Perpetual Humanoid Control for Real-time Simulated Avatars

Zhengyi Luo[1,2] Jinkun Cao[2] Alexander Winkler[1] Kris Kitani[1,2] Weipeng Xu[1]
[1]Reality Labs Research, Meta; [2]Carnegie Mellon University
https://zhengyiluo.github.io/PHC/

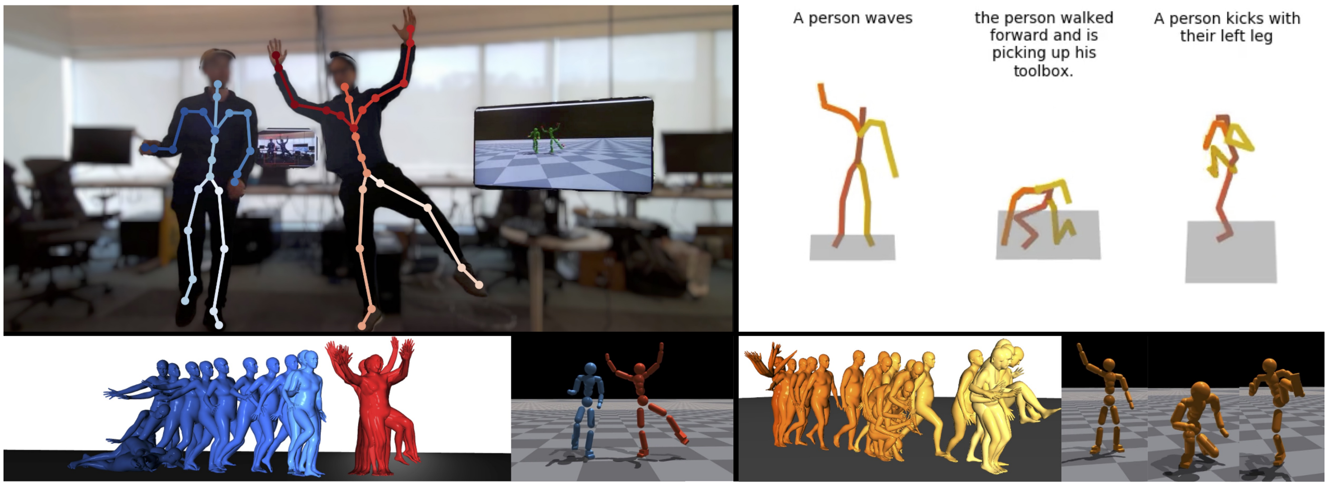

Figure 1: We propose a motion imitator that can naturally recover from falls and walk to far-away reference motion, perpetually controlling simulated avatars without requiring reset. Left: real-time avatars from video, where the blue humanoid recovers from a fall. Right: Imitating 3 *disjoint* clips of motion generated from language, where our controller fills in the blank. The color gradient indicates the passage of time.

Abstract

*We present a physics-based humanoid controller that achieves high-fidelity motion imitation and fault-tolerant behavior in the presence of noisy input (*e.g. *pose estimates from video or generated from language) and unexpected falls. Our controller scales up to learning ten thousand motion clips without using any external stabilizing forces and learns to naturally recover from fail-state. Given reference motion, our controller can perpetually control simulated avatars without requiring resets. At its core, we propose the progressive multiplicative control policy (PMCP), which dynamically allocates new network capacity to learn harder and harder motion sequences. PMCP allows efficient scaling for learning from large-scale motion databases and adding new tasks, such as fail-state recovery, without catastrophic forgetting. We demonstrate the effectiveness of our controller by using it to imitate noisy poses from video-based pose estimators and language-based motion generators in a live and real-time multi-person avatar use case.*

1. Introduction

Physics-based motion imitation has captured the imagination of vision and graphics communities due to its potential for creating realistic human motion, enabling plausible environmental interactions, and advancing virtual avatar technologies of the future. However, controlling high-degree-of-freedom (DOF) humanoids in simulation presents significant challenges, as they can fall, trip, or deviate from their reference motions, and struggle to recover. For example, controlling simulated humanoids using poses estimated from noisy video observations can often lead humanoids to fall to the ground[48, 49, 20, 22]. These limitations prevent the widespread adoption of physics-based methods, as current control policies cannot handle noisy observations such as video or language.

In order to apply physically simulated humanoids for avatars, the first major challenge is learning a motion imitator (controller) that can faithfully reproduce human-like motion with a high success rate. While reinforcement learning (RL)-based imitation policies have shown promising results, successfully imitating motion from a large dataset, such as AMASS (ten thousand clips, 40 hours of motion), with a *single* policy has yet to be achieved. Attempts to use larger or a mixture of expert policies have been met with some success [43, 45], although they have not yet scaled to the largest dataset. Therefore, researchers have resorted to using external forces to help stabilize the humanoid. Residual force control (RFC) [50] has helped to create motion imitators that can mimic up to 97% of the AMASS dataset [20], and has seen successful applications in human pose estimation from video[52, 21, 11] and language-based motion generation [51]. However, the external force compromises physical realism by acting as a "hand of God" that puppets the humanoid, leading to artifacts such as flying and floating. One might argue that, with RFC, the realism of simulation is compromised, as the model can freely apply a non-physical force on the humanoid.

Another important aspect of controlling simulated humanoids is how to handle noisy input and failure cases. In this work, we consider human poses estimated from video or language input. Especially with respect to video input, artifacts such as floating [51], foot sliding [55], and physically impossible poses are prevalent in popular pose estimation methods due to occlusion, challenging view point and lighting, fast motions *etc*. To handle these cases, most physics-based methods resort to resetting the humanoid when a failure condition is triggered [22, 20, 49]. However, resetting successfully requires a high-quality reference pose, which is often difficult to obtain due to the noisy nature of the pose estimates, leading to a vicious cycle of falling and resetting to unreliable poses. Thus, it is important to have a controller that can gracefully handle unexpected falls and noisy input, naturally recover from fail-state, and resume imitation.

In this work, our aim is to create a humanoid controller specifically designed to control real-time virtual avatars, where video observations of a human user are used to control the avatar. We design the Perpetual Humanoid Controller (PHC), a *single* policy that achieves a high success rate on motion imitation **and** can recover from fail-state naturally. We propose a progressive multiplicative control policy (PMCP) to learn from motion sequences in the entire AMASS dataset without suffering catastrophic forgetting. By treating harder and harder motion sequences as a different "task" and gradually allocating new network capacity to learn, PMCP retains its ability to imitate easier motion clips when learning harder ones. PMCP also allows the controller to learn fail-state recovery tasks *without compromising* its motion imitation capabilities. Additionally, we adopt Adversarial Motion Prior (AMP)[33] throughout our pipeline and ensure natural and human-like behavior during fail-state recovery. Furthermore, while most motion imitation methods require both estimates of link position and rotation as input, we show that we can design controllers that require only the link positions. This input can be generated more easily by vision-based 3D keypoint estimators or 3D pose estimates from VR controllers.

To summarize, our contributions are as follows: (1) we propose a Perpetual Humanoid Controller that can successfully imitate 98.9% of the AMASS dataset without applying any external forces; (2) we propose the progressive multiplicative control policy to learn from a large motion dataset without catastrophic forgetting and unlock additional capabilities such as fail-state recovery; (3) our controller is task-agnostic and is compatible with off-the-shelf video-based pose estimators as a drop-in solution. We demonstrate the capabilities of our controller by evaluating on both Motion Capture (MoCap) and estimated motion from videos. We also show a live (30 fps) demo of driving perpetually simulated avatars using a webcam video as input.

2. Related Works

Physics-based Motion Imitation. Governed by the laws of physics, simulated characters [30, 29, 31, 33, 32, 6, 43, 50, 26, 12, 2, 10, 44, 11] have the distinct advantage of creating natural human motion, human-to-human interaction [18, 46], and human-object interactions [26, 32]. Since most modern physics simulators are not differentiable, training these simulated agents requires RL, which is time-consuming & costly. As a result, most of the work focuses on small-scale use cases such as interactive control based on user input [43, 2, 33, 32], playing sports [46, 18, 26], or other modular tasks (reaching goals [47], dribbling [33], moving around [30], *etc*.). On the other hand, imitating large-scale motion datasets is a challenging yet fundamental task, as an agent that can imitate reference motion can be easily paired with a motion generator to achieve different tasks. From learning to imitate a single clip [29] to datasets [45, 43, 6, 42], motion imitators have demonstrated their impressive ability to imitate reference motion, but are often limited to imitating high-quality MoCap data. Among them, ScaDiver [45] uses a mixture of expert policy to scale up to the CMU MoCap dataset and achieves a success rate of around 80% measured by time to failure. Unicon[43] shows qualitative results in imitation and transfer, but does not quantify the imitator's ability to imitate clips from datasets. MoCapAct[42] first learns single-clip experts on the CMU MoCap dataset, and distills them into a single that achieves around 80% of the experts' performance. The effort closest to ours is UHC [20], which successfully imitates 97% of the AMASS dataset. However, UHC uses residual force control [49], which applies a non-physical force at the root of the humanoid to help balance. Although effective in preventing the humanoid from falling, RFC reduces physical realism and creates artifacts such as floating and swinging, especially when motion sequences become challenging [20, 21]. Compared to UHC, our controller does not utilize any external force.

Fail-state Recovery for Simulated Characters. As simulated characters can easily fall when losing balance, many approaches [37, 49, 32, 40, 6] have been proposed to help recovery. PhysCap [37] uses a floating-base humanoid that does not require balancing. This compromises physical realism, as the humanoid is no longer properly simulated. Egopose [49] designs a fail-safe mechanism to reset the humanoid to the kinematic pose when it is about to fall, leading to potential teleport behavior in which the humanoid keeps resetting to unreliable kinematic poses. NeruoMoCon [13] utilizes sampling-based control and reruns the sampling process if the humanoid falls. Although effective, this approach does not guarantee success and prohibits real-time use cases. Another natural approach is to use an additional recovery policy [6] when the humanoid has deviated from the reference motion. However, since such a recovery policy no longer has access to the reference motion, it produces unnatural behavior, such as high-frequency jitters. To combat this, ASE [32] demonstrates the ability to rise naturally from the ground for a sword-swinging policy. While impressive, in motion imitation the policy not only needs to get up from the ground, but also goes back to tracking the reference motion. In this work, we propose a comprehensive solution to the fail-state recovery problem in motion imitation: our PHC can rise from fallen state and naturally walks back to the reference motion and resume imitation.

Progressive Reinforcement Learning. When learning from data containing diverse patterns, catastrophic forgetting [8, 25] is observed when attempting to perform multi-task or transfer learning by fine-tuning. Various approaches [7, 15, 16] have been proposed to combat this phenomenon, such as regularizing the weights of the network [16], learning multiple experts [15], or increasing the capacity using a mixture of experts [54, 36, 45] or multiplicative control [31]. A paradigm has been studied in transfer learning and domain adaption as progressive learning [5, 4] or curriculum learning [1]. Recently, progressive reinforcement learning [3] has been proposed to distill skills from multiple expert policies. It aims to find a policy that best matches the action distribution of experts instead of finding an optimal mix of experts. Progressive Neural Networks (PNN) [34] proposes to avoid catastrophic forgetting by freezing the weights of the previously learned subnetworks and initializing additional subnetworks to learn new tasks. The experiences from previous subnetworks are forwarded through lateral connections. PNN requires manually choosing which subnetwork to use based on the task, preventing it from being used in motion imitation since reference motion does not have the concept of task labels.

3. Method

We define the reference pose as $\hat{\boldsymbol{q}}_t \triangleq (\hat{\boldsymbol{\theta}}_t, \hat{\boldsymbol{p}}_t)$, consisting of 3D joint rotation $\hat{\boldsymbol{\theta}}_t \in \mathbb{R}^{J\times 6}$ and position $\hat{\boldsymbol{p}}_t \in \mathbb{R}^{J\times 3}$ of all J links on the humanoid (we use the 6 DoF rotation representation [53]). From reference poses $\hat{\boldsymbol{q}}_{1:T}$, one can compute the reference velocities $\hat{\dot{\boldsymbol{q}}}_{1:T}$ through finite difference, where $\hat{\dot{\boldsymbol{q}}}_t \triangleq (\hat{\boldsymbol{\omega}}_t, \hat{\boldsymbol{v}}_t)$ consist of angular $\hat{\boldsymbol{\omega}}_t \in \mathbb{R}^{J\times 3}$ and linear velocities $\hat{\boldsymbol{v}}_t \in \mathbb{R}^{J\times 3}$. We differentiate rotation-based and keypoint-based motion imitation by input: rotation-based imitation relies on reference poses $\hat{\boldsymbol{q}}_{1:T}$ (both rotation and keypoints), while keypoint-based imitation only requires 3D keypoints $\hat{\boldsymbol{p}}_{1:T}$. As a notation convention, we use $\tilde{\cdot}$ to represent kinematic quantities (without physics simulation) from pose estimator/keypoint detectors, $\hat{\cdot}$ to denote ground truth quantities from Motion Capture (MoCap), and normal symbols without accents for values from the physics simulation. We use "imitate", "track", and "mimic" reference motion interchangeably. In Sec.3.1, we first set up the preliminary of our main framework. Sec.3.2 describes our progressive multiplicative control policy to learn to imitate a large dataset of human motion and recover from fail-states. Finally, in Sec.3.3, we briefly describe how we connect our task-agnostic controller to off-the-shelf video pose estimators and generators for real-time use cases.

3.1. Goal Conditioned Motion Imitation with Adversarial Motion Prior

Our controller follows the general framework of goal-conditioned RL (Fig.3), where a goal-conditioned policy π_{PHC} is tasked to imitate reference motion $\hat{\boldsymbol{q}}_{\boldsymbol{1:t}}$ or keypoints $\hat{\boldsymbol{p}}_{1:T}$. Similar to prior work [20, 29], we formulate the task as a Markov Decision Process (MDP) defined by the tuple $\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, \mathcal{T}, \boldsymbol{\mathcal{R}}, \gamma \rangle$ of states, actions, transition dynamics, reward function, and discount factor. The physics simulation determines state $\boldsymbol{s}_t \in \mathcal{S}$ and transition dynamics $\mathcal{T}$ while our policy π_{PHC} computes per-step action $\boldsymbol{a}_t \in \mathcal{A}$. Based on the simulation state $\boldsymbol{s}_t$ and reference motion $\hat{\boldsymbol{q}}_t$, the reward function $\boldsymbol{\mathcal{R}}$ computes a reward $r_t = \boldsymbol{\mathcal{R}}(\boldsymbol{s}_t, \hat{\boldsymbol{q}}_t)$ as the learning signal for our policy. The policy's goal is to maximize the discounted reward $\mathbb{E}\left[\sum_{t=1}^{T} \gamma^{t-1} r_t\right]$, and we use the proximal policy gradient (PPO) [35] to learn π_{PHC}.

State. The simulation state $\boldsymbol{s}_t \triangleq (\boldsymbol{s}^{\text{p}}_t, \boldsymbol{s}^{\text{g}}_t)$ consists of humanoid proprioception $\boldsymbol{s}^{\text{p}}_t$ and the goal state $\boldsymbol{s}^{\text{g}}_t$. Proprioception $\boldsymbol{s}^{\text{p}}_t \triangleq (\boldsymbol{q}_t, \dot{\boldsymbol{q}}_t, \boldsymbol{\beta})$ contains the 3D body pose $\boldsymbol{q}_t$, velocity $\dot{\boldsymbol{q}}_t$, and (optionally) body shapes $\boldsymbol{\beta}$. When trained with different body shapes, $\boldsymbol{\beta}$ contains information about the length of the limb of each body link [22]. For rotation-based motion imitation, the goal state $\boldsymbol{s}^{\text{g}}_t$ is defined as the difference between the next time step reference quantitives and their simulated counterpart:

$$\boldsymbol{s}^{\text{g-rot}}_t \triangleq (\hat{\boldsymbol{\theta}}_{t+1} \ominus \boldsymbol{\theta}_t, \hat{\boldsymbol{p}}_{t+1} - \boldsymbol{p}_t, \hat{\boldsymbol{v}}_{t+1} - \boldsymbol{v}_t, \hat{\boldsymbol{\omega}}_t - \boldsymbol{\omega}_t, \hat{\boldsymbol{\theta}}_{t+1}, \hat{\boldsymbol{p}}_{t+1})$$

where $\ominus$ calculates the rotation difference. For keypoint-only imtiation, the goal state becomes

$$\boldsymbol{s}^{\text{g-kp}}_t \triangleq (\hat{\boldsymbol{p}}_{t+1} - \boldsymbol{p}_t, \hat{\boldsymbol{v}}_{t+1} - \boldsymbol{v}_t, \hat{\boldsymbol{p}}_{t+1}).$$

All of the above quantities in $\boldsymbol{s}^{\text{g}}_t$ and $\boldsymbol{s}^{\text{p}}_t$ are normalized with respect to the humanoid's current facing direction and root position [47, 20].

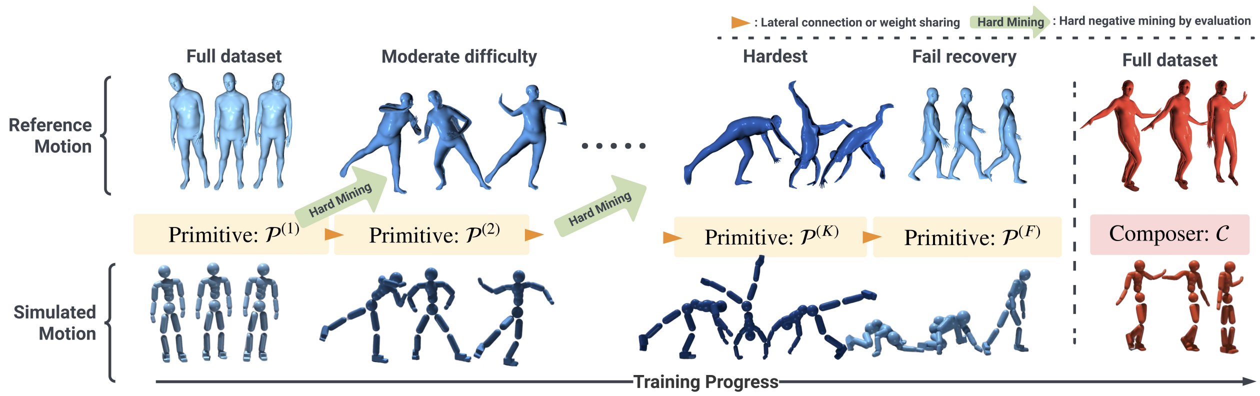

Figure 2: Our progressive training procedure to train primitives $\boldsymbol{\mathcal{P}}^{(1)}, \boldsymbol{\mathcal{P}}^{(2)}, \cdots, \boldsymbol{\mathcal{P}}^{(K)}$ by gradually learning harder and harder sequences. Fail recovery $\boldsymbol{\mathcal{P}}^{(F)}$ is trained in the end on simple locomotion data; a composer is then trained to combine these frozen primitives.

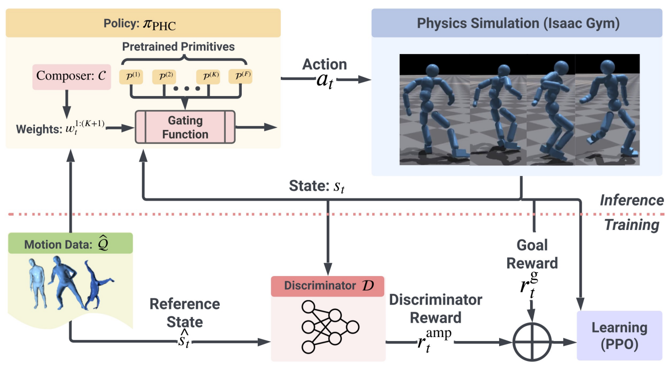

Figure 3: Goal-conditioned RL framework with Adversarial Motion Prior. Each primitive $\boldsymbol{\mathcal{P}}^{(k)}$ and composer $\boldsymbol{\mathcal{C}}$ is trained using the same procedure, and here we visualize the final product π_{PHC}.

Reward. Unlike prior motion tracking policies that only use a motion imitation reward, we use the recently proposed Adversarial Motion Prior [33] and include a discriminator reward term throughout our framework. Including the discriminator term helps our controller produce stable and natural motion and is especially crucial in learning natural fail-state recovery behaviors. Specifically, our reward is defined as the sum of a task reward r_t^{g}, a style reward r_t^{amp}, and an additional energy penalty r_t^{energy} [29]:

$$r_t = 0.5 r_t^{\text{g}} + 0.5 r_t^{\text{amp}} + r_t^{\text{energy}}. \tag{1}$$

For the discriminator, we use the same observations, loss formulation, and gradient penalty as AMP [33]. The energy penalty is expressed as $-0.0005 \cdot \sum_{j \in \text{joints}} |\boldsymbol{\mu_j} \boldsymbol{\omega_j}|^2$ where $\boldsymbol{\mu_j}$ and $\boldsymbol{\omega_j}$ correspond to the joint torque and the joint angular velocity, respectively. The energy penalty [9] regulates the policy and prevents high-frequency jitter of the foot that can manifest in a policy trained without external force (see Sec.4.1). The task reward is defined based on the current training objective, which can be chosen by switching the reward function for motion imitation $\boldsymbol{\mathcal{R}}^{\text{imitation}}$ and fail-state recovery $\boldsymbol{\mathcal{R}}^{\text{recover}}$. For motion tracking, we use:

$$\begin{aligned} r_t^{\text{g-imitation}} &= \boldsymbol{\mathcal{R}}^{\text{imitation}}(\boldsymbol{s}_t, \hat{\boldsymbol{q}}_t) = w_{\text{jp}} e^{-100\|\hat{\boldsymbol{p}}_t - \boldsymbol{p}_t\|} \\ &+ w_{\text{jr}} e^{-10\|\hat{\boldsymbol{q}}_t \ominus \boldsymbol{q}_t\|} + w_{\text{jv}} e^{-0.1\|\hat{\boldsymbol{v}}_t - \boldsymbol{v}_t\|} + w_{\text{j}\omega} e^{-0.1\|\hat{\boldsymbol{\omega}}_t - \boldsymbol{\omega}_t\|} \end{aligned} \tag{2}$$

where we measure the difference between the translation, rotation, linear velocity, and angular velocity of the rigid body for all links in the humanoid. For fail-state recovery, we define the reward $r_t^{\text{g-recover}}$ in Eq.3.

Action. We use a proportional derivative (PD) controller at each DoF of the humanoid and the action $\boldsymbol{a}_t$ specifies the PD target. With the target joint set as $\boldsymbol{q}_t^d = \boldsymbol{a}_t$, the torque applied at each joint is $\boldsymbol{\tau}^i = \boldsymbol{k}^p \circ (\boldsymbol{a}_t - \boldsymbol{q}_t) - \boldsymbol{k}^d \circ \dot{\boldsymbol{q}}_t$. Notice that this is different from the residual action representation [50, 20, 28] used in prior motion imitation methods, where the action is added to the reference pose: $\boldsymbol{q}_t^d = \hat{\boldsymbol{q}}_t + \boldsymbol{a}_t$ to speed up training. As our PHC needs to remain robust to noisy and ill-posed reference motion, we remove such a dependency on reference motion in our action space. We do not use any external forces [50] or meta-PD control[52].

Control Policy and Discriminator. Our control policy $\pi_{\text{PHC}}(\boldsymbol{a}_t|\boldsymbol{s}_t) = \mathcal{N}(\mu(\boldsymbol{s}_t), \sigma)$ represents a Gaussian distribution with fixed diagonal covariance. The AMP discriminator $\boldsymbol{\mathcal{D}}(\boldsymbol{s}_{t-10:t}^{\text{p}})$ computes a real and fake value based on the current prioproception of the humanoid. All of our networks (discriminator, primitive, value function, and discriminator) are two-layer multilayer perceptrons (MLP) with dimensions [1024, 512].

Humanoid. Our humanoid controller can support any human kinematic structure, and we use the SMPL [19] kinematic structure following prior arts [52, 20, 21]. The SMPL body contains 24 rigid bodies, of which 23 are actuated, resulting in an action space of $\boldsymbol{a}_t \in \mathbb{R}^{23 \times 3}$. The body proportion can vary based on a body shape parameter $\beta \in \mathbb{R}^{10}$.

Initialization and Relaxed Early Termination. We use reference state initialization (RSI) [29] during training and randomly select a starting point for a motion clip for imitation. For early termination, we follow UHC [20] and terminate the episode when the joints are more than 0.5 meters globally on average from the reference motion. Unlike UHC, we remove the ankle and toe joints from the termination condition. As observed by RFC [50], there exists a dynamics mismatch between simulated humanoids and real humans, especially since the real human foot is multisegment [27].

Thus, it is not possible for the simulated humanoid to have the exact same foot movement as MoCap, and blindly following the reference foot movement may lead to the humanoid losing balance. Thus, we propose Relaxed Early Termination (RET), which allows the humanoid's ankle and toes to slightly deviate from the MoCap motion to remain balanced. Notice that the humanoid still receives imitation and discriminator rewards for these body parts, which prevents these joints from moving in a nonhuman manner. We show that though this is a small detail, it is conducive to achieving a good motion imitation success rate.

Hard Negative Mining. When learning from a large motion dataset, it is essential to train on harder sequences in the later stages of training to gather more informative experiences. We use a similar hard negative mining procedure as in UHC [20] and define hard sequences by whether or not our controller can successfully imitate this sequence. From a motion dataset $\hat{\boldsymbol{Q}}$, we find hard sequences $\hat{\boldsymbol{Q}}_{\text{hard}} \subseteq \hat{\boldsymbol{Q}}$ by evaluating our model over the entire dataset and choosing sequences that our policy fails to imitate.

3.2. Progressive Multiplicative Control Policy

As training continues, we notice that the performance of the model plateaus as it forgets older sequences when learning new ones. Hard negative mining alleviates the problem to a certain extent, yet suffers from the same issue. Introducing new tasks, such as fail-state recovery, may further degrade imitation performance due to catastrophic forgetting. These effects are more concretely categorized in the Appendix (App. C). Thus, we propose a progressive multiplicative control policy (PMCP), which allocates new subnetworks (primitives $\boldsymbol{\mathcal{P}}$) to learn harder sequences.

Progressive Neural Networks (PNN). A PNN [34] starts with a single primitive network $\boldsymbol{\mathcal{P}}^{(1)}$ trained on the full dataset $\hat{\boldsymbol{Q}}$. Once $\boldsymbol{\mathcal{P}}^{(1)}$ is trained to convergence on the entire motion dataset $\hat{\boldsymbol{Q}}$ using the imitation task, we create a subset of hard motions by evaluating $\boldsymbol{\mathcal{P}}^{(1)}$ on $\hat{\boldsymbol{Q}}$. We define convergence as the success rate on $\hat{\boldsymbol{Q}}^{(k)}_{\text{hard}}$ no longer increases. The sequences that $\boldsymbol{\mathcal{P}}^{(1)}$ fails on is formed as $\hat{\boldsymbol{Q}}^{(1)}_{\text{hard}}$. We then freeze the parameters of $\boldsymbol{\mathcal{P}}^{(1)}$ and create a new primitive $\boldsymbol{\mathcal{P}}^{(2)}$ (randomly initialized) along with lateral connections that connect each layer of $\boldsymbol{\mathcal{P}}^{(1)}$ to $\boldsymbol{\mathcal{P}}^{(2)}$. For more information about PNN, please refer to our supplementary material. During training, we construct each $\hat{\boldsymbol{Q}}^{(k)}_{\text{hard}}$ by selecting the failed sequences from the previous step $\hat{\boldsymbol{Q}}^{(k-1)}_{\text{hard}}$, resulting in a smaller and smaller hard subset: $\hat{\boldsymbol{Q}}^{(k)}_{\text{hard}} \subseteq \hat{\boldsymbol{Q}}^{(k-1)}_{\text{hard}}$. In this way, we ensure that each newly initiated primitive $\boldsymbol{\mathcal{P}}^{(k)}$ is responsible for learning a new and harder subset of motion sequences, as can be seen in Fig.2. Notice that this is different from hard-negative mining in UHC [20], as we initialize a new primitive $\boldsymbol{\mathcal{P}}^{(k+1)}$ to train. Since the original PNN is proposed to solve completely new tasks (such as different Atari games), a lateral connection mechanism is proposed to allow later tasks to choose between reuse, modify, or discard prior experiences. However, mimicking human motion is highly correlated, where fitting to harder sequences $\hat{\boldsymbol{Q}}^{(k)}_{\text{hard}}$ can effectively draw experiences from previous motor control experiences. Thus, we also consider a variant of PNN where there are **no lateral** connections, but the new primitives are initialized from the weights of the prior layer. This weight sharing scheme is similar to fine-tuning on the harder motion sequences using a new primitive $\boldsymbol{\mathcal{P}}^{(k+1)}$ and preserve $\boldsymbol{\mathcal{P}}^{(k)}$'s ability to imitate learned sequences.

Algo 1: Learn Progressive Multiplicative Control Policy

Function `TrainPPO(`π, $\hat{\boldsymbol{Q}}^{(k)}$, $\boldsymbol{\mathcal{D}}$, $\boldsymbol{\mathcal{V}}$, $\boldsymbol{\mathcal{R}}$`)`**:**
    while *not converged* **do**
        $\boldsymbol{M} \leftarrow \emptyset$ initialize sampling memory ;
        while $\boldsymbol{M}$ *not full* **do**
            $\hat{\boldsymbol{q}}_{1:T} \leftarrow$ sample motion from $\hat{\boldsymbol{Q}}$;
            for $t \leftarrow 1...T$ **do**
                $\boldsymbol{s}_t \leftarrow (\boldsymbol{s}^{\text{p}}_t, \boldsymbol{s}^{\text{g}}_t)$;
                $\boldsymbol{a}_t \leftarrow \pi(\boldsymbol{a}_t | \boldsymbol{s}_t)$;
                $\boldsymbol{s}_{t+1} \leftarrow \mathcal{T}(\boldsymbol{s}_{t+1} | \boldsymbol{s}_t, \boldsymbol{a}_t)$ `// simulation;`
                $\boldsymbol{r}_t \leftarrow \boldsymbol{\mathcal{R}}(\boldsymbol{s}_t, \hat{\boldsymbol{q}}_{t+1})$;
                store $(\boldsymbol{s}_t, \boldsymbol{a}_t, \boldsymbol{r}_t, \boldsymbol{s}_{t+1})$ into memory $\boldsymbol{M}$;
        $\boldsymbol{\mathcal{P}}^{(k)}, \boldsymbol{\mathcal{V}} \leftarrow$ PPO update using experiences collected in $\boldsymbol{M}$;
        $\boldsymbol{\mathcal{D}} \leftarrow$ Discriminator update using experiences collected in $\boldsymbol{M}$
    return π ;

Input: Ground truth motion dataset $\hat{\boldsymbol{Q}}$;
$\boldsymbol{\mathcal{D}}, \boldsymbol{\mathcal{V}}, \hat{\boldsymbol{Q}}^{(1)}_{\text{hard}} \leftarrow \hat{\boldsymbol{Q}}$ `// Initialize discriminator, value function, and dataset;`
for $k \leftarrow 1...K$ **do**
    Initialize $\boldsymbol{\mathcal{P}}^{(k)}$`// Lateral connection/weight sharing;`
    $\boldsymbol{\mathcal{P}}^{(k)} \leftarrow$ `TrainPPO(`$\boldsymbol{\mathcal{P}}^{(k)}$, $\hat{\boldsymbol{Q}}^{(k+1)}_{hard}$, $\boldsymbol{\mathcal{D}}$, $\boldsymbol{\mathcal{V}}$, $\boldsymbol{\mathcal{R}}^{imitation}$`)` ;
    $\hat{\boldsymbol{Q}}^{(k+1)}_{\text{hard}} \leftarrow$ eval($\boldsymbol{\mathcal{P}}^{(k)}$, $\hat{\boldsymbol{Q}}^{(k)}$);
    $\boldsymbol{\mathcal{P}}^{(k)} \leftarrow$ freeze $\boldsymbol{\mathcal{P}}^{(k)}$;
$\boldsymbol{\mathcal{P}}^{(F)} \leftarrow$ `TrainPPO(`$\boldsymbol{\mathcal{P}}^{(F)}$, $\boldsymbol{Q}^{loco}$, $\boldsymbol{\mathcal{D}}$, $\boldsymbol{\mathcal{V}}$, $\boldsymbol{\mathcal{R}}^{recover}$`)` `// Fail-state Recovery;`
$\pi_{\text{PHC}} \leftarrow \{\boldsymbol{\mathcal{P}}^{(1)} \cdots \boldsymbol{\mathcal{P}}^{(K)}, \boldsymbol{\mathcal{P}}^{(F)}, \boldsymbol{\mathcal{C}}\}$;
$\pi_{\text{PHC}} \leftarrow$ `TrainPPO(`π_{PHC}, $\hat{\boldsymbol{Q}}$, $\boldsymbol{\mathcal{D}}$, $\boldsymbol{\mathcal{V}}$, $\{\boldsymbol{\mathcal{R}}^{imitation}, \boldsymbol{\mathcal{R}}^{recover}\}$`)` `// Train Composer;`

Fail-state Recovery. In addition to learning harder sequences, we also learn new tasks, such as recovering from fail-state. We define three types of fail-state: 1) fallen on the ground; 2) faraway from the reference motion ($> 0.5m$); 3) their combination: fallen and faraway. In these situations, the humanoid should get up from the ground, approach the reference motion in a natural way, and resume motion imitation. For this new task, we initialize a primitive $\boldsymbol{\mathcal{P}}^{(F)}$ at the end of the primitive stack. $\boldsymbol{\mathcal{P}}^{(F)}$ shares the same input and output space as $\boldsymbol{\mathcal{P}}^{(1)} \cdots \boldsymbol{\mathcal{P}}^{(k)}$, but since the reference motion does not provide useful information about fail-state recovery (the humanoid should not attempt to imitate the reference motion when lying on the ground), we modify the state space during fail-state recovery to remove all information about the reference motion except the root. For the reference joint rotation $\hat{\boldsymbol{\theta}}_t = [\hat{\boldsymbol{\theta}}^0_t, \hat{\boldsymbol{\theta}}^1_t, \cdots \hat{\boldsymbol{\theta}}^J_t]$ where $\hat{\boldsymbol{\theta}}^i_t$ corresponds to the i^{th} joint, we construct $\hat{\boldsymbol{\theta}}'_t = [\hat{\boldsymbol{\theta}}^0_t, \boldsymbol{\theta}^1_t, \cdots \boldsymbol{\theta}^j_t]$ where all joint rotations except the root are replaced with simulated values (without $\hat{\cdot}$). This amounts to setting the non-root joint goals to be identity when computing the goal states: $\boldsymbol{s}^{\text{g-Fail}}_t \triangleq (\hat{\boldsymbol{\theta}}'_t \ominus \boldsymbol{\theta}_t, \hat{\boldsymbol{p}}'_t - \boldsymbol{p}_t, \hat{\boldsymbol{v}}'_t - \boldsymbol{v}_t, \hat{\boldsymbol{\omega}}'_t - \boldsymbol{\omega}_t, \hat{\boldsymbol{\theta}}'_t, \hat{\boldsymbol{p}}'_t)$. $\boldsymbol{s}^{\text{g-Fail}}_t$ thus collapse from an imitation objective to a point-goal [47] objective where the only information provided is the relative position and orientation of the target root. When the reference root is too far ($> 5m$), we normalize $\hat{\boldsymbol{p}}'_t - \boldsymbol{p}_t$ as $\frac{5 \times (\hat{\boldsymbol{p}}'_t - \boldsymbol{p}_t)}{\|\hat{\boldsymbol{p}}'_t - \boldsymbol{p}_t\|_2}$ to clamp the goal position. Once the humanoid is close enough (*e.g.* $< 0.5m$), the goal will switch back to full-motion imitation:

$$\boldsymbol{s}^{\text{g}}_t = \begin{cases} \boldsymbol{s}^{\text{g}}_t & \|\hat{\boldsymbol{p}}^0_t - \boldsymbol{p}^0_t\|_2 \leq 0.5 \\ \boldsymbol{s}^{\text{g-Fail}}_t & \text{otherwise.} \end{cases} \tag{3}$$

To create fallen states, we follow ASE [32] and randomly drop the humanoid on the ground at the beginning of the episode. The

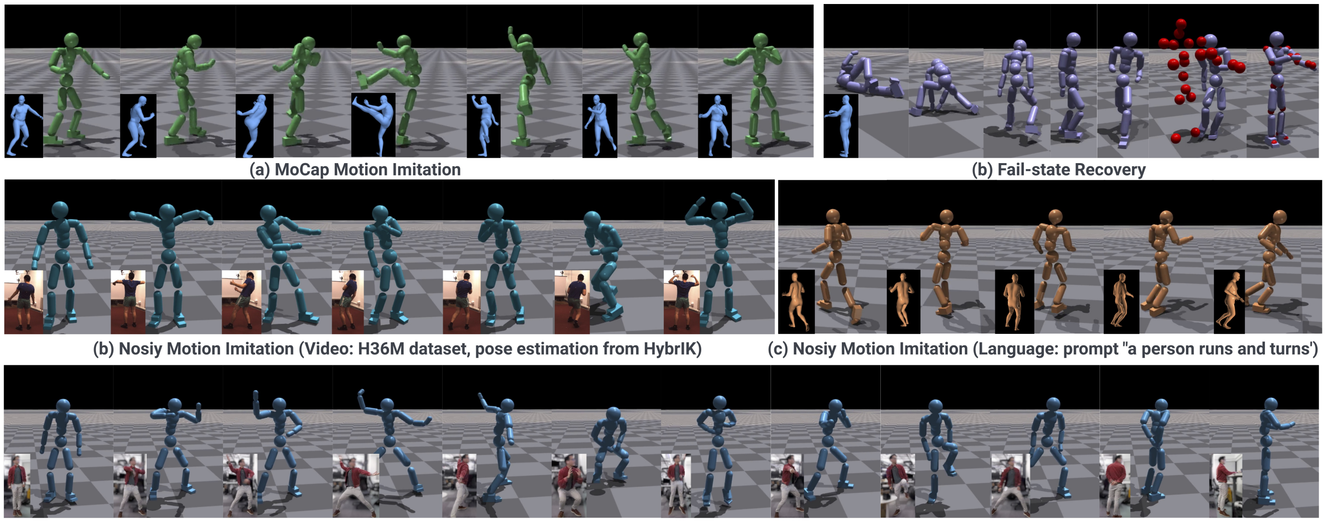

Figure 4: (a) Imitating high-quality MoCap – spin and kick. (b) Recover from fallen state and go back to reference motion (indicated by red dots). (b) Imitating noisy motion estimated from video. (c) Imitating motion generated from language. (d) Using poses estimated from a webcam stream for a real-time simulated avatar.

faraway state can be created by initializing the humanoid $2 \sim 5$ meters from the reference motion. The reward for fail-state recovery consists of the AMP reward r_t^{amp}, point-goal reward $r_t^{\text{g-point}}$, and energy penality r_t^{energy}, calculated by the reward function $\boldsymbol{\mathcal{R}}^{\text{recover}}$:

$$r_t^{\text{g-recover}} = \boldsymbol{\mathcal{R}}^{\text{recover}}(\boldsymbol{s}_t, \hat{\boldsymbol{q}}_t) = 0.5 r_t^{\text{g-point}} + 0.5 r_t^{\text{amp}} + 0.1 r_t^{\text{energy}}, \tag{4}$$

The point-goal reward is formulated as $r_t^{\text{g-point}} = (d_{t-1} - d_t)$ where d_t is the distance between the root reference and simulated root at the time step t [47]. For training $\boldsymbol{\mathcal{P}}^{(F)}$, we use a handpicked subset of the AMASS dataset named $\boldsymbol{Q}^{\text{loco}}$ where it contains mainly walking and running sequences. Learning using only $\boldsymbol{Q}^{\text{loco}}$ coaxes the discriminator $\boldsymbol{\mathcal{D}}$ and the AMP reward r_t^{amp} to bias toward simple locomotion such as walking and running. We do not initialize a new value function and discriminator while training the primitives and continuously fine-tune the existing ones.

Multiplicative Control. Once each primitive has been learned, we obtain $\{\boldsymbol{\mathcal{P}}^{(1)} \cdots \boldsymbol{\mathcal{P}}^{(K)}, \boldsymbol{\mathcal{P}}^{(F)}\}$, with each primitive capable of imitating a subset of the dataset $\hat{\boldsymbol{Q}}$. In Progressive Networks [34], task switching is performed manually. In motion imitation, however, the boundary between hard and easy sequences is blurred. Thus, we utilize Multiplicative Control Policy (MCP) [31] and train an additional composer $\boldsymbol{\mathcal{C}}$ to dynamically combine the learned primitives. Essentially, we use the pretrained primitives as a informed search space for the composer $\boldsymbol{\mathcal{C}}$, and $\boldsymbol{\mathcal{C}}$ only needs to select which primitives to activate for imitation. Specifically, our composer $\boldsymbol{\mathcal{C}}(\boldsymbol{w}_t^{1:K+1} | \boldsymbol{s}_t)$ consumes the same input as the primitives and outputs a weight vector $\boldsymbol{w}_t^{1:K+1} \in \mathbb{R}^{k+1}$ to activate the primitives. Combining our composer and primitives, we have the PHC's output distribution:

$$\pi_{\text{PHC}}(\boldsymbol{a}_t \mid \boldsymbol{s}_t) = \frac{1}{\boldsymbol{\mathcal{C}}(\boldsymbol{s}_t)} \prod_i^k \boldsymbol{\mathcal{P}}^{(i)}(\boldsymbol{a}_t^{(i)} \mid \boldsymbol{s}_t)^{\boldsymbol{\mathcal{C}}(\boldsymbol{s}_t)}, \quad \boldsymbol{\mathcal{C}}(\boldsymbol{s}_t) \geq 0. \tag{5}$$

As each $\boldsymbol{\mathcal{P}}^{(k)}$ is an independent Gaussian, the action distribution:

$$\mathcal{N}\left(\frac{1}{\sum_l^k \frac{\boldsymbol{\mathcal{C}}_l(\boldsymbol{s}_t)}{\sigma_l^j(\boldsymbol{s}_t)}} \sum_i^k \frac{\boldsymbol{\mathcal{C}}_i(\boldsymbol{s}_t)}{\sigma_i^j(\boldsymbol{s}_t)} \mu_i^j(\boldsymbol{s}_t), \sigma^j(\boldsymbol{s}_t) = \left(\sum_i^k \frac{\boldsymbol{\mathcal{C}}_i(\boldsymbol{s}_t)}{\sigma_i^j(\boldsymbol{s}_t)}\right)^{-1}\right), \tag{6}$$

where $\mu_i^j(\boldsymbol{s}_t)$ corresponds to the $\boldsymbol{\mathcal{P}}^{(i)}$'s j^{th} action dimension. Unlike a Mixture of Expert policies that only activates one at a time (top-1 MOE), MCP combines the actors' distribution and activates all actors at the same (similar to top-inf MOE). Unlike MCP, we progressively train our primitives and make the composer and actor share the same input space. Since primitives are independently trained for different harder sequences, we observe that the composite policy sees a significant boost in performance. During composer training, we interleave fail-state recovery training. The training process is described in Alg.1 and Fig.2.

3.3. Connecting with Motion Estimators

Our PHC is task-agnostic as it only requires the next time-step reference pose $\tilde{\boldsymbol{q}}_t$ or the keypoint $\tilde{\boldsymbol{p}}_t$ for motion tracking. Thus, we can use any off-the-shelf video-based human pose estimator or generator compatible with the SMPL kinematic structure. For driving simulated avatars from videos, we employ HybrIK [17] and MeTRAbs [39, 38], both of which estimate in the metric space with the important distinction that HybrIK outputs joint rotation $\tilde{\boldsymbol{\theta}}_t$ while MeTRAbs only outputs 3D keypoints $\tilde{\boldsymbol{p}}_t$. For language-based motion generation, we use the Motion Diffusion Model (MDM) [41]. MDM generates disjoint motion sequences based on prompts, and we use our controller's recovery ability to achieve in-betweening.

4. Experiments

We evaluate and ablate our humanoid controller's ability to imitate high-quality MoCap sequences and noisy motion sequences estimated from videos in Sec.4.1. In Sec.4.2, we test our controller's ability to recovery from fail-state. As motion is best in

Table 1: Quantitative results on imitating MoCap motion sequences (* indicates removing sequences containing human-object interaction). AMASS-Train*, AMASS-Test*, and H36M-Motion* contains 11313, 140, and 140 high-quality MoCap sequences, respectively.

		AMASS-Train*					AMASS-Test*					H36M-Motion*				
Method	RFC	Succ ↑	$E_{\text{g-mpjpe}}$ ↓	E_{mpjpe} ↓	E_{acc} ↓	E_{vel} ↓	Succ ↑	$E_{\text{g-mpjpe}}$ ↓	E_{mpjpe} ↓	E_{acc} ↓	E_{vel} ↓	Succ ↑	$E_{\text{g-mpjpe}}$ ↓	E_{mpjpe} ↓	E_{acc} ↓	E_{vel} ↓
UHC	✓	97.0 %	36.4	25.1	4.4	5.9	96.4 %	50.0	31.2	9.7	12.1	87.0%	59.7	35.4	4.9	7.4
UHC	✗	84.5 %	62.7	39.6	10.9	10.9	62.6%	58.2	98.1	22.8	21.9	23.6%	133.14	67.4	14.9	17.2
Ours	✗	**98.9 %**	**37.5**	**26.9**	**3.3**	**4.9**	96.4%	**47.4**	**30.9**	6.8	**9.1**	92.9%	50.3	**33.3**	3.7	**5.5**
Ours-kp	✗	98.7%	40.7	32.3	3.5	5.5	**97.1%**	53.1	39.5	**7.5**	10.4	**95.7%**	**49.5**	39.2	**3.7**	5.8

Table 2: Motion imitation on noisy motion. We use HybrIK[17] to estimate the joint rotations $\tilde{\boldsymbol{\theta}}_t$ and uses MeTRAbs [39] for global 3D keypoints $\tilde{\boldsymbol{p}}_t$. HybrIK + MeTRAbs (root): using joint rotations $\tilde{\boldsymbol{\theta}}_t$ from HybrIK and root position $\tilde{\boldsymbol{p}}_t^0$ from MeTRAbs. MeTRAbs (all keypoints): using all keypoints $\tilde{\boldsymbol{p}}_t$ from MeTRAbs, only applicabile to our keypoint-based controller.

			H36M-Test-Video*		
Method	RFC	Pose Estimate	Succ ↑	$E_{\text{g-mpjpe}}$ ↓	E_{mpjpe} ↓
UHC	✓	HybrIK + MeTRAbs (root)	58.1%	75.5	49.3
UHC	✗	HybrIK + MeTRAbs (root)	18.1%	126.1	67.1
Ours	✗	HybrIK + MeTRAbs (root)	88.7%	**55.4**	**34.7**
Ours-kp	✗	HybrIK + MeTRAbs (root)	90.0%	55.8	41.0
Ours-kp	✗	MeTRAbs (all keypoints)	**91.9%**	55.7	41.1

videos, we provide extensive qualitative results in the supplementary materials. All experiments are run three times and averaged.

Baselines. We compare with the SOTA motion imitator UHC [20] and use the official implementation. We compare against UHC both *with and without* residual force control.

Implementation Details. We uses four primitives (including fail-state recovery) for all our evaluations. PHC can be trained on a single NVIDIA A100 GPU; it takes around a week to train all primitives and the composer. Once trained, the composite policy runs at $>$ 30 FPS. Physics simulation is carried out in NVIDIA's Isaac Gym [24]. The control policy is run at 30 Hz, while simulation runs at 60 Hz. For evaluation, we do not consider body shape variation and use the mean SMPL body shape.

Datasets. PHC is trained on the training split of the AMASS [23] dataset. We follow UHC [20] and remove sequences that are noisy or involve interactions of human objects, resulting in 11313 high-quality training sequences and 140 test sequences. To evaluate our policy's ability to handle unseen MoCap sequences and noisy pose estimate from pose estimation methods, we use the popular H36M dataset [14]. From H36M, we derive two subsets *H36M-Motion** and *H36M-Test-Video**. H36M-Motion* contains 140 high-quality MoCap sequences from the entire H36M dataset. H36M-Test-Video* contains 160 sequences of noisy poses estimated from videos in the H36M test split (since SOTA pose estimation methods are trained on H36M's training split). * indicates the removal of sequences containing human-chair interaction.

Metrics. We use a series of pose-based and physics-based metrics to evaluate our motion imitation performance. We report the success rate (Succ) as in UHC [20], deeming imitation unsuccessful when, at *any point* during imitation, the body joints are on average $> 0.5m$ from the reference motion. Succ measures whether the humanoid can track the reference motion without losing balance or significantly lags behind. We also report the root-relative mean per-joint position error (MPJPE) E_{mpjpe} and the global MPJPE $E_{\text{g-mpjpe}}$ (in mm), measuring our imitator's ability to imitate the reference motion both locally (root-relative) and globally. To show physical realism, we also compare acceleration E_{acc} (mm/frame^2) and velocity E_{vel} (mm/frame) difference between simulated and MoCap motion. All the baseline and our methods are physically simulated, so we do not report any foot sliding or penetration.

4.1. Motion Imitation

Motion Imitation on High-quality MoCap. Table1 reports our motion imitation result on the AMASS train, test, and H36M-Motion* dataset. Comparing with the baseline **with RFC**, our method outperforms it on almost all metrics across training and test datasets. On the training dataset, PHC has a better success rate while achieving better or similar MPJPE, showcasing its ability to better imitate sequences from the training split. On testing, PHC shows a high success rate on unseen MoCap sequences from both the AMASS and H36M data. Unseen motion poses additional challenges, as can be seen in the larger per-joint error. UHC trained without residual force performs poorly on the test set, showing that it lacks the ability to imitate unseen reference motion. Noticeably, it also has a much larger acceleration error because it uses high-frequency jitter to stay balanced. Compared to UHC, our controller has a low acceleration error even when facing unseen motion sequences, benefiting from the energy penalty and motion prior. Surprisingly, our keypoint-based controller is on par and sometimes outperforms the rotation-based one. This validates that the keypoint-based motion imitator can be a simple and strong alternative to the rotation-based ones.

Motion Imitation on Noisy Input from Video. We use off-the-shelf pose estimators HybrIK [17] and MeTRAbs [39] to extract joint rotation (HybrIK) and keypoints (MeTRAbs) using images from the H36M test set. As a post-processing step, we apply a Gaussian filter to the extracted pose and keypoints. Both HyBrIK and MeTRAbs are per-frame models that do not use any temporal information. Due to depth ambiguity, monocular global pose estimation is highly noisy [39] and suffers from severe depth-wise jitter, posing significant challenge to motion imitators. We find that MeTRAbs outputs better global root estimation $\tilde{\boldsymbol{p}}_t^0$, so we use its $\tilde{\boldsymbol{p}}_t^0$ combined with HybrIK's estimated joint rotation $\tilde{\boldsymbol{\theta}}_t$ (HybrIK + Metrabs (root)). In Table2, we report our controller and baseline's performance on imitating these noisy sequences. Similar to results on MoCap Imitation, PHC outperforms the baselines

Table 3: Ablation on components of our pipeline, performed using noisy pose estimate from HybrIK + Metrabs (root) on the H36M-Test-Video* data. RET: relaxed early termination. MCP: multiplicative control policy. PNN: progressive neural networks.

H36M-Test-Video*							
RET	MCP	PNN	Rotation	Fail-Recover	Succ ↑	$E_{\text{g-mpjpe}}$ ↓	E_{mpjpe} ↓
✗	✗	✗	✓	✗	51.2%	56.2	34.4
✓	✗	✗	✓	✗	59.4%	60.2	37.2
✓	✓	✗	✓	✗	66.2%	59.0	38.3
✓	✓	✓	✓	✗	86.9%	**53.1**	**33.7**
✓	✓	✓	✓	✓	88.7%	55.4	34.7
✓	✓	✓	✗	✓	**90.0%**	55.8	41.0

by a large margin and achieves a high success rate ($\sim 90\%$). This validates our hypothesis that PHC is robust to noisy motion and can be used to drive simulated avatars directly from videos. Similarly, we see that keypoint-based controller (ours-kp) outperforms rotation-based, which can be explained by 1) estimating 3D keypoint directly from images is an easier task than estimating joint rotations, so keypoints from MeTRABs are of higher quality than joint rotations from HybrIK; 2) our keypoint-based controller is more robust to noisy input as it has the freedom to use any joint configuration to try to match the keypoints.

Ablations. Table3 shows our controller trained with various components disabled. We perform ablation on the noisy input from H36M-Test-Image* to better showcase the controller's ability to imitate noisy data. First, we study the performance of our controller before training to recover from fail-state. Comparing row 1 (R1) and R2, we can see that relaxed early termination (RET) allows our policy to better use the ankle and toes for balance. R2 vs R3 shows that using MCP directly without our progressive training process boosts the network performance due to its enlarged network capacity. However, using the PMCP pipeline significantly boosts robustness and imitation performance (R3 vs. R4). Comparing R4 and R5 shows that PMCP is effective in adding fail-state recovery capability **without** compromising motion imitation. Finally, R5 vs. R6 shows that our keypoint-based imitator can be on-par with rotation-based ones, offering a simpler formulation where only keypoints is needed. For additional ablation on MOE vs. MCP, number of primitives, please refer to the supplement.

Real-time Simulated Avatars. We demonstrate our controller's ability to imitate pose estimates streamed in real-time from videos. Fig.4 shows a qualitative result on a live demonstration of using poses estimated from an office environment. To achieve this, we use our keypoint-based controller and MeTRAbs-estimated keypoints in a streaming fashion. The actor performs a series of motions, such as posing and jumping, and our controller can remain stable. Fig.4 also shows our controller's ability to imitate reference motion generated directly from a motion language model MDM [41]. We provide extensive qualitative results in our supplementary materials for our real-time use cases.

4.2. Fail-state Recovery

To evaluate our controller's ability to recover from fail-state, we measure whether our controller can successfully reach the reference motion within a certain time frame. We consider three scenarios: 1) fallen on the ground, 2) far away from reference motion, and 3) fallen and far from reference. We use a single clip of standing-still reference motion during this evaluation. We generate fallen-states by dropping the humanoid on the ground and applying random joint torques for 150 time steps. We create the far-state by initializing the humanoid 3 meters from the reference motion. Experiments are run randomly 1000 trials. From Tab.4 we can see that both of our keypoint-based and rotation-based controllers can recover from fall state with high success rate ($> 90\%$) even in the challenging scenario when the humanoid is both fallen and far away from the reference motion. For a more visual analysis of fail-state recovery, see our supplementary videos.

Table 4: We measure whether our controller can recover from the fail-states by generating these scenarios (dropping the humanoid on the ground & far from the reference motion) and measuring the time it takes to resume tracking.

	Fallen-State		Far-State		Fallen + Far-State	
Method	Succ-5s ↑	Succ-10s ↑	Succ-5s ↑	Succ-10s ↑	Succ-5s ↑	Succ-10s ↑
Ours	95.0%	98.8%	83.7%	99.5%	93.4%	98.8%
Ours-kp	92.5%	94.6%	95.1%	96.0%	79.4%	93.2%

5. Discussions

Limitations. While our purposed PHC can imitate human motion from MoCap and noisy input faithfully, it does not achieve a 100% success rate on the training set. Upon inspection, we find that highly dynamic motions such as high jumping and back flipping are still challenging. Although we can train single-clip controller to **overfit** on these sequences (see the supplement), our full controller often fails to learn these sequences. We hypothesize that learning such highly dynamic clips (together with simpler motion) requires more planning and intent (*e.g.* running up to a high jump), which is not conveyed in the single-frame pose target $\hat{\boldsymbol{q}}_{t+1}$ for our controller. The training time is also long due to our progressive training procedure. Furthermore, to achieve better downstream tasks, the current disjoint process (where the video pose estimator is unaware of the physics simulation) may be insufficient; tighter integration with pose estimation [52, 21] and language-based motion generation [51] is needed.

Conclusion and Future Work. We introduce Perpetual Humanoid Controller, a general purpose physics-based motion imitator that achieves high quality motion imitation while being able to recover from fail-states. Our controller is robust to noisy estimated motion from video and can be used to perpetually simulate a real-time avatar without requiring reset. Future directions include 1) improving imitation capability and learning to imitate 100% of the motion sequences of the training set; 2) incorporating terrain and scene awareness to enable human-object interaction; 3) tighter integration with downstream tasks such as pose estimation and motion generation, *etc*.

Acknowledgements. We thank Zihui Lin for her help in making the plots in this paper. Zhengyi Luo is supported by the Meta AI Mentorship (AIM) program.

References

[1] Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. Curriculum learning. In *Proceedings of the 26th Annual International Conference on Machine Learning*, ICML '09, pages 41–48, New York, NY, USA, June 2009. Association for Computing Machinery.

[2] Kevin Bergamin, Simon Clavet, Daniel Holden, and James Richard Forbes. DReCon: Data-driven responsive control of physics-based characters. *ACM Trans. Graph.*, 38(6):11, 2019.

[3] Glen Berseth, Cheng Xie, Paul Cernek, and Michiel Van de Panne. Progressive reinforcement learning with distillation for multi-skilled motion control. Feb. 2018.

[4] Jinkun Cao, Hongyang Tang, Hao-Shu Fang, Xiaoyong Shen, Cewu Lu, and Yu-Wing Tai. Cross-domain adaptation for animal pose estimation. Aug. 2019.

[5] Chaoqi Chen, Weiping Xie, Wenbing Huang, Yu Rong, Xinghao Ding, Yue Huang, Tingyang Xu, and Junzhou Huang. Progressive feature alignment for unsupervised domain adaptation. Nov. 2018.

[6] Nuttapong Chentanez, Matthias Müller, Miles Macklin, Viktor Makoviychuk, and Stefan Jeschke. Physics-based motion capture imitation with deep reinforcement learning. *Proceedings - MIG 2018: ACM SIGGRAPH Conference on Motion, Interaction, and Games*, 2018.

[7] Matthias De Lange, Rahaf Aljundi, Marc Masana, Sarah Parisot, Xu Jia, Ales Leonardis, Gregory Slabaugh, and Tinne Tuytelaars. A continual learning survey: Defying forgetting in classification tasks. *IEEE Trans. Pattern Anal. Mach. Intell.*, 44(7):3366–3385, July 2022.

[8] Robert M French and Nick Chater. Using noise to compute error surfaces in connectionist networks: a novel means of reducing catastrophic forgetting. *Neural Comput.*, 14(7):1755–1769, July 2002.

[9] Zipeng Fu, Xuxin Cheng, and Deepak Pathak. Deep whole-body control: Learning a unified policy for manipulation and locomotion. Oct. 2022.

[10] Levi Fussell, Kevin Bergamin, and Daniel Holden. SuperTrack: motion tracking for physically simulated characters using supervised learning. *ACM Trans. Graph.*, 40(6):1–13, Dec. 2021.

[11] Kehong Gong, Bingbing Li, Jianfeng Zhang, Tao Wang, Jing Huang, Michael Bi Mi, Jiashi Feng, and Xinchao Wang. PoseTriplet: Co-evolving 3D human pose estimation, imitation, and hallucination under self-supervision. *CVPR*, Mar. 2022.

[12] Leonard Hasenclever, Fabio Pardo, Raia Hadsell, Nicolas Heess, and Josh Merel. CoMic: Complementary task learning & mimicry for reusable skills. http://proceedings.mlr.press/v119/hasenclever20a/hasenclever20a.pdf. Accessed: 2023-2-13.

[13] Buzhen Huang, Liang Pan, Yuan Yang, Jingyi Ju, and Yangang Wang. Neural MoCon: Neural motion control for physically plausible human motion capture. Mar. 2022.

[14] Catalin Ionescu, Dragos Papava, Vlad Olaru, and Cristian Sminchisescu. Human3.6M: Large scale datasets and predictive methods for 3D human sensing in natural environments. *IEEE Trans. Pattern Anal. Mach. Intell.*, 36(7):1325–1339, 2014.

[15] Zhiwei Jia, Xuanlin Li, Zhan Ling, Shuang Liu, Yiran Wu, and Hao Su. Improving policy optimization with generalist-specialist learning. June 2022.

[16] James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, Demis Hassabis, Claudia Clopath, Dharshan Kumaran, and Raia Hadsell. Overcoming catastrophic forgetting in neural networks. *Proc. Natl. Acad. Sci. U. S. A.*, 114(13):3521–3526, 2017.

[17] Jiefeng Li, Chao Xu, Zhicun Chen, Siyuan Bian, Lixin Yang, and Cewu Lu. HybrIK: A hybrid analytical-neural inverse kinematics solution for 3D human pose and shape estimation. Nov. 2020.

[18] Siqi Liu, Guy Lever, Zhe Wang, Josh Merel, S M Ali Eslami, Daniel Hennes, Wojciech M Czarnecki, Yuval Tassa, Shayegan Omidshafiei, Abbas Abdolmaleki, Noah Y Siegel, Leonard Hasenclever, Luke Marris, Saran Tunyasuvunakool, H Francis Song, Markus Wulfmeier, Paul Muller, Tuomas Haarnoja, Brendan D Tracey, Karl Tuyls, Thore Graepel, and Nicolas Heess. From motor control to team play in simulated humanoid football. May 2021.

[19] Matthew Loper, Naureen Mahmood, Javier Romero, Gerard Pons-Moll, and Michael J Black. SMPL: A skinned multi-person linear model. *ACM Trans. Graph.*, 34(6), 2015.

[20] Zhengyi Luo, Ryo Hachiuma, Ye Yuan, and Kris Kitani. Dynamics-regulated kinematic policy for egocentric pose estimation. *NeurIPS*, 34:25019–25032, 2021.

[21] Zhengyi Luo, Shun Iwase, Ye Yuan, and Kris Kitani. Embodied scene-aware human pose estimation. *NeurIPS*, June 2022.

[22] Zhengyi Luo, Ye Yuan, and Kris M Kitani. From universal humanoid control to automatic physically valid character creation. June 2022.

[23] Naureen Mahmood, Nima Ghorbani, Nikolaus F Troje, Gerard Pons-Moll, and Michael J Black. AMASS: Archive of motion capture as surface shapes. *Proceedings of the IEEE International Conference on Computer Vision*, 2019-Octob:5441–5450, 2019.

[24] Viktor Makoviychuk, Lukasz Wawrzyniak, Yunrong Guo, Michelle Lu, Kier Storey, Miles Macklin, David Hoeller, Nikita Rudin, Arthur Allshire, Ankur Handa, and Gavriel State. Isaac gym: High performance GPU-based physics simulation for robot learning. Aug. 2021.

[25] Michael McCloskey and Neal J Cohen. Catastrophic interference in connectionist networks: The sequential learning problem. In Gordon H Bower, editor, *Psychology of Learning and Motivation*, volume 24, pages 109–165. Academic Press, Jan. 1989.

[26] Josh Merel, Saran Tunyasuvunakool, Arun Ahuja, Yuval Tassa, Leonard Hasenclever, Vu Pham, Tom Erez, Greg Wayne, and Nicolas Heess. Catch and carry: Reusable neural controllers for vision-guided whole-body tasks. *ACM Trans. Graph.*, 39(4), 2020.

[27] Hwangpil Park, Ri Yu, and Jehee Lee. Multi-segment foot modeling for human animation. In *Proceedings of the 11th Annual International Conference on Motion, Interaction, and Games*, number Article 16 in MIG '18, pages 1–10, New York, NY, USA, Nov. 2018. Association for Computing Machinery.

[28] Soohwan Park, Hoseok Ryu, Seyoung Lee, Sunmin Lee, and Jehee Lee. Learning predict-and-simulate policies from unorganized human motion data. *ACM Trans. Graph.*, 38(6):1–11, Nov. 2019.

[29] Xue Bin Peng, Pieter Abbeel, Sergey Levine, and Michiel van de Panne. DeepMimic. *ACM Trans. Graph.*, 37(4):1–14, 2018.

[30] Xue Bin Peng, Glen Berseth, Kangkang Yin, and Michiel Van De Panne. DeepLoco: dynamic locomotion skills using hierarchical deep reinforcement learning. *ACM Trans. Graph.*, 36(4):1–13, July 2017.

[31] Xue Bin Peng, Michael Chang, Grace Zhang, Pieter Abbeel, and Sergey Levine. MCP: Learning composable hierarchical control with multiplicative compositional policies. May 2019.

[32] Xue Bin Peng, Yunrong Guo, Lina Halper, Sergey Levine, and Sanja Fidler. ASE: Large-scale reusable adversarial skill embeddings for physically simulated characters. May 2022.

[33] Xue Bin Peng, Ze Ma, Pieter Abbeel, Sergey Levine, and Angjoo Kanazawa. AMP: Adversarial motion priors for stylized physics-based character control. *ACM Trans. Graph.*, (4):1–20, Apr. 2021.

[34] Andrei A Rusu, Neil C Rabinowitz, Guillaume Desjardins, Hubert Soyer, James Kirkpatrick, Koray Kavukcuoglu, Razvan Pascanu, and Raia Hadsell. Progressive neural networks. *arXiv [cs.LG]*, June 2016.

[35] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. Technical report, 2017.

[36] Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc Le, Geoffrey Hinton, and Jeff Dean. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. *arXiv [cs.LG]*, Jan. 2017.

[37] Soshi Shimada, Vladislav Golyanik, Weipeng Xu, and Christian Theobalt. PhysCap: Physically plausible monocular 3D motion capture in real time. (1), Aug. 2020.

[38] István Sárándi, Alexander Hermans, and Bastian Leibe. Learning 3D human pose estimation from dozens of datasets using a geometry-aware autoencoder to bridge between skeleton formats. Dec. 2022.

[39] István Sárándi, Timm Linder, Kai O Arras, and Bastian Leibe. MeTRAbs: Metric-scale truncation-robust heatmaps for absolute 3D human pose estimation. *arXiv*, pages 1–14, 2020.

[40] Tianxin Tao, Matthew Wilson, Ruiyu Gou, and Michiel van de Panne. Learning to get up. *arXiv [cs.GR]*, Apr. 2022.

[41] Guy Tevet, Sigal Raab, Brian Gordon, Yonatan Shafir, Amit H Bermano, and Daniel Cohen-Or. Human motion diffusion model. *arXiv [cs.CV]*, Sept. 2022.

[42] Nolan Wagener, Andrey Kolobov, Felipe Vieira Frujeri, Ricky Loynd, Ching-An Cheng, and Matthew Hausknecht. MoCapAct: A multi-task dataset for simulated humanoid control. Aug. 2022.

[43] Tingwu Wang, Yunrong Guo, Maria Shugrina, and Sanja Fidler. UniCon: Universal neural controller for physics-based character motion. *arXiv*, 2020.

[44] Alexander Winkler, Jungdam Won, and Yuting Ye. QuestSim: Human motion tracking from sparse sensors with simulated avatars. Sept. 2022.

[45] Jungdam Won, Deepak Gopinath, and Jessica Hodgins. A scalable approach to control diverse behaviors for physically simulated characters. *ACM Trans. Graph.*, 39(4), 2020.

[46] Jungdam Won, Deepak Gopinath, and Jessica Hodgins. Control strategies for physically simulated characters performing two-player competitive sports. *ACM Trans. Graph.*, 40(4):1–11, July 2021.

[47] Jungdam Won, Deepak Gopinath, and Jessica Hodgins. Physics-based character controllers using conditional VAEs. *ACM Trans. Graph.*, 41(4):1–12, July 2022.

[48] Ye Yuan and Kris Kitani. 3D ego-pose estimation via imitation learning. In *Computer Vision – ECCV 2018*, volume 11220 LNCS, pages 763–778. Springer International Publishing, 2018.

[49] Ye Yuan and Kris Kitani. Ego-pose estimation and forecasting as real-time PD control. *Proceedings of the IEEE International Conference on Computer Vision*, 2019-Octob:10081–10091, 2019.

[50] Ye Yuan and Kris Kitani. Residual force control for agile human behavior imitation and extended motion synthesis. (NeurIPS), June 2020.

[51] Ye Yuan, Jiaming Song, Umar Iqbal, Arash Vahdat, and Jan Kautz. PhysDiff: Physics-guided human motion diffusion model. *arXiv [cs.CV]*, Dec. 2022.

[52] Ye Yuan, Shih-En Wei, Tomas Simon, Kris Kitani, and Jason Saragih. SimPoE: Simulated character control for 3D human pose estimation. *CVPR*, Apr. 2021.

[53] Yi Zhou, Connelly Barnes, Jingwan Lu, Jimei Yang, and Hao Li. On the continuity of rotation representations in neural networks. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2019-June:5738–5746, 2019.

[54] Yanqi Zhou, Tao Lei, Hanxiao Liu, Nan Du, Yanping Huang, Vincent Zhao, Andrew Dai, Zhifeng Chen, Quoc Le, and James Laudon. Mixture-of-experts with expert choice routing. Feb. 2022.

[55] Yuliang Zou, Jimei Yang, Duygu Ceylan, Jianming Zhang, Federico Perazzi, and Jia-Bin Huang. Reducing footskate in human motion reconstruction with ground contact constraints. In *2020 IEEE Winter Conference on Applications of Computer Vision (WACV)*. IEEE, Mar. 2020.