SimPoE: Simulated Character Control for 3D Human Pose Estimation

Ye Yuan[1]   Shih-En Wei[2]   Tomas Simon[2]   Kris Kitani[1]   Jason Saragih[2]

[1]Carnegie Mellon University   [2]Facebook Reality Labs
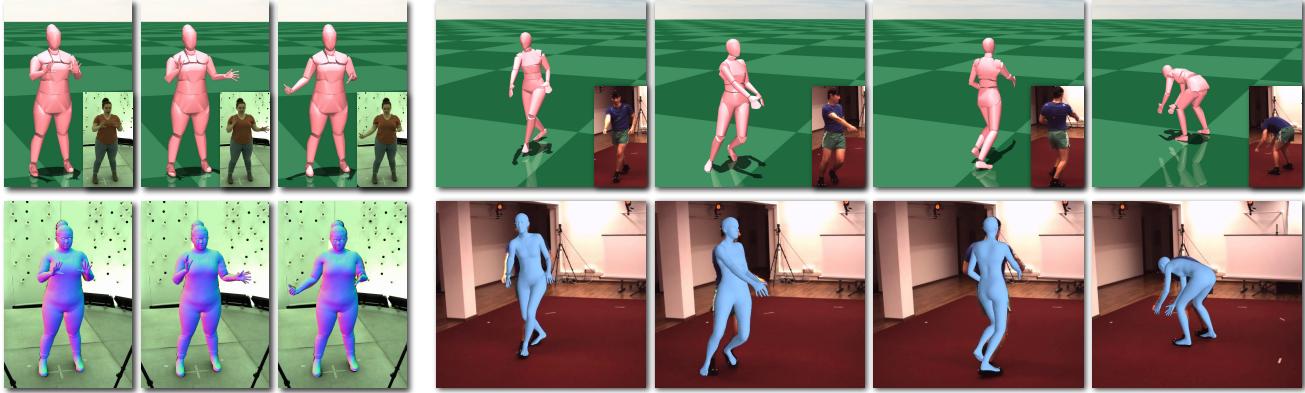
https://www.ye-yuan.com/simpoe

Figure 1. Our SimPoE framework learns a kinematics-aware video-conditioned policy that controls a character in a physics simulator (**Top**) and estimates accurate and physically-plausible human motion (**Bottom**).

Abstract

*Accurate estimation of 3D human motion from monocular video requires modeling both kinematics (body motion without physical forces) and dynamics (motion with physical forces). To demonstrate this, we present SimPoE, a **Sim**ulation-based approach for 3D human **Po**se **E**stimation, which integrates image-based kinematic inference and physics-based dynamics modeling. SimPoE learns a policy that takes as input the current-frame pose estimate and the next image frame to control a physically-simulated character to output the next-frame pose estimate. The policy contains a learnable kinematic pose refinement unit that uses 2D keypoints to iteratively refine its kinematic pose estimate of the next frame. Based on this refined kinematic pose, the policy learns to compute dynamics-based control (e.g., joint torques) of the character to advance the current-frame pose estimate to the pose estimate of the next frame. This design couples the kinematic pose refinement unit with the dynamics-based control generation unit, which are learned jointly with reinforcement learning to achieve accurate and physically-plausible pose estimation. Furthermore, we propose a meta-control mechanism that dynamically adjusts the character's dynamics parameters based on the character state to attain more accurate pose estimates. Experiments on large-scale motion datasets demonstrate that our approach establishes the new state of the art in pose accuracy while ensuring physical plausibility.*

1. Introduction

We aim to show that accurate 3D human pose estimation from monocular video requires modeling both kinematics and dynamics. Human dynamics, *i.e.,* body motion modeling with physical forces, has gained relatively little attention in 3D human pose estimation compared to its counterpart, kinematics, which models motion without physical forces. There are two main reasons for the disparity between these two equally important approaches. First, kinematics is a more direct approach that focuses on the geometric relationships of 3D poses and 2D images; it sidesteps the challenging problem of modeling the physical forces underlying human motion, which requires significant domain knowledge about physics and control. Second, compared to kinematic measurements such as 3D joint positions, physical forces present unique challenges in their measurement and annotation, which renders standard supervised learning paradigms unsuitable. Thus, almost all state-of-the-art methods [38, 63, 22, 21, 35] for 3D human pose estimation from monocular video are based only on kinematics. Although these kinematic methods can estimate human motion with high pose accuracy, they often fail to produce physically-plausible motion. Without modeling the physics of human dynamics, kinematic methods have no notion of force, mass or contact; they also do not have the ability to impose physical constraints such as joint torque limits or friction. As a result, kinematic methods often generate

1

physically-implausible motions with pronounced artifacts: body parts (*e.g.,* feet) penetrate the ground; the estimated poses are jittery and vibrate excessively; the feet slide back and forth when they should be in static contact with the ground. All these physical artifacts significantly limit the application of kinematic pose estimation methods. For instance, jittery motions can be misleading for medical monitoring and sports training; physical artifacts also prevent applications in computer animation and virtual/augmented reality since people are exceptionally good at discerning even the slightest clue of physical inaccuracy [46, 13].

To improve the physical plausibility of estimated human motion from video, recent work [24, 47, 50] has started to adopt the use of dynamics in their formulation. These methods first estimate kinematic motion and then use physics-based trajectory optimization to optimize the forces to induce the kinematic motion. Although they can generate physically-grounded motion, there are several drawbacks of trajectory optimization-based approaches. First, trajectory optimization entails solving a highly-complex optimization problem at test time. This can be computationally intensive and requires the batch processing of a temporal window or even the entire motion sequence, causing high latency in pose predictions and making it unsuitable for interactive real-time applications. Second, trajectory optimization requires simple and differentiable physics models to make optimization tractable, which can lead to high approximation errors compared to advanced and non-differentiable physics simulators (*e.g.,* MuJoCo [54], Bullet [8]). Finally and most importantly, the application of physics in trajectory optimization-based methods is implemented as a post-processing step that projects a given kinematic motion to a physically-plausible one. Since it is optimization-based, there is no learning mechanism in place that tries to match the optimized motion to the ground truth. As such, the resulting motion from trajectory optimization can be physically-plausible but still far from the ground-truth, especially when the input kinematic motion is inaccurate.

To address these limitations, we present a new approach, SimPoE (***Simulated Character Control for Human Pose Estimation***), that tightly integrates image-based kinematic inference and physics-based dynamics modeling into a joint learning framework. Unlike trajectory optimization, SimPoE is a causal temporal model with an integrated physics simulator. Specifically, SimPoE learns a policy that takes the current pose and the next image frame as input, and produces controls for a proxy character inside the simulator that outputs the pose estimate for the next frame. To perform kinematic inference, the policy contains a learnable kinematic pose refinement unit that uses image evidence (2D keypoints) to iteratively refine a kinematic pose estimate. Concretely, the refinement unit takes as input the gradient of keypoint reprojection loss, which encodes rich information about the geometry of pose and keypoints, and outputs the kinematic pose update. Based on this refined kinematic pose, the policy then computes a character control action, *e.g.,* target joint angles for the character's proportional-derivative (PD) controllers, to advance the character state and obtain the next-frame pose estimate. This policy design couples the kinematic pose refinement unit with the dynamics-based control generation unit, which are learned jointly with reinforcement learning (RL) to ensure both accurate and physically-plausible pose estimation. At each time step, a reward is assigned based on the similarity between the estimated motion and the ground truth. To further improve pose estimation accuracy, SimPoE also includes a new control mechanism called meta-PD control. PD controllers are widely used in prior work [44, 41, 65] to convert the action produced by the policy into the joint torques that control the character. However, the PD controller parameters typically have fixed values that require manual tuning, which can produce sub-optimal results. Instead, in meta-PD control, SimPoE's policy is also trained to dynamically adjust the PD controller parameters across simulation steps based on the state of the character to achieve a finer level of control over the character's motion.

We validate our approach, SimPoE, on two large-scale datasets, Human3.6M [15] and an in-house human motion dataset that also contains *detailed finger motion*. We compare SimPoE against state-of-the-art monocular 3D human pose estimation methods including both kinematic and physics-based approaches. On both datasets, SimPoE outperforms previous art in both pose-based and physics-based metrics, with significant pose accuracy improvement over prior physics-based methods. We further conduct extensive ablation studies to investigate the contribution of our proposed components including the kinematic refinement unit, meta-PD control, as well as other design choices.

The main contributions of this paper are as follows: (1) We present a joint learning framework that tightly integrates image-based kinematic inference and physics-based dynamics modeling to achieve accurate and physically-plausible 3D human pose estimation from monocular video. (2) Our approach is causal, runs in real-time without batch trajectory optimization, and addresses several drawbacks of prior physics-based methods. (3) Our proposed meta-PD control mechanism eliminates manual dynamics parameter tuning and enables finer character control to improve pose accuracy. (4) Our approach outperforms previous art in both pose accuracy and physical plausibility. (5) We perform extensive ablations to validate the proposed components to establish good practices for RL-based human pose estimation.

2. Related Work

Kinematic 3D Human Pose Estimation. Numerous prior works estimate 3D human joint locations from monoc-

ular video using either two-stage [9, 45, 40] or end-to-end [30, 29] frameworks. On the other hand, parametric human body models [2, 27, 38] are widely used as the human pose representation since they additionally provide skeletal joint angles and a 3D body mesh. Optimization-based methods have been used to fit the SMPL body model [27] to 2D keypoints extracted from an image [5, 23]. Alternatively, regression-based approaches use deep neural networks to directly regress the parameters of the SMPL model from an image [55, 53, 39, 36, 17, 10], using weak supervision from 2D keypoints [55, 53, 17] or body part segmentation [36, 39]. Song *et al.* [51] propose neural gradient descent to fit the SMPL model using 2D keypoints. Regression-based [17] and optimization-based [5] methods have also been combined to produce pseudo ground truth from weakly-labeled images [22] to facilitate learning. Recent work [3, 14, 18, 52, 21, 28] starts to exploit the temporal structure of human motion to estimate smooth motion. Kanazawa *et al.* [18] model human kinematics by predicting past and future poses. Transformers [56] have also been used to improve the temporal modeling of human motion [52]. All the aforementioned methods disregard human dynamics, *i.e.,* the physical forces that generate human motion. As a result, these methods often produce physically-implausible motions with pronounced physical artifacts such as jitter, foot sliding, and ground penetration.

Physics-Based Human Pose Estimation. A number of works have addressed human dynamics for 3D human pose estimation. Most prior works [6, 61, 58, 67, 65, 47, 50] use trajectory optimization to optimize the physical forces to induce the human motion in a video. As discussed in Sec. 1, trajectory optimization is a batch procedure which has high latency and is typically computationally expensive, making it unsuitable for real-time applications. Furthermore, these methods cannot utilize advanced physics simulators with non-differentiable dynamics. Most importantly, there is no learning mechanism in trajectory optimization-based methods that tries to match the optimized motion to the ground truth. Our approach addresses these drawbacks with a framework that integrates kinematic inference with RL-based character control, which runs in real-time, is compatible with advanced physics simulators, and has learning mechanisms that aim to match the output motion to the ground truth. Although prior work [64, 65, 16] has used RL to produce simple human locomotions from videos, these methods only learn policies that coarsely mimic limited types of motion instead of precisely tracking the motion presented in the video. In contrast, our approach can achieve accurate pose estimation by integrating images-based kinematic inference and RL-based character control with the proposed policy design and meta-PD control.

Reinforcement Learning for Character Control. Deep RL has become the preferred approach for learning character control policies with manually-designed rewards [25, 26, 41, 43]. GAIL [12] based methods are proposed to learn character control without reward engineering [33, 59]. To produce long-term behaviors, prior work has used hierarchical RL to control characters to achieve high-level tasks [32, 31, 42, 34]. Recent work also uses deep RL to learn user-controllable policies from motion capture data for character animation [4, 37, 62]. Prior work in this domain learns control policies that reproduce training motions, but the policies do not transfer to unseen test motions, nor do they estimate motion from video as our method does.

3. Approach

The overview of our SimPoE (***Sim**ulated Character Control for Human **Po**se **E**stimation*) framework is illustrated in Fig. 2. The input to SimPoE is a video $I_{1:T} = (I_1, \ldots, I_T)$ of a person with T frames. For each frame I_t, we first use an off-the-shelf kinematic pose estimator to estimate an initial kinematic pose $\widetilde{q}_t$, which consists of the joint angles and root translation of the person; we also extract 2D keypoints $\check{x}_t$ and their confidence c_t from I_t using a given pose detector (*e.g.,* OpenPose [7]). As the estimated kinematic motion $\widetilde{q}_{1:T} = (\widetilde{q}_1, \ldots, \widetilde{q}_T)$ is obtained without modeling human dynamics, it often contains physically-implausible poses with artifacts like jitter, foot sliding, and ground penetration. This motivates the main stage of our method, *simulated character control*, where we model human dynamics with a proxy character inside a physics simulator. The character's initial pose q_1 is set to $\widetilde{q}_1$. At each time step t shown in Fig. 2 (b), SimPoE learns a policy that takes as input the current character pose q_t, velocities $\dot{q}_t$, as well as the next frame's kinematic pose $\widetilde{q}_{t+1}$ and keypoints $(\check{x}_{t+1}, c_{t+1})$ to produce an action that controls the character in the simulator to output the next pose q_{t+1}. By repeating this causal process, we obtain the physically-grounded estimated motion $q_{1:T} = (q_1, \ldots, q_T)$ of SimPoE.

3.1. Automated Character Creation

The character we use as a proxy to simulate human motion is created from skinned human mesh models, *e.g.,* the SMPL model [27], which can be recovered via SMPL-based pose estimation methods such as VIBE [21]. These skinned mesh models provide a skeleton of B bones, a mesh of V vertices, and a skinning weight matrix $W \in \mathbb{R}^{V \times B}$ where each element W_{ij} specifies the influence of the j-th bone's transformation on the i-th vertex's position. We can obtain a rigid vertex-to-bone association $A \in \mathbb{R}^V$ by assigning each vertex i to the bone with the largest skinning weight for it: $A_i = \arg\max_j W_{ij}$. With the vertex-to-bone association A, we can then create the geometry of each bone by computing the 3D convex hull of all the vertices assigned to the bone. Assuming constant density, the mass of each bone is determined by the volume of its geometry. Our character
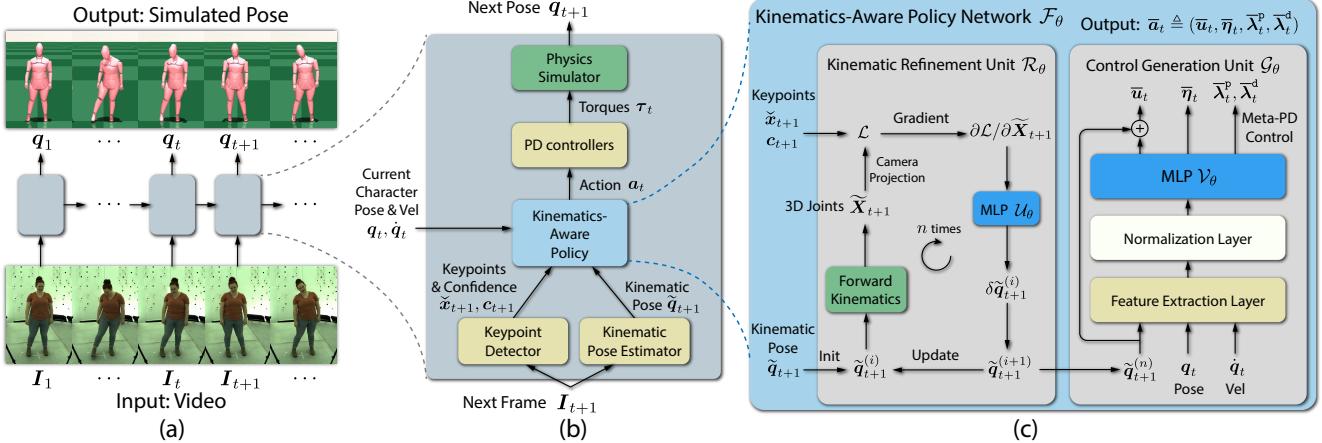
Figure 2. **Overview of our SimPoE framework.** (a) SimPoE is a physics-based causal temporal model. (b) At each frame (30Hz), the policy network $\mathcal{F}_\theta$ use the current pose $\boldsymbol{q}_t$, velocities $\dot{\boldsymbol{q}}_t$, and the next frame's estimated kinematic pose $\widetilde{\boldsymbol{q}}_{t+1}$ and keypoints $(\breve{\boldsymbol{x}}_{t+1}, \boldsymbol{c}_{t+1})$ to generate an action $\boldsymbol{a}_t$, which controls the character in the physics simulator (450Hz) via PD controllers to produce the next pose $\boldsymbol{q}_{t+1}$. (c) The policy network $\mathcal{F}_\theta$ outputs the mean action $\overline{\boldsymbol{a}}_t \triangleq (\overline{\boldsymbol{u}}_t, \overline{\boldsymbol{\eta}}_t, \overline{\boldsymbol{\lambda}}_t^{\mathrm{p}}, \overline{\boldsymbol{\lambda}}_t^{\mathrm{d}})$. The kinematic refinement unit iteratively refines a kinematic pose estimate by learning pose updates. The refined pose $\widetilde{\boldsymbol{q}}_{t+1}^{(n)}$ is used by the control generation unit to produce the mean action $\overline{\boldsymbol{a}}_t$.

creation process is fully automatic, is compatible with popular body mesh models (*e.g.,* SMPL), and ensures proper body geometry and mass assignment.

3.2. Simulated Character Control

The task of controlling a character agent in physics simulation to generate desired human motions can be formulated as a Markov decision process (MDP), which is defined by a tuple $\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathcal{T}, R, \gamma)$ of states, actions, transition dynamics, a reward function, and a discount factor. The character agent interacts with the physics simulator according to a policy $\pi(\boldsymbol{a}_t | \boldsymbol{s}_t)$, which models the conditional distribution of choosing an action $\boldsymbol{a}_t \in \mathcal{A}$ given the current state $\boldsymbol{s}_t \in \mathcal{S}$ of the agent. Starting from some initial state $\boldsymbol{s}_1$, the character agent iteratively samples an action $\boldsymbol{a}_t$ from the policy π and the simulation environment with transition dynamics $\mathcal{T}(\boldsymbol{s}_{t+1} | \boldsymbol{s}_t, \boldsymbol{a}_t)$ generates the next state $\boldsymbol{s}_{t+1}$ and gives the agent a reward r_t. The reward is assigned based on how well the character's motion aligns with the ground-truth human motion. The goal of our character control learning process is to learn an optimal policy π^* that maximizes the expected return $J(\pi) = \mathbb{E}_\pi \left[\sum_t \gamma^t r_t \right]$ which translates to imitating the ground-truth motion as closely as possible. We apply a standard reinforcement learning algorithm (PPO [49]) to solve for the optimal policy. In the following, we provide a detailed description of the states, actions and rewards of our control learning process. We then use a dedicated Sec. 3.3 to elaborate on our policy design.

States. The character state $\boldsymbol{s}_t \triangleq (\boldsymbol{q}_t, \dot{\boldsymbol{q}}_t, \widetilde{\boldsymbol{q}}_{t+1}, \breve{\boldsymbol{x}}_{t+1}, \boldsymbol{c}_{t+1})$ consists of the character's current pose $\boldsymbol{q}_t$, joint velocities (time derivative of the pose) $\dot{\boldsymbol{q}}_t$, as well as the estimated kinematic pose $\widetilde{\boldsymbol{q}}_{t+1}$, 2D keypoints $\breve{\boldsymbol{x}}_{t+1}$ and keypoint confidence $\boldsymbol{c}_{t+1}$ of the next frame. The state includes informa-

tion of both the current frame $(\boldsymbol{q}_t, \dot{\boldsymbol{q}}_t)$ and next frame $(\widetilde{\boldsymbol{q}}_{t+1}, \breve{\boldsymbol{x}}_{t+1}, \boldsymbol{c}_{t+1})$, so that the agent learns to take the right action $\boldsymbol{a}_t$ to transition from the current pose $\boldsymbol{q}_t$ to a desired next pose $\boldsymbol{q}_{t+1}$, *i.e.,* pose close to the ground truth.

Actions. The policy $\pi(\boldsymbol{a}_t | \boldsymbol{s}_t)$ runs at 30Hz, the input video's frame rate, while our physics simulator runs at 450Hz to ensure stable simulation. This means one policy step corresponds to 15 simulation steps. One common design of the policy's action $\boldsymbol{a}_t$ is to directly output the torques $\boldsymbol{\tau}_t$ to be applied at each joint (except the root), which are used repeatedly by the simulator during the 15 simulation steps. However, finer control can be achieved by adjusting the torques at each step based on the state of the character. Thus, we follow prior work [44, 65] and use proportional-derivative (PD) controllers at each non-root joint to produce torques. With this design, the action $\boldsymbol{a}_t$ includes the target joint angles $\boldsymbol{u}_t$ of the PD controllers. At the j-th of the 15 simulation (PD controller) steps, the joint torques $\boldsymbol{\tau}_t$ are computed as

$$\boldsymbol{\tau}_t = \boldsymbol{k}_{\mathrm{p}} \circ (\boldsymbol{u}_t - \boldsymbol{q}_t^{\mathrm{nr}}) - \boldsymbol{k}_{\mathrm{d}} \circ \dot{\boldsymbol{q}}_t^{\mathrm{nr}}, \qquad (1)$$

where $\boldsymbol{k}_{\mathrm{p}}$ and $\boldsymbol{k}_{\mathrm{d}}$ are the parameters of the PD controllers, $\boldsymbol{q}_t^{\mathrm{nr}}$ and $\dot{\boldsymbol{q}}_t^{\mathrm{nr}}$ denote the joint angles and velocities of non-root joints at the start of the simulation step, and $\circ$ denotes element-wise multiplication. The PD controllers act like damped springs that drive joints to target angles $\boldsymbol{u}_t$, where $\boldsymbol{k}_{\mathrm{p}}$ and $\boldsymbol{k}_{\mathrm{d}}$ are the stiffness and damping of the springs. In Sec. 3.4, we will introduce a new control mechanism, meta-PD control, that allows $\boldsymbol{k}_{\mathrm{p}}$ and $\boldsymbol{k}_{\mathrm{d}}$ to be dynamically adjusted by the policy to achieve an even finer level of character control. With Meta-PD control, the action $\boldsymbol{a}_t$ includes elements $\boldsymbol{\lambda}_t^{\mathrm{p}}$ and $\boldsymbol{\lambda}_t^{\mathrm{d}}$ for adjusting $\boldsymbol{k}_{\mathrm{p}}$ and $\boldsymbol{k}_{\mathrm{d}}$ respectively. As observed in prior work [66], allowing the policy to apply external residual forces to the root greatly improves the ro-

4

bustness of character control. Thus, we also add the residual forces and torques $\boldsymbol{\eta}_t$ of the root into the action $\boldsymbol{a}_t$. Overall, the action is defined as $\boldsymbol{a}_t \triangleq (\boldsymbol{u}_t, \boldsymbol{\eta}_t, \boldsymbol{\lambda}_t^{\mathrm{p}}, \boldsymbol{\lambda}_t^{\mathrm{d}})$.

Rewards. In order to learn the policy, we need to define a reward function that encourages the motion $\boldsymbol{q}_{1:T}$ generated by the policy to match the ground-truth motion $\widehat{\boldsymbol{q}}_{1:T}$. Note that we use $\widehat{\cdot}$ to denote ground-truth quantities. The reward r_t at each time step is defined as the multiplication of four sub-rewards:

$$r_t = r_t^{\mathrm{p}} \cdot r_t^{\mathrm{v}} \cdot r_t^{\mathrm{j}} \cdot r_t^{\mathrm{k}} . \tag{2}$$

The pose reward r_t^{p} measures the difference between the local joint orientations $\boldsymbol{o}_t^j$ and the ground truth $\widehat{\boldsymbol{o}}_t^j$:

$$r_t^{\mathrm{p}} = \exp\left[-\alpha_{\mathrm{p}} \left(\sum_{j=1}^{J} \|\boldsymbol{o}_t^j \ominus \widehat{\boldsymbol{o}}_t^j\|^2\right)\right], \tag{3}$$

where J is the total number of joints, $\ominus$ denotes the relative rotation between two rotations, and $\|\cdot\|$ computes the rotation angle. The velocity reward r_t^{v} measures the mismatch between joint velocities $\dot{\boldsymbol{q}}_t$ and the ground truth $\widehat{\dot{\boldsymbol{q}}}_t$:

$$r_t^{\mathrm{v}} = \exp\left[-\alpha_{\mathrm{v}} \|\dot{\boldsymbol{q}}_t - \widehat{\dot{\boldsymbol{q}}}_t\|^2\right]. \tag{4}$$

The joint position reward r_t^{j} encourages the 3D world joint positions $\boldsymbol{X}_t^j$ to match the ground truth $\widehat{\boldsymbol{X}}_t^j$:

$$r_t^{\mathrm{j}} = \exp\left[-\alpha_{\mathrm{j}} \left(\sum_{j=1}^{J} \|\boldsymbol{X}_t^j - \widehat{\boldsymbol{X}}_t^j\|^2\right)\right]. \tag{5}$$

Finally, the keypoint reward r_t^{k} pushes the 2D image projection $\boldsymbol{x}_t^j$ of the joints to match the ground truth $\widehat{\boldsymbol{x}}_t^j$:

$$r_t^{\mathrm{k}} = \exp\left[-\alpha_{\mathrm{k}} \left(\sum_{j=1}^{J} \|\boldsymbol{x}_t^j - \widehat{\boldsymbol{x}}_t^j\|^2\right)\right]. \tag{6}$$

Note that the orientations $\boldsymbol{o}_t^j$, 3D joint positions $\boldsymbol{X}_t^j$ and 2D image projections $\boldsymbol{x}_t^j$ are functions of the pose $\boldsymbol{q}_t$. The joint velocities $\dot{\boldsymbol{q}}_t$ are computed via finite difference. There are also weighting factors $\alpha_{\mathrm{p}}, \alpha_{\mathrm{v}}, \alpha_{\mathrm{j}}, \alpha_{\mathrm{k}}$ inside each reward. These sub-rewards complement each other by matching different features of the generated motion to the ground-truth: joint angles, velocities, as well as 3D and 2D joint positions. Our reward design is multiplicative, which eases policy learning as noticed by prior work [62]. The multiplication of the sub-rewards ensures that none of them can be overlooked in order to achieve a high reward.

3.3. Kinematics-Aware Policy

As the action $\boldsymbol{a}_t$ is continuous, we adopt a parametrized Gaussian policy $\pi_\theta(\boldsymbol{a}_t|\boldsymbol{s}_t) = \mathcal{N}(\overline{\boldsymbol{a}}_t, \boldsymbol{\Sigma})$ where the mean $\overline{\boldsymbol{a}}_t \triangleq (\overline{\boldsymbol{u}}_t, \overline{\boldsymbol{\eta}}_t, \overline{\boldsymbol{\lambda}}_t^{\mathrm{p}}, \overline{\boldsymbol{\lambda}}_t^{\mathrm{d}})$ is output by a neural network $\mathcal{F}_\theta$ with parameters θ, and $\boldsymbol{\Sigma}$ is a fixed diagonal covariance matrix

whose elements are treated as hyperparameters. The noise inside the Gaussian policy governed by $\boldsymbol{\Sigma}$ allows the agent to explore different actions around the mean action $\overline{\boldsymbol{a}}_t$ and use these explorations to improve the policy during training. At test time, the noise is removed and the character agent always takes the mean action $\overline{\boldsymbol{a}}_t$ to improve performance.

Now let us focus on the design of the policy network $\mathcal{F}_\theta$ that maps the state $\boldsymbol{s}_t$ to the mean action $\overline{\boldsymbol{a}}_t$. Based on the design of $\boldsymbol{s}_t$, the mapping can be written as

$$\overline{\boldsymbol{a}}_t = \mathcal{F}_\theta\left(\boldsymbol{q}_t, \dot{\boldsymbol{q}}_t, \widetilde{\boldsymbol{q}}_{t+1}, \check{\boldsymbol{x}}_{t+1}, \boldsymbol{c}_{t+1}\right). \tag{7}$$

Recall that $\widetilde{\boldsymbol{q}}_{t+1}$ is the kinematic pose, $\check{\boldsymbol{x}}_{t+1}$ and $\boldsymbol{c}_{t+1}$ are the detected 2D keypoints and their confidence, and that they are all information about the next frame. The overall architecture of our policy network $\mathcal{F}_\theta$ is illustrated in Fig. 2 (c). The components $(\overline{\boldsymbol{u}}_t, \overline{\boldsymbol{\eta}}_t, \overline{\boldsymbol{\lambda}}_t^{\mathrm{p}}, \overline{\boldsymbol{\lambda}}_t^{\mathrm{d}})$ of the mean action $\overline{\boldsymbol{a}}_t$ are computed as follows:

$$\widetilde{\boldsymbol{q}}_{t+1}^{(n)} = \mathcal{R}_\theta\left(\widetilde{\boldsymbol{q}}_{t+1}, \check{\boldsymbol{x}}_{t+1}, \boldsymbol{c}_{t+1}\right), \tag{8}$$

$$(\delta\overline{\boldsymbol{u}}_t, \overline{\boldsymbol{\eta}}_t, \overline{\boldsymbol{\lambda}}_t^{\mathrm{p}}, \overline{\boldsymbol{\lambda}}_t^{\mathrm{d}}) = \mathcal{G}_\theta\left(\widetilde{\boldsymbol{q}}_{t+1}^{(n)}, \boldsymbol{q}_t, \dot{\boldsymbol{q}}_t\right), \tag{9}$$

$$\overline{\boldsymbol{u}}_t = \widetilde{\boldsymbol{q}}_{t+1}^{(n)} + \delta\overline{\boldsymbol{u}}_t . \tag{10}$$

In Eq. (8), $\mathcal{R}_\theta$ is a kinematic refinement unit that iteratively refines the kinematic pose $\widetilde{\boldsymbol{q}}_{t+1}$ using the 2D keypoints $\check{\boldsymbol{x}}_{t+1}$ and confidence $\boldsymbol{c}_{t+1}$, and $\widetilde{\boldsymbol{q}}_{t+1}^{(n)}$ is the refined pose after n iterations of refinement. Eq. (9) and (10) describe a control generation unit $\mathcal{G}_\theta$ that maps the refined pose $\widetilde{\boldsymbol{q}}_{t+1}^{(n)}$, current pose $\boldsymbol{q}_t$ and velocities $\dot{\boldsymbol{q}}_t$ to the components of the mean action $\overline{\boldsymbol{a}}_t$. Specifically, the control generation unit $\mathcal{G}_\theta$ includes a hand-crafted feature extraction layer, a normalization layer (based on running estimates of mean and variance) and another MLP $\mathcal{V}_\theta$, as illustrated in Fig. 2 (c). As described in Eq. (10), an important design of $\mathcal{G}_\theta$ is a residual connection that produces the mean PD controller target angles $\overline{\boldsymbol{u}}_t$ using the refined kinematic pose $\widetilde{\boldsymbol{q}}_{t+1}^{(n)}$, where we ignore the root angles and positions in $\widetilde{\boldsymbol{q}}_{t+1}^{(n)}$ for ease of notation. This design builds in proper inductive bias since $\widetilde{\boldsymbol{q}}_{t+1}^{(n)}$ provides a good guess for the desired next pose $\boldsymbol{q}_{t+1}$ and thus a good base value for $\overline{\boldsymbol{u}}_t$. It is important to note that the PD controller target angles $\boldsymbol{u}_t$ do not translate to the same next pose $\boldsymbol{q}_{t+1}$ of the character, *i.e.,* $\boldsymbol{q}_{t+1} \neq \boldsymbol{u}_t$. The reason is that the character is subject to gravity and contact forces, and under these external forces the joint angles $\boldsymbol{q}_{t+1}$ will not be $\boldsymbol{u}_t$ when the PD controllers reach their equilibrium. As an analogy, since PD controllers act like springs, a spring will reach a different equilibrium position when you apply external forces to it. Despite this, the next pose $\boldsymbol{q}_{t+1}$ generally will not be far away from $\boldsymbol{u}_t$ and learning the residual $\delta\overline{\boldsymbol{u}}_t$ to $\widetilde{\boldsymbol{q}}_{t+1}^{(n)}$ is easier than learning from scratch as we will demonstrate in the experiments. This design also synergizes the kinematics of the character with its dynamics as the kinematic pose $\widetilde{\boldsymbol{q}}_{t+1}^{(n)}$ is now tightly coupled with

the input of the character's PD controllers that control the character in the physics simulator.

Kinematic Refinement Unit. The kinematic refinement unit $\mathcal{R}_\theta$ is formed by an MLP $\mathcal{U}_\theta$ that maps a feature vector $\boldsymbol{z}$ (specific form will be described later) to a pose update:

$$\delta\widetilde{\boldsymbol{q}}_{t+1}^{(i)} = \mathcal{U}_\theta\left(\boldsymbol{z}\right), \tag{11}$$

$$\widetilde{\boldsymbol{q}}_{t+1}^{(i+1)} = \widetilde{\boldsymbol{q}}_{t+1}^{(i)} + \delta\widetilde{\boldsymbol{q}}_{t+1}^{(i)}, \tag{12}$$

where i denotes the i-th refinement iteration and $\widetilde{\boldsymbol{q}}_{t+1}^{(0)} = \widetilde{\boldsymbol{q}}_{t+1}$. To fully leverage the 2D keypoints and kinematic pose at hand, we design the feature $\boldsymbol{z}$ to be the gradient of the keypoint reprojection loss with respect to current 3D joint positions, inspired by recent work [51] on kinematic body fitting. The purpose of using the gradient is not to minimize the reprojection loss, but to use it as an informative kinematic feature to learn a pose update that eventually results in stable and accurate control of the character; there is no explicit minimization of the reprojection loss in our formulation. Specifically, we first obtain the 3D joint positions $\widetilde{\boldsymbol{X}}_{t+1} = \text{FK}(\widetilde{\boldsymbol{q}}_{t+1}^{(i)})$ through forward kinematics and then compute the reprojection loss as

$$\mathcal{L}(\widetilde{\boldsymbol{X}}_{t+1}) = \sum_{j=1}^{J} \left\| \Pi\left(\widetilde{\boldsymbol{X}}_{t+1}^j\right) - \breve{\boldsymbol{x}}_{t+1}^j \right\|^2 \cdot c_{t+1}^j, \tag{13}$$

where $\widetilde{\boldsymbol{X}}_{t+1}^j$ denotes the j-th joint position in $\widetilde{\boldsymbol{X}}_{t+1}$, $\Pi(\cdot)$ denotes the perspective camera projection, and $(\breve{\boldsymbol{x}}_{t+1}^j, c_{t+1}^j)$ are the j-th detected keypoint and its confidence. The gradient feature $\boldsymbol{z} \triangleq \partial\mathcal{L}/\partial\widetilde{\boldsymbol{X}}_{t+1}$ is informative about the kinematic pose $\widetilde{\boldsymbol{q}}_{t+1}^{(i)}$ as it tells us how each joint should move to match the 2D keypoints $\breve{\boldsymbol{x}}_{t+1}^j$. It also accounts for keypoint uncertainty by weighting the loss with the keypoint confidence c_{t+1}^j. Note that $\boldsymbol{z}$ is converted to the character's root coordinate to be invariant of the character's orientation. The refinement unit integrates kinematics and dynamics as it utilizes a kinematics-based feature $\boldsymbol{z}$ to learn the update of a kinematic pose, which is used to produce dynamics-based control of the character. The joint learning of the kinematic refinement unit $\mathcal{R}_\theta$ and the control generation unit $\mathcal{G}_\theta$ ensures accurate and physically-plausible pose estimation.

Feature Extraction Layer. After refinement, the control generation unit $\mathcal{G}_\theta$ needs to extract informative features from its input to output an action that advances the character from the current pose $\boldsymbol{q}_t$ to the next pose $\boldsymbol{q}_{t+1}$. To this end, the feature extraction layer uses information from both the current frame and next frame. Specifically, the extracted feature includes $\boldsymbol{q}_t$, $\dot{\boldsymbol{q}}_t$, the current 3D joint positions $\boldsymbol{X}_t$, the pose difference vector between $\boldsymbol{q}_t$ and the refined kinematic pose $\widetilde{\boldsymbol{q}}_{t+1}^{(n)}$, and the difference vector between $\boldsymbol{X}_t$ and the next-frame joint position $\widetilde{\boldsymbol{X}}_{t+1}$ computed from $\widetilde{\boldsymbol{q}}_{t+1}^{(n)}$. All features are converted to the character's root coordinate

to be orientation-invariant and encourage robustness against variations in absolute pose encountered at test time.

3.4. Meta-PD control

PD controllers are essential in our approach as they relate the kinematics and dynamics of the character by converting target joint angles in pose space to joint torques. However, an undesirable aspect of PD controllers is the need to specify the parameters $\boldsymbol{k}_{\text{p}}$ and $\boldsymbol{k}_{\text{d}}$ for computing the joint torques $\boldsymbol{\tau}_t$ as described in Eq. (1). It is undesirable because (i) manual parameter tuning requires significant domain knowledge and (ii) even carefully designed parameters can be suboptimal. The difficulty, here, lies in balancing the ratio between $\boldsymbol{k}_{\text{p}}$ and $\boldsymbol{k}_{\text{d}}$. Large ratios can lead to unstable and jittery motion while small values can result in motion that is too smooth and lags behind ground truth.

Motivated by this problem, we propose meta-PD control, a method that allows the policy to dynamically adjust $\boldsymbol{k}_{\text{p}}$ and $\boldsymbol{k}_{\text{d}}$ based on the state of the character. Specifically, given some initial values $\boldsymbol{k}_{\text{p}}'$ and $\boldsymbol{k}_{\text{d}}'$, the policy outputs λ_{p} and λ_{d} as additional elements of the action $\boldsymbol{a}_t$ that act to scale $\boldsymbol{k}_{\text{p}}'$ and $\boldsymbol{k}_{\text{d}}'$. Moreover, we take this idea one step further and let the policy output two sequences of scales $\boldsymbol{\lambda}_t^{\text{p}} = (\lambda_{t1}^{\text{p}}, \ldots, \lambda_{tm}^{\text{p}})$ and $\boldsymbol{\lambda}_t^{\text{d}} = (\lambda_{t1}^{\text{d}}, \ldots, \lambda_{tm}^{\text{d}})$ where $m = 15$ corresponds to the number of PD controller (simulation) steps during a policy step. The PD controller parameters $\boldsymbol{k}_{\text{p}}$ and $\boldsymbol{k}_{\text{d}}$ at the j-th step of the 15 PD controller steps are then computed as follows:

$$\boldsymbol{k}_{\text{p}} = \lambda_{tj}^{\text{p}} \boldsymbol{k}_{\text{p}}', \quad \boldsymbol{k}_{\text{d}} = \lambda_{tj}^{\text{d}} \boldsymbol{k}_{\text{d}}'. \tag{14}$$

Instead of using fixed $\boldsymbol{k}_{\text{p}}$ and $\boldsymbol{k}_{\text{d}}$, meta-PD control allows the policy to plan the scaling of $\boldsymbol{k}_{\text{p}}$ and $\boldsymbol{k}_{\text{d}}$ through the 15 PD controller steps to have more granular control over the torques produced by the PD controllers, which in turn enables a finer level of character control. With meta-PD control, the action $\boldsymbol{a}_t$ is now defined as $\boldsymbol{a}_t \triangleq (\boldsymbol{u}_t, \boldsymbol{\eta}_t, \boldsymbol{\lambda}_t^{\text{p}}, \boldsymbol{\lambda}_t^{\text{d}})$.

4. Experiments

Datasets. We perform experiments on two large-scale human motion datasets. The first dataset is Human3.6M [15], which includes 7 annotated subjects captured at 50Hz and a total of 1.5 million training images. Following prior work [22, 21, 35], we train our model on 5 subjects (S1, S5, S6, S7, S8) and test on the other 2 subjects (S9, S11). We subsample the dataset to 25Hz for both training and testing. The second dataset we use is an in-house human motion dataset that also contains *detailed finger motion*. It consists of 3 subjects captured at 30Hz performing various actions from free body motions to natural conversations. There are around 335k training frames and 87k test frames. Our in-house dataset has complex skeletons with twice more joints than the SMPL model, including fingers. The body shape variation among subjects is also greater than that of SMPL, which further evaluates the robustness of our approach.
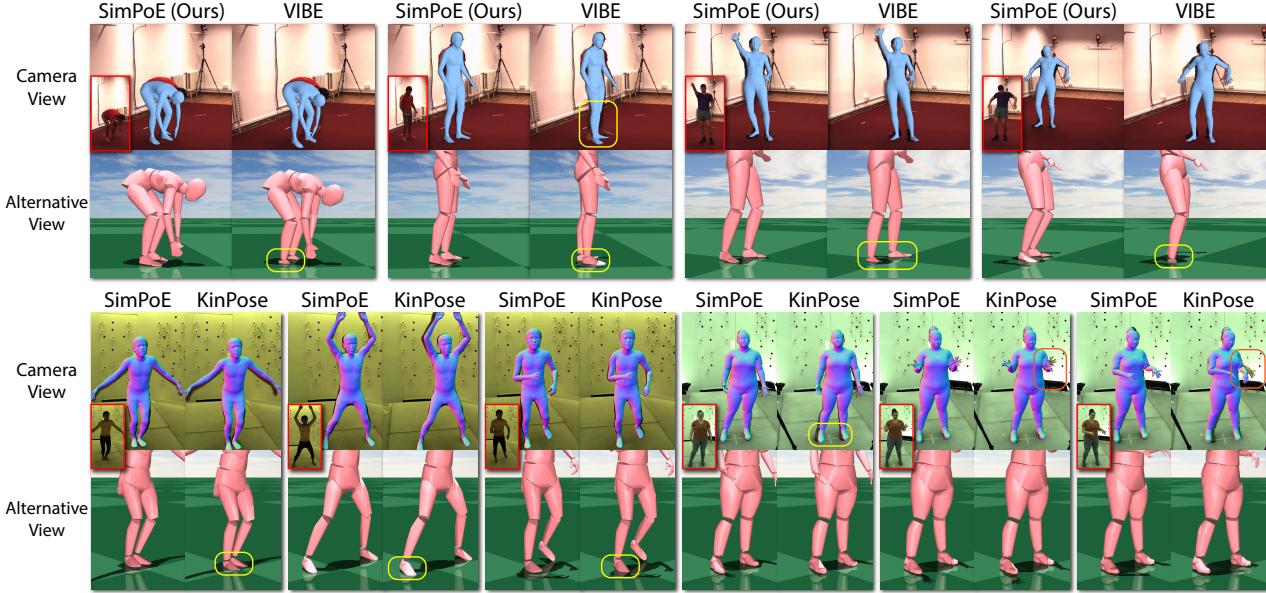
Figure 3. **Visualization** of estimated poses in the camera view and an alternative view. SimPoE estimates more accurate poses and foot contact. Pose mismatch and ground penetration are highlighted with boxes. Please see the supplementary video for more comparisons.

Metrics. We use both pose-based and *physics-based* metrics for evaluation. To assess pose accuracy, we report mean per joint position error (MPJPE) and Procrustes-aligned mean per joint position error (PA-MPJPE). We also use three physics-based metrics that measure jitter, foot sliding, and ground penetration, respectively. For jitter, we compute the difference in acceleration (Accel) between the predicted 3D joint and the ground-truth. For foot sliding (FS), we find body mesh vertices that contact the ground in two adjacent frames and compute their average displacement within the frames. For ground penetration (GP), we compute the average distance to the ground for mesh vertices below the ground. The units for these metrics are millimeters (mm) except for Accel (mm/frame^2). MPJPE, PA-MPJPE and Accel are computed in the root-centered coordinate.

4.1. Implementation Details.

Character Models. We use MuJoCo [54] as the physics simulator. For the character creation process in Sec. 3.1, we use VIBE [21] to recover an SMPL model for each subject in Human3.6M. Each MuJoCo character created from the SMPL model has 25 bones and 76 degrees of freedom (DoFs). For our in-house motion dataset, we use non-rigid ICP [1] and linear blend skinning [19] to reconstruct a skinned human mesh model for each subject. Each of these models has fingers and includes 63 bones and 114 DoFs.

Initialization. For Human3.6M, we use VIBE to provide the initial kinematic motion $\widetilde{q}_{1:T}$. For our in-house motion dataset, since our skinned human models have more complex skeletons and meshes than the SMPL model, we develop our own kinematic pose estimator, which is detailed

Human3.6M						
Method	Physics	MPJPE ↓	PA-MPJPE ↓	Accel ↓	FS ↓	GP ↓
VIBE [21]	✗	61.3	43.1	15.2	15.1	12.6
NeurGD* [51]	✗	57.3	42.2	14.2	16.7	24.4
PhysCap [50]	✓	113.0	68.9	-	-	-
EgoPose [65]	✓	130.3	79.2	31.3	5.9	3.5
SimPoE (Ours)	✓	**56.7**	**41.6**	**6.7**	**3.4**	**1.6**
In-House Motion Dataset						
Method	Physics	MPJPE ↓	PA-MPJPE ↓	Accel ↓	FS ↓	GP ↓
KinPose	✗	49.7	40.4	12.8	6.4	3.9
NeurGD* [51]	✗	36.7	30.9	16.2	7.7	3.6
EgoPose [65]	✓	202.2	131.4	32.6	2.2	0.5
SimPoE (Ours)	✓	**26.6**	**21.2**	**8.4**	**0.5**	**0.1**

Table 1. Results of pose-based (MPJPE, PA-MPJPE) and physics-based (Accel, FS, GP) metrics on Human3.6M and our in-house motion dataset. Symbol "-" means results are not available and "*" means self-implementation (better results than the original paper).

in Appendix A. To recover the global root position of the person, we assume the camera intrinsic parameters are calibrated and optimize the root position by minimizing the reprojection loss of 2D keypoints, similar to the kinematic initialization in [50].

Other Details. The kinematic refinement unit in the policy network refines the kinematic pose $n = 5$ times. To facilitate learning, we first pretrain the refinement unit with supervised learning using an MSE loss on the refined kinematic pose. The normalization layer in the policy computes the running average of the mean and variance of the input feature during training, and uses it to produce a normalized feature. Our learned policy runs at 38 FPS on a standard PC with an Intel Core i9 Processor. More implementation

Method	Human3.6M					In-House Motion Dataset				
	MPJPE ↓	PA-MPJPE ↓	Accel ↓	FS ↓	GP ↓	MPJPE ↓	PA-MPJPE ↓	Accel ↓	FS ↓	GP ↓
w/o Meta-PD	59.9	44.7	**5.9**	**2.2**	**1.4**	39.8	31.7	7.1	**0.4**	**0.1**
w/o Refine	61.2	43.5	8.0	3.4	2.0	47.9	38.9	9.6	0.6	**0.1**
w/o ResAngle	68.7	51.0	6.4	4.1	2.1	193.4	147.6	**6.5**	0.9	0.3
w/o ResForce	115.2	65.1	23.5	6.1	3.2	48.4	31.3	12.5	0.9	0.3
w/o FeatLayer	81.4	47.6	9.3	5.0	1.8	36.9	27.5	9.5	0.6	**0.1**
SimPoE (Ours)	**56.7**	**41.6**	6.7	3.4	1.6	**26.6**	**21.2**	8.4	0.5	**0.1**

Table 2. Ablation studies on Human3.6M and our in-house motion dataset.

Figure 4. Effect of refinement unit.

details such as training procedures and hyperparameter settings can be found in Appedix B.

4.2. Comparison to state-of-the-art methods

We compare SimPoE against state-of-the-art monocular 3D human pose estimation methods, including both kinematics-based (VIBE [21], NeurGD [51]) and physics-based (PhysCap [50], EgoPose [65]) approaches. The results of VIBE and EgoPose are obtained using their publicly released code and models. As PhysCap and NeurGD have not released their code, we directly use the reported results on Human3.6M from the PhysCap paper and implement our own version of NeurGD. Table 1 summarizes the quantitative results on Human3.6M and the in-house motion dataset. On Human3.6M, we can observe that our method, SimPoE, outperforms previous methods in pose accuracy as indicated by the smaller MPJPE and PA-MPJPE. In particular, SimPoE shows large pose accuracy improvements over state-of-the-art physics-based approaches (EgoPose [65] and PhysCap [50]), reducing the MPJPE almost by half. For physics-based metrics (Accel, FS and GP), SimPoE also outperforms prior methods by large margins. It means that SimPoE significantly reduces the physical artifacts – jitter (Accel), foot sliding (FS), and ground penetration (GP), which particularly deteriorate the results of kinematic methods (VIBE [21] and NeurGD [51]). On the in-house motion dataset, SimPoE again outperforms previous methods in terms of both pose-based and physics-based metrics. In the table, KinPose denotes our own kinematic pose estimator used by SimPoE. We note that the large acceleration error (Accel) of EgoPose is due to the frequent falling of the character, which is a common problem in physics-based methods since the character can lose balance when performing agile motions. The learned policy of SimPoE is robust enough to stably control the character without falling, which prevents irregular accelerations.

We also provide qualitative comparisons in Fig. 3, where we show the estimated poses in the camera view and the same poses rendered from an alternative view. The alternative view shows that SimPoE can estimate foot contact with the ground more accurately and without penetration. As the quality and physical plausibility of the estimated motions are best seen in videos, please refer to the supplementary video for additional qualitative results and comparisons.
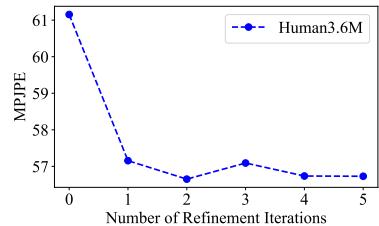
4.3. Ablation Studies

To further validate our proposed approach, we conduct extensive ablation studies to investigate the contribution of each proposed component to the performance. Table 2 summarizes the results where we train different variants of Sim-PoE by removing a single component each time. First, we can observe that both meta-PD control and the kinematic refinement unit contribute to better pose accuracy as indicated by the corresponding ablations (w/o Meta-PD and w/o Refine). Second, the ablation (w/o ResAngle) shows that it is important to have the residual connection in the policy network for producing the mean PD controller target angles $\overline{u}_t$. Next, the residual forces η_t we use in action a_t are also indispensable as demonstrated by the drop in performance of the variant (w/o ResForce). Without the residual forces, the policy is not robust and the character often falls down as indicated by the large acceleration error (Accel). Finally, it is evident from the ablation (w/o FeatLayer) that our feature extraction layer in the policy is also instrumental, because it extracts informative features of both the current frame and next frame to learn control that advances the character to the next pose. We also perform ablations to investigate how the number of refinement iterations in the policy affects pose accuracy. As shown in Fig. 4, the performance gain saturates around 5 refinement iterations.

5. Discussion and Future Work

In this work, we demonstrate that modeling both kinematics and dynamics improves the accuracy and physical plausibility of 3D human pose estimation from monocular video. Our approach, SimPoE, unifies kinematics and dynamics by integrating image-based kinematic inference and physics-based character control into a joint reinforcement learning framework. It runs in real-time, is compatible with advanced physics simulators, and addresses several drawbacks of prior physics-based approaches.

However, due to its physics-based formulation, SimPoE depends on 3D scene modeling to enforce contact constraints during motion estimation. This hinders direct evaluation on in-the-wild datasets, such as 3DPW [57], which includes motions such as climbing stairs or even trees. Future work may include integration of video-based 3D scene reconstruction to address this limitation.

References

[1] Brian Amberg, Sami Romdhani, and Thomas Vetter. Optimal step nonrigid icp algorithms for surface registration. In *2007 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8. IEEE, 2007. 7, 12

[2] Dragomir Anguelov, Praveen Srinivasan, Daphne Koller, Sebastian Thrun, Jim Rodgers, and James Davis. Scape: shape completion and animation of people. In *ACM SIGGRAPH 2005 Papers*, pages 408–416. 2005. 3

[3] Anurag Arnab, Carl Doersch, and Andrew Zisserman. Exploiting temporal context for 3d human pose estimation in the wild. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3395–3404, 2019. 3

[4] Kevin Bergamin, Simon Clavet, Daniel Holden, and James Richard Forbes. Drecon: data-driven responsive control of physics-based characters. *ACM Transactions on Graphics (TOG)*, 38(6):1–11, 2019. 3

[5] Federica Bogo, Angjoo Kanazawa, Christoph Lassner, Peter Gehler, Javier Romero, and Michael J Black. Keep it smpl: Automatic estimation of 3d human pose and shape from a single image. In *European Conference on Computer Vision*, pages 561–578. Springer, 2016. 3

[6] Marcus A Brubaker, Leonid Sigal, and David J Fleet. Estimating contact dynamics. In *2009 IEEE 12th International Conference on Computer Vision*, pages 2389–2396. IEEE, 2009. 3

[7] Zhe Cao, Tomas Simon, Shih-En Wei, and Yaser Sheikh. Realtime multi-person 2d pose estimation using part affinity fields. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7291–7299, 2017. 3

[8] Erwin Coumans. Bullet physics engine. *Open Source Software: http://bulletphysics. org*, 1(3):84, 2010. 2

[9] Rishabh Dabral, Anurag Mundhada, Uday Kusupati, Safeer Afaque, Abhishek Sharma, and Arjun Jain. Learning 3d human pose from structure and motion. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 668–683, 2018. 3

[10] Riza Alp Guler and Iasonas Kokkinos. Holopose: Holistic 3d human reconstruction in-the-wild. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 10884–10894, 2019. 3

[11] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *arXiv preprint arXiv:1512.03385*, 2015. 12

[12] Jonathan Ho and Stefano Ermon. Generative adversarial imitation learning. In *Advances in Neural Information Processing Systems*, pages 4565–4573, 2016. 3

[13] Ludovic Hoyet, Rachel McDonnell, and Carol O'Sullivan. Push it real: Perceiving causality in virtual interactions. *ACM Transactions on Graphics (TOG)*, 31(4):1–9, 2012. 2

[14] Yinghao Huang, Federica Bogo, Christoph Lassner, Angjoo Kanazawa, Peter V Gehler, Javier Romero, Ijaz Akhter, and Michael J Black. Towards accurate marker-less human shape and pose estimation over time. In *2017 international conference on 3D vision (3DV)*, pages 421–430. IEEE, 2017. 3

[15] 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 transactions on pattern analysis and machine intelligence*, 36(7):1325–1339, 2013. 2, 6, 12

[16] Mariko Isogawa, Ye Yuan, Matthew O'Toole, and Kris M Kitani. Optical non-line-of-sight physics-based 3d human pose estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7013–7022, 2020. 3

[17] Angjoo Kanazawa, Michael J Black, David W Jacobs, and Jitendra Malik. End-to-end recovery of human shape and pose. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7122–7131, 2018. 3

[18] Angjoo Kanazawa, Jason Y Zhang, Panna Felsen, and Jitendra Malik. Learning 3d human dynamics from video. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5614–5623, 2019. 3

[19] Ladislav Kavan, Steven Collins, Jiří Žára, and Carol O'Sullivan. Skinning with dual quaternions. In *Proceedings of the 2007 symposium on Interactive 3D graphics and games*, pages 39–46, 2007. 7, 12

[20] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 13

[21] Muhammed Kocabas, Nikos Athanasiou, and Michael J Black. Vibe: Video inference for human body pose and shape estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5253–5263, 2020. 1, 3, 6, 7, 8, 12

[22] Nikos Kolotouros, Georgios Pavlakos, Michael J Black, and Kostas Daniilidis. Learning to reconstruct 3d human pose and shape via model-fitting in the loop. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2252–2261, 2019. 1, 3, 6

[23] Christoph Lassner, Javier Romero, Martin Kiefel, Federica Bogo, Michael J Black, and Peter V Gehler. Unite the people: Closing the loop between 3d and 2d human representations. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 6050–6059, 2017. 3

[24] Zongmian Li, Jiri Sedlar, Justin Carpentier, Ivan Laptev, Nicolas Mansard, and Josef Sivic. Estimating 3d motion and forces of person-object interactions from monocular video. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8640–8649, 2019. 2

[25] Libin Liu and Jessica Hodgins. Learning to schedule control fragments for physics-based characters using deep q-learning. *ACM Transactions on Graphics (TOG)*, 36(3):29, 2017. 3

[26] Libin Liu and Jessica Hodgins. Learning basketball dribbling skills using trajectory optimization and deep reinforcement learning. *ACM Transactions on Graphics (TOG)*, 37(4):1–14, 2018. 3

[27] Matthew Loper, Naureen Mahmood, Javier Romero, Gerard Pons-Moll, and Michael J Black. Smpl: A skinned multi-person linear model. *ACM transactions on graphics (TOG)*, 34(6):1–16, 2015. 3, 12

[28] Zhengyi Luo, S Alireza Golestaneh, and Kris M Kitani. 3d human motion estimation via motion compression and refinement. In *ACCV*, 2020. 3

[29] Dushyant Mehta, Oleksandr Sotnychenko, Franziska Mueller, Weipeng Xu, Srinath Sridhar, Gerard Pons-Moll, and Christian Theobalt. Single-shot multi-person 3d pose estimation from monocular rgb. In *2018 International Conference on 3D Vision (3DV)*, pages 120–130. IEEE, 2018. 3

[30] Dushyant Mehta, Srinath Sridhar, Oleksandr Sotnychenko, Helge Rhodin, Mohammad Shafiei, Hans-Peter Seidel, Weipeng Xu, Dan Casas, and Christian Theobalt. Vnect: Real-time 3d human pose estimation with a single rgb camera. *ACM Transactions on Graphics (TOG)*, 36(4):1–14, 2017. 3

[31] Josh Merel, Arun Ahuja, Vu Pham, Saran Tunyasuvunakool, Siqi Liu, Dhruva Tirumala, Nicolas Heess, and Greg Wayne. Hierarchical visuomotor control of humanoids. *arXiv preprint arXiv:1811.09656*, 2018. 3

[32] Josh Merel, Leonard Hasenclever, Alexandre Galashov, Arun Ahuja, Vu Pham, Greg Wayne, Yee Whye Teh, and Nicolas Heess. Neural probabilistic motor primitives for humanoid control. *arXiv preprint arXiv:1811.11711*, 2018. 3

[33] Josh Merel, Yuval Tassa, Sriram Srinivasan, Jay Lemmon, Ziyu Wang, Greg Wayne, and Nicolas Heess. Learning human behaviors from motion capture by adversarial imitation. *arXiv preprint arXiv:1707.02201*, 2017. 3

[34] Josh Merel, Saran Tunyasuvunakool, Arun Ahuja, Yuval Tassa, Leonard Hasenclever, Vu Pham, Tom Erez, Greg Wayne, and Nicolas Heess. Catch & carry: reusable neural controllers for vision-guided whole-body tasks. *ACM Transactions on Graphics (TOG)*, 39(4):39–1, 2020. 3

[35] Gyeongsik Moon and Kyoung Mu Lee. I2l-meshnet: Image-to-lixel prediction network for accurate 3d human pose and mesh estimation from a single rgb image. In *ECCV*, 2020. 1, 6

[36] Mohamed Omran, Christoph Lassner, Gerard Pons-Moll, Peter Gehler, and Bernt Schiele. Neural body fitting: Unifying deep learning and model based human pose and shape estimation. In *2018 international conference on 3D vision (3DV)*, pages 484–494. IEEE, 2018. 3

[37] Soohwan Park, Hoseok Ryu, Seyoung Lee, Sunmin Lee, and Jehee Lee. Learning predict-and-simulate policies from unorganized human motion data. *ACM Transactions on Graphics (TOG)*, 38(6):1–11, 2019. 3

[38] Georgios Pavlakos, Vasileios Choutas, Nima Ghorbani, Timo Bolkart, Ahmed AA Osman, Dimitrios Tzionas, and Michael J Black. Expressive body capture: 3d hands, face, and body from a single image. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 10975–10985, 2019. 1, 3

[39] Georgios Pavlakos, Luyang Zhu, Xiaowei Zhou, and Kostas Daniilidis. Learning to estimate 3d human pose and shape from a single color image. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 459–468, 2018. 3

[40] Dario Pavllo, Christoph Feichtenhofer, David Grangier, and Michael Auli. 3d human pose estimation in video with temporal convolutions and semi-supervised training. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7753–7762, 2019. 3

[41] Xue Bin Peng, Pieter Abbeel, Sergey Levine, and Michiel van de Panne. Deepmimic: Example-guided deep reinforcement learning of physics-based character skills. *ACM Transactions on Graphics (TOG)*, 37(4):1–14, 2018. 2, 3

[42] Xue Bin Peng, Michael Chang, Grace Zhang, Pieter Abbeel, and Sergey Levine. Mcp: Learning composable hierarchical control with multiplicative compositional policies. In *Advances in Neural Information Processing Systems*, pages 3681–3692, 2019. 3

[43] Xue Bin Peng, Angjoo Kanazawa, Jitendra Malik, Pieter Abbeel, and Sergey Levine. Sfv: Reinforcement learning of physical skills from videos. *ACM Transactions on Graphics (TOG)*, 37(6):1–14, 2018. 3

[44] Xue Bin Peng and Michiel van de Panne. Learning locomotion skills using deeprl: Does the choice of action space matter? In *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pages 1–13, 2017. 2, 4

[45] Mir Rayat Imtiaz Hossain and James J Little. Exploiting temporal information for 3d human pose estimation. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 68–84, 2018. 3

[46] Paul SA Reitsma and Nancy S Pollard. Perceptual metrics for character animation: sensitivity to errors in ballistic motion. In *ACM SIGGRAPH 2003 Papers*, pages 537–542. 2003. 2

[47] Davis Rempe, Leonidas J. Guibas, Aaron Hertzmann, Bryan Russell, Ruben Villegas, and Jimei Yang. Contact and human dynamics from monocular video. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2020. 2, 3

[48] John Schulman, Philipp Moritz, Sergey Levine, Michael Jordan, and Pieter Abbeel. High-dimensional continuous control using generalized advantage estimation. *arXiv preprint arXiv:1506.02438*, 2015. 13

[49] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017. 4, 13

[50] Soshi Shimada, Vladislav Golyanik, Weipeng Xu, and Christian Theobalt. Physcap: Physically plausible monocular 3d motion capture in real time. *ACM Transactions on Graphics*, 39(6), dec 2020. 2, 3, 7, 8

[51] Jie Song, Xu Chen, and Otmar Hilliges. Human body model fitting by learned gradient descent. In *ECCV*, 2020. 3, 6, 7, 8

[52] Yu Sun, Yun Ye, Wu Liu, Wenpeng Gao, YiLi Fu, and Tao Mei. Human mesh recovery from monocular images via a skeleton-disentangled representation. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 5349–5358, 2019. 3, 12

[53] Jun Kai Vince Tan, Ignas Budvytis, and Roberto Cipolla. Indirect deep structured learning for 3d human body shape and pose prediction. 2017. 3

[54] Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ*

International Conference on Intelligent Robots and Systems, pages 5026–5033. IEEE, 2012. 2, 7

[55] Hsiao-Yu Tung, Hsiao-Wei Tung, Ersin Yumer, and Katerina Fragkiadaki. Self-supervised learning of motion capture. In *Advances in Neural Information Processing Systems*, pages 5236–5246, 2017. 3

[56] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017. 3

[57] Timo von Marcard, Roberto Henschel, Michael J Black, Bodo Rosenhahn, and Gerard Pons-Moll. Recovering accurate 3d human pose in the wild using imus and a moving camera. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 601–617, 2018. 8

[58] Marek Vondrak, Leonid Sigal, Jessica Hodgins, and Odest Jenkins. Video-based 3d motion capture through biped control. *ACM Transactions On Graphics (TOG)*, 31(4):1–12, 2012. 3

[59] Ziyu Wang, Josh S Merel, Scott E Reed, Nando de Freitas, Gregory Wayne, and Nicolas Heess. Robust imitation of diverse behaviors. In *Advances in Neural Information Processing Systems*, pages 5320–5329, 2017. 3

[60] Shih-En Wei, Varun Ramakrishna, Takeo Kanade, and Yaser Sheikh. Convolutional pose machines. In *CVPR*, 2016. 12

[61] Xiaolin Wei and Jinxiang Chai. Videomocap: Modeling physically realistic human motion from monocular video sequences. In *ACM SIGGRAPH 2010 papers*, pages 1–10. 2010. 3

[62] Jungdam Won, Deepak Gopinath, and Jessica Hodgins. A scalable approach to control diverse behaviors for physically simulated characters. *ACM Transactions on Graphics (TOG)*, 39(4):33–1, 2020. 3, 5

[63] Donglai Xiang, Hanbyul Joo, and Yaser Sheikh. Monocular total capture: Posing face, body, and hands in the wild. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 10965–10974, 2019. 1

[64] Ye Yuan and Kris Kitani. 3d ego-pose estimation via imitation learning. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 735–750, 2018. 3

[65] Ye Yuan and Kris Kitani. Ego-pose estimation and forecasting as real-time pd control. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 10082–10092, 2019. 2, 3, 4, 7, 8

[66] Ye Yuan and Kris Kitani. Residual force control for agile human behavior imitation and extended motion synthesis. In *Advances in Neural Information Processing Systems*, 2020. 4

[67] Petrissa Zell, Bastian Wandt, and Bodo Rosenhahn. Joint 3d human motion capture and physical analysis from monocular videos. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 17–26, 2017. 3

A. In-House Motion Dataset and Kinematic Pose Estimator

In-House Dataset. Our in-house motion dataset uses a more complex skeleton model (with twice as many joints, including fingers) than SMPL [27]. To recover the human skinning mesh model of each subject, we use an offline process that uses multiview 3D reconstruction to produce a person-specific skinning template with 32 bone scaling parameters based on non-rigid ICP deformation [1] and linear blend skinning [19]. For motion, we solve for 94 local joint angles (degrees of freedom) and the global 3D position of 159 joints, including finger joints, for every frame.

Kinematic Pose Estimator. Since existing kinematic pose estimators, such as VIBE [21], cannot be directly applied to the in-house dataset due to the dataset's more complex skeletons and skinning models, we design a simple kinematic tracker ("KinPose" in the main paper) that also uses monocular inputs to produce kinematic pose estimates. The model does not have any temporal component, outputting both 2D keypoint heatmaps and joint angles frame-by-frame. These two outputs are required by our approach (SimPoE) and NeurGD [52]. Below, we detail the network architecture and training procedure of this model, which are typical for such models as the performance of SimPoE is not sensitive to these design choices.

Network Architecture. We use a 3-stage cascaded network [60] with a backbone based on ResNet-50 [11]. The output of the network at each stage is a tensor of $m + n$ channels with a spatial size that is 8x smaller than the input image, where $m = 77$ is the number of heatmap channels for 2D keypoints (a subset of the 159 joints), and $n = 94 + 6$ is the number of local joint angles and global pose dimensions. Each of the 94 angle channels is an "angle map" corresponding to a joint. The final output angle (scalar) is calculated by summing the element-wise product between an angle map and its corresponding keypoint heatmaps, where the correspondence is defined based on the skeleton. This design is a type of attention mechanism, which encourages the model to predict the angle of a joint based only on relevant image regions.

Training. At each stage of the network, we apply L_2 losses on heatmaps, 2D keypoints, 3D joint positions, and joint angles to train the model, similar to VIBE [21].

B. Additional Implementation Details

Parameter	Value
Num. of time steps	50000
Num. of epochs	2000
Num. of policy updates per epoch	10
Policy step size	5×10^{-5}
Value step size	3×10^{-4}
PPO clip ϵ	0.2
Discount factor γ	0.95
GAE coefficient λ	0.95
Reward weights $(\alpha_{\mathrm{p}}, \alpha_{\mathrm{v}}, \alpha_{\mathrm{j}}, \alpha_{\mathrm{k}})$ (Human3.6M)	(30, 0.2, 100, 0.02)
Reward weights $(\alpha_{\mathrm{p}}, \alpha_{\mathrm{v}}, \alpha_{\mathrm{j}}, \alpha_{\mathrm{k}})$ (In-house)	(60, 0.2, 300, 0.02)
Elements of diagonal covariance $\boldsymbol{\Sigma}$ (Human3.6M)	0.1
Elements of diagonal covariance $\boldsymbol{\Sigma}$ (In-house)	0.05
Residual force scale	500

Table 3. Hyperparameters for experiments on Human3.6M [15] and our in-house motion dataset.

For both Human3.6M and our in-house motion dataset, our method uses the same hyperparameter settings unless stated otherwise. Table 3 summarizes the hyperparameter setting.

Policy Network. The learnable parts in the policy network are the two MLPs, *i.e.,* $\mathcal{U}_\theta$ inside the kinematic refinement unit and $\mathcal{V}_\theta$ inside the control generation unit. We use ReLU activations for both $\mathcal{U}_\theta$ and $\mathcal{V}_\theta$. The MLP $\mathcal{U}_\theta$ consists of hidden layers with size (256, 512, 256). The MLP $\mathcal{V}_\theta$ contains hidden layers with size (2048, 1024).

Policy Training. For Human3.6M, we train a single policy using data from all the training subjects and directly transfer the policy to test subjects, so it is a cross-subject experiment. For our in-house motion dataset, due to the large variation of body proportion and shape, we train a model for each subject using subject-specific data and test on separate data. All baselines are trained using the same data as our method. For learning the policy, each RL episode is constructed by randomly sampling a video segment of 200 frames from all training data. For the initial pose q_1 of the character, we initialize it to the refined kinematic pose $\widetilde{q}_1^{(n)}$. For the initial velocity $\dot{q}_1$, we set it to the kinematic velocity $\widetilde{\dot{q}}_1^{(n)}$ computed using finite differences.

The episode is terminated when the end frame is reached or the character's root height is 0.5 below the root height of the kinematic pose (i.e., to detect if the character has lost balance). We train the policy π_θ for 2000 epochs. For each epoch, we keep collecting data by sampling RL episodes until the total number of time steps reaches 50000. The reward weighting factors $(\alpha_p, \alpha_v, \alpha_j, \alpha_k)$ are set to (30, 0.2, 100, 0.02) for Human3.6M and (60, 0.2, 300, 0.02) for the in-house dataset. For Human3.6M, we only have access to ground-truth 3D joint positions but not ground-truth joint angles, so we use the refined kinematic pose as pseudo-ground truth (for regularization) when computing rewards that need ground-truth joint angles. The elements of the policy's diagonal covariance matrix Σ are set to 0.1 for Human3.6M and 0.05 for our in-house motion dataset. The residual forces η_t output by the policy is scaled by 500 before being input to the physics simulator. We use the proximal policy optimization (PPO [49]) to learn the policy π_θ. The discount factor for the Markov decision process (MDP) is 0.95. We use the generalized advantage estimator GAE(λ) [48] to compute the advantage estimate for policy gradient and the GAE coefficient λ is 0.95. At each epoch, the policy is updated 10 times using Adam [20] with a step size 5×10^{-5}. The clipping coefficient ϵ in PPO is set to 0.2. Since PPO is an actor-critic based method, it also learns a value function that mirrors the design of the policy but outputs a single value estimate. The value function is updated using Adam with a step size 3×10^{-4} whenever the policy is updated.