. 1995. Further experience with controller-based automatic motion synthesis for articulated figures. *ACM Trans. Graph.* 14, 4 (Oct. 1995), 311–336. doi:10.1145/225294.225295

Kevin Bergamin, Simon Clavet, Daniel Holden, and James Richard Forbes. 2019. DReCon: data-driven responsive control of physics-based characters. *ACM Trans. Graph.* 38, 6, Article 206 (Nov. 2019), 11 pages. doi:10.1145/3355089.3356536

Nuttapong Chentanez, Matthias Müller, Miles Macklin, Viktor Makoviychuk, and Stefan Jeschke. 2018. Physics-based motion capture imitation with deep reinforcement learning. In *Proceedings of the 11th ACM SIGGRAPH Conference on Motion, Interaction and Games* (Limassol, Cyprus) *(MIG '18)*. Association for Computing Machinery, New York, NY, USA, Article 1, 10 pages. doi:10.1145/3274247.3274506

Stelian Coros, Philippe Beaudoin, and Michiel van de Panne. 2009. Robust task-based control policies for physics-based characters. In *ACM SIGGRAPH Asia 2009 Papers* (Yokohama, Japan) *(SIGGRAPH Asia '09)*. Association for Computing Machinery, New York, NY, USA, Article 170, 9 pages. doi:10.1145/1661412.1618516

Marco da Silva, Yeuhi Abe, and Jovan Popović. 2008. Interactive simulation of stylized human locomotion. In *ACM SIGGRAPH 2008 Papers* (Los Angeles, California) *(SIGGRAPH '08)*. Association for Computing Machinery, New York, NY, USA, Article 82, 10 pages. doi:10.1145/1399504.1360681

Marco da Silva, Frédo Durand, and Jovan Popović. 2009. Linear Bellman combination for control of character animation. *ACM Trans. Graph.* 28, 3, Article 82 (July 2009), 10 pages. doi:10.1145/1531326.1531388

Zhiyang Dou, Xuelin Chen, Qingnan Fan, Taku Komura, and Wenping Wang. 2023. C·ASE: Learning Conditional Adversarial Skill Embeddings for Physics-based Characters. *arXiv preprint arXiv:2309.11351* (2023).

Petros Faloutsos, Michiel van de Panne, and Demetri Terzopoulos. 2001. Composable controllers for physics-based character animation. In *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '01)*. Association for Computing Machinery, New York, NY, USA, 251–260. doi:10.1145/383259.383287

Andrew Feng, Yazhou Huang, Marcelo Kallmann, and Ari Shapiro. 2012. An Analysis of Motion Blending Techniques. In *Motion in Games*, Marcelo Kallmann and Kostas Bekris (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 232–243.

Jessica K. Hodgins, Wayne L. Wooten, David C. Brogan, and James F. O'Brien. 1995. Animating human athletics. In *Proceedings of the 22nd annual conference on Computer graphics and interactive techniques - SIGGRAPH '95 (SIGGRAPH '95)*. ACM Press, 71–78. doi:10.1145/218380.218414

Daniel Holden, Taku Komura, and Jun Saito. 2017. Phase-functioned neural networks for character control. *ACM Trans. Graph.* 36, 4, Article 42 (July 2017), 13 pages. doi:10.1145/3072959.3073663

Jordan Juravsky, Yunrong Guo, Sanja Fidler, and Xue Bin Peng. 2022. PADL: Language-Directed Physics-Based Character Control. In *SA '22 Conference Papers*. doi:10.1145/3550469.3555391

Lucas Kovar, Michael Gleicher, and Frédéric Pighin. 2002. Motion graphs. *ACM Trans. Graph.* 21, 3 (July 2002), 473–482. doi:10.1145/566654.566605

Ariel Kwiatkowski, Eduardo Alvarado, Vicky Kalogeiton, C. Karen Liu, Julien Pettré, Michiel van de Panne, and Marie-Paule Cani. 2022. A Survey on Reinforcement Learning Methods in Character Animation. *Computer Graphics Forum* (2022). doi:10.1111/cgf.14504

Peizhuo Li, Kfir Aberman, Zihan Zhang, Rana Hanocka, and Olga Sorkine-Hornung. 2022. GANimator: neural motion synthesis from a single sequence. *ACM Trans. Graph.* 41, 4, Article 138 (July 2022), 12 pages. doi:10.1145/3528223.3530157

C. Karen Liu, Aaron Hertzmann, and Zoran Popović. 2005. Learning physics-based motion style with nonlinear inverse optimization. *ACM Trans. Graph.* 24, 3 (July 2005), 1071–1081. doi:10.1145/1073204.1073314

Antoine Maiorca, Youngwoo Yoon, and Thierry Dutoit. 2022. Evaluating the Quality of a Synthesized Motion with the Fréchet Motion Distance. In *ACM SIGGRAPH 2022 Posters* (Vancouver, BC, Canada) *(SIGGRAPH '22)*. Association for Computing Machinery, New York, NY, USA, Article 9, 2 pages. doi:10.1145/3532719.3543228

Igor Mordatch, Emanuel Todorov, and Zoran Popović. 2012. Discovery of complex behaviors through contact-invariant optimization. *ACM Trans. Graph.* 31, 4, Article 43 (July 2012), 8 pages. doi:10.1145/2185520.2185539

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), 143. doi:10.1145/3197517.3201311

Xue Bin Peng, Michael Chang, Grace Zhang, Pieter Abbeel, and Sergey Levine. 2019. MCP: Learning Composable Hierarchical Control with Multiplicative Compositional Policies. In *Advances in Neural Information Processing Systems*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett (Eds.), Vol. 32. Curran Associates, Inc. https://proceedings.neurips.cc/paper_files/paper/2019/file/95192c98732387165bf8e396c0f2dad2-Paper.pdf

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 Transactions on Graphics (TOG)* 41, 4 (2022), 94. doi:10.1145/3528223.3530110

Xue Bin Peng, Ze Ma, Pieter Abbeel, Sergey Levine, and Angjoo Kanazawa. 2021. Adversarial Motion Priors for Physics-Based Character Animation. *ACM Transactions on Graphics (TOG)* 40, 4 (2021), 142. doi:10.1145/3450626.3459670

Jiawei Ren, Mingyuan Zhang, Cunjun Yu, Xiao Ma, Liang Pan, and Ziwei Liu. 2023. InsActor: Instruction-driven Physics-based Characters. *NeurIPS* (2023).

Michele Rocca, Sune Darkner, Kenny Erleben, and Sheldon Andrews. 2025. Policy-Space Diffusion for Physics-Based Character Animation. *ACM Trans. Graph.* 44, 3, Article 25 (May 2025), 18 pages. doi:10.1145/3732285

Alla Safonova and Jessica K. Hodgins. 2007. Construction and optimal search of interpolated motion graphs. In *ACM SIGGRAPH 2007 Papers* (San Diego, California) *(SIGGRAPH '07)*. Association for Computing Machinery, New York, NY, USA, 106–es. doi:10.1145/1275808.1276510

Dana Sharon and Michiel van de Panne. 2005. Synthesis of controllers for stylized planar bipedal walking. In *2005 IEEE International Conference on Robotics and Automation*. 1174–1179. doi:10.1109/ROBOT.2005.1570352

Sebastian Starke, Yiwei Zhao, Fabio Zinno, and Taku Komura. 2021. Neural animation layering for synthesizing martial arts movements. *ACM Trans. Graph.* 40, 4, Article 92 (July 2021), 16 pages. doi:10.1145/3450626.3459881

Chen Tessler, Yunrong Guo, Ofir Nabati, Gal Chechik, and Xue Bin Peng. 2024. MaskedMimic: Unified Physics-Based Character Control Through Masked Motion Inpainting. *ACM Transactions on Graphics (TOG)* (2024).

Nolan Wagener, Andrey Kolobov, Felipe Vieira Frujeri, Ricky Loynd, Ching-An Cheng, and Matthew Hausknecht. 2022. MoCapAct: A Multi-Task Dataset for Simulated Humanoid Control. In *Advances in Neural Information Processing Systems*,

Vol. 35. 35418–35431.

Jungdam Won, Deepak Gopinath, and Jessica Hodgins. 2020. Learning Motion Manifolds with Mixture of Experts. *ACM Transactions on Graphics (TOG)* 39, 4 (2020), 33:1–33:13. doi:10.1145/3386569.3392381

Jungdam Won, Deepak Gopinath, and Jessica Hodgins. 2022. Physics-based character controllers using conditional VAEs. *ACM Trans. Graph.* 41, 4, Article 96 (July 2022), 12 pages. doi:10.1145/3528223.3530067

Pei Xu and Ioannis Karamouzas. 2021. A GAN-Like Approach for Physics-Based Imitation Learning and Interactive Character Control. *Proc. ACM Comput. Graph. Interact. Tech.* 4, 3, Article 44 (2021), 44:1–44:22 pages. doi:10.1145/3480148

Pei Xu, Xiumin Shang, Victor Zordan, and Ioannis Karamouzas. 2023. Composite Motion Learning with Task Control. *ACM Trans. Graph.* 42, 4, Article 93 (July 2023), 16 pages. doi:10.1145/3592447

KangKang Yin, Kevin Loken, and Michiel van de Panne. 2007. SIMBICON: Simple Biped Locomotion Control. *ACM Trans. Graph.* 26, 3 (2007), Article 105.

He Zhang, Sebastian Starke, Taku Komura, and Jun Saito. 2018. Mode-adaptive neural networks for quadruped motion control. *ACM Trans. Graph.* 37, 4, Article 145 (July 2018), 11 pages. doi:10.1145/3197517.3201366