DOI: 10.1111/cgf.14634
ACM SIGGRAPH / Eurographics Symposium on Computer Animation 2022
D. L. Michels and S. Pirk
(Guest Editors)

Volume 41 (2022), Number 8

Neural3Points: Learning to Generate Physically Realistic Full-body Motion for Virtual Reality Users

Yongjing Ye[1,2], Libin Liu[3†], Lei Hu[1,2], and Shihong Xia[1,2‡]

[1]Institute of Computing Technology, Chinese Academy of Sciences, China
[2]University of Chinese Academy of Sciences, China
[3]Peking University, China

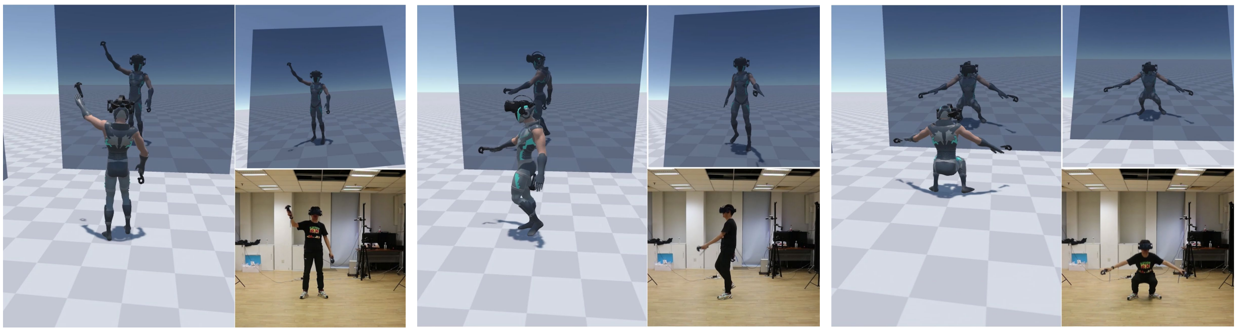


Figure 1: *We present a method for real-time full-body tracking using three VR trackers provided by a typical VR system: one HMD (head-mounted display) and two hand-held controllers. The bottom-right image of each sub-figure shows a user playing with the VR system. The other images show the simulated avatar of the user in the third-person point of view (left) and the first-person point of view (top-right), respectively. Virtual mirrors are placed in front of the user, so they can see their avatar easily.*

Abstract

Animating an avatar that reflects a user's action in the VR world enables natural interactions with the virtual environment. It has the potential to allow remote users to communicate and collaborate in a way as if they met in person. However, a typical VR system provides only a very sparse set of up to three positional sensors, including a head-mounted display (HMD) and optionally two hand-held controllers, making the estimation of the user's full-body movement a difficult problem. In this work, we present a data-driven physics-based method for predicting the realistic full-body movement of the user according to the transformations of these VR trackers and simulating an avatar character to mimic such user actions in the virtual world in real-time. We train our system using reinforcement learning with carefully designed pretraining processes to ensure the success of the training and the quality of the simulation. We demonstrate the effectiveness of the method with an extensive set of examples.

CCS Concepts

• ***Computing methodologies*** → ***Physical simulation; Virtual reality;*** *Motion capture;* • ***Theory of computation*** → ***Reinforcement Learning;***

1. Introduction

Allowing a user to see their body in the virtual reality (VR) world is an important part of an immersive experience. It enables natural interactions with the virtual environment and other users, and potentially allows the remote users to communicate and collaborate in a way as if they met in person. A VR system needs to track the full-body movement of a user in real-time and animate an avatar in the VR world that can faithfully reproduce the user's actions to support such an experience. However, typical consumer VR systems, such as HTC Vive and Occulus Quest, only provides up to three tracker devices in their default configurations, including a head-mounted

† Corresponding author:libin.liu@pku.edu.cn

‡ Corresponding author:xsh@ict.ac.cn

display (HMD) and optionally two hand-held controllers (HHC). Estimating full-body motions with high degrees of freedom using such limited information is an ill-formed problem. While it is relatively easy to compute good-quality upper body movement with carefully tuned IK solvers when the user is sitting or standing in place, finding correct heuristics for predicting plausible footsteps and lower-body motion is usually very difficult when the user needs to move around, and the solutions are often prone to artifacts such as foot-skating and unrealistic movement.

Data-driven methods, especially those learning generative models from massive motion capture data, have been proven to be an effective way to generate high-quality interactive character animation in recent years. The latent structure embedded in the motion data helps regularize the animation process and ensure the naturalness of the generated motions. Physics-based methods, on the other hand, have long been a promising avenue for creating realistic character animation, where physics-based simulations naturally prevent artifacts like foot-skating and can generate physically accurate motions. The recent advances in deep reinforcement learning have demonstrated promising results that flexible motion control strategies can be learned by imitating reference motion, making the combination of the data-driven and physics-based method a possible way to address the full-body motion tracking problem in VR as discussed above.

In this work, we present a data-driven physics-based method for estimating realistic full-body movement using up to three built-in VR tracker devices included in a typical VR system. Our system simulates an avatar character that can mimic user actions in the virtual world in real-time, which has the potential to enhance the immersive experience in VR applications. We train our system using reinforcement learning with carefully designed pretraining processes to ensure the success of the training and the quality of the simulation. The principal contributions of this work include: (1) We build a novel real-time full-body motion tracking system that generate physically realistic motions from sparse VR trackers. Our configuration is directly compatible with commercial VR systems, and potentially supports HMD-only VR systems. (2) We develop a full-body motion predictor module with decoupled upper-body and lower-body pose predictors and combine them via an aggregated representation of the state of the character. We find this network architecture performs better than the baseline methods that directly predict the full-body movement and is more robust with respect to unseen upper-body motions.

2. Related Work

Human motion capture plays an important role in character animation. In game and film industries, commercial motion capture solutions, such as Vicon [Vic] and Xsens [Xse] has been widely adopted to capture high quality human performance. These high-end mocap systems often use tens of optical markers or IMU sensors to achieve accurate capturing and high motion quality. Such a configuration is often expensive and not suitable for a commercial VR systems in everyday settings. Reconstructing full-body human motions using a small number of sensors or markers becomes more and more demanding given the popularization of the VR/AR devices. Many state-of-the-art approaches achieve this goal using six body-worn sensors on the user's head, limbs, and waist [LZWM06; LWC*11; vRBP17; HKA*18], which does not directly work with the out-of-box components of a typical VR systems. Research on estimating full-body human motion using even fewer tracking signals remains relatively sparse, where a sensor mounted on waist is often needed to reconstruct flexible lower-body motions. For example, [KSL12; KSL13] demonstrate systems based on kernel canonical correlation analysis (CCA) to predict full-body poses from five motion sensors mounted on a user's limbs and the waist. [WGR*19] consider a similar setting, but employ a LSTM-based model to reconstruct the full-body motion. [YKL21] show that plausible lower-body motions can be generated by a GRU-based model from four upper-body VR sensors mount on head, hands, and the waist. DeepMotion [Dee] offers a physics-based three-point full-body tracking solution. Their technique is based on a simplified control model and the simulated avatar can look robotic. [DDC*21] recently shows a VAE-based model can be trained to predict full-body poses from a single head-mounted device, but their approach does not predict root transformations. Unlike those approaches, our method enables natural and physically plausible full-body tracking using up to three VR trackers, which is compatible with typical commercial VR systems.

Synthesizing interactive human motions using low-dimensional control signals, such as those using a keyboard or a game controller to control a game character [SZKZ20; BC15] or manipulating a virtual puppet using hand or full-body gestures [SOL13; RTI*14], has been a long-term topic of character animation. The motion graphs and its variations [KGP02; HG07] has been widely adopted in game industry as a standard technique, where interaction can be achieved using either hand-crafted state-machines or control policies trained using reinforcement learning [WAH*10; TLP07]. Motion synthesis using unorganized motion data had attracted a lot of attentions in the past years. [LWB*10] introduced a framework that learns to blend the nearest neighbors of the character's pose in a dataset to achieve fast transitions between motions under user control. This method later inspired the development of Motion Matching methods [BC15; HKPP20]. Learning generative models from massive unorganized data is another promising way to achieve interactive control. Previous research has explored many statistic models, such as PCA [SHP04], mixture of Gaussian [MCC09; MC12], and Gaussian Process [LWH*12; WFH08]. More recently, deep generative models have demonstrated great potential in achieving realistic and interactive motion generation. Recent research has exploited popular models, such as GAN [WCX21], autoencoder [HSK16], VAE [LZCv20], and normalizing flows [HAB20], with many different network structures, such as CNN [HSK16], LSTM [LLL19; HYNP20], mixture of expert [SZKZ20; ZSKS18], and transformers [LYRK21]. Our system develops a data-driven generative model to predict a full-body poses from VR tracker inputs, with a carefully designed structure that reduces the coupling between motion generations of different parts of the character, thus allowing additional robustness with respect to unseen input.

Unlike data-driven methods, physics-based approaches explicitly incorporate physics simulation into the motion generation pipeline, which ensures physical accuracy of the generated motions and allows responses to unexpected perturbations. However, designing a physics-based controller for complex human skills has

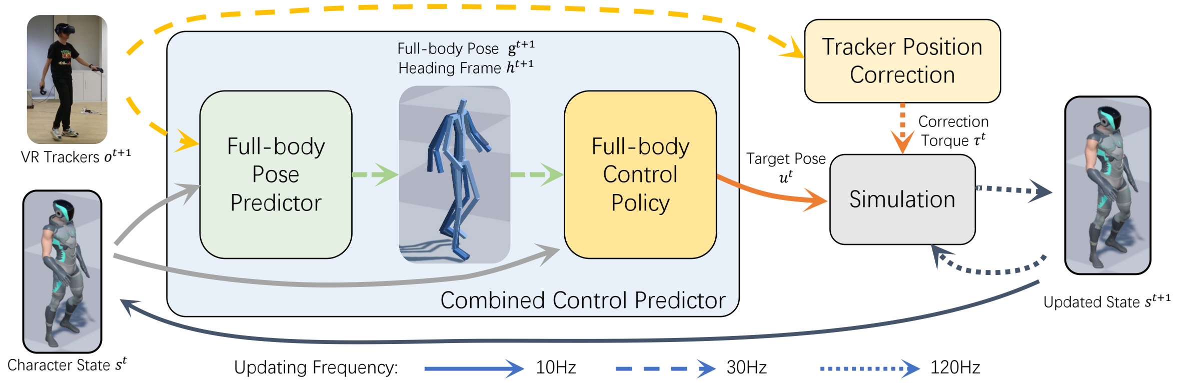


Figure 2: *The architecture of our system. Our system is composed of a full-body pose predictor, a full-body control policy, and a simulation module. The pose predictor takes as input the transformations of the VR trackers and predicts the full-body pose as well as the location and heading direction of the user. The control policy uses this information to calculate target poses for the PD controllers, combined with a tracker position correction module that compute necessary joint torques to compensate for the tracking error of the trackers. The simulation module then performs physics simulation and updates the character's state. This new state will be used as the input of the next frame.*

been a notorious challenge in computer graphics and robotics. Early approaches often rely on hand-crafted controllers [HWBO95; YLv07; LKL10], optimized feedback policies [TGLT14; LYvG12; WHDK12], abstract models [CBv10; MdH10; KH17], optimal control [MLPP09], and model predictive controllers [EHSN19; HHC*19]. The recent advancement of reinforcement learning (RL) makes imitating motion data a feasible way to learn control policies for complex skills [LvY16; PALv18; CMM*18]. To create a multi-skilled character, individual controllers can be organized and scheduled by high-level policies [PBYv17; LH17; MAP*19] or be used to train an integrated policy [MHG*19; MTA*20; WGH20]. Direct training of multi-skilled policies can also be achieved using a mixture of expert structure [PCZ*19; LSCC20] or with the help of adversarial losses [MTT*17]. Combining the advantage of both data-driven motion generators and RL-based control policy is another avenue to creating multi-skilled and interactive control policies [PRL*19; BCHF19], and similar ideas are also adopted to reduce the ambiguity caused by incomplete input signals [YPL21; SGXT20; XWI*21]. Our work also takes benefit of both the data-driven approaches and physics simulation to achieve physically realistic motion generation.

3. System Overview

The goal of our system is to reconstruct realistic full-body movement of the user according to the positions and orientations of three *VR trackers*, *i.e.* the head-mounted display (HMD) and two hand-held controllers (HHC), and to simulate an avatar character in the virtual world to reproduce the user's motion at real-time. As sketched in Figure 2, our system is composed of four major components operating at different timescales.

Our system obtains a stream of transformations of the VR trackers using the built-in functions provided by the VR systems. The frame rate of these input signals is assumed to be 30 Hz, where resampling is applied when necessary. The *Full-body Pose Predictor* module reads the transformation signals of the VR trackers and predicts the full-body pose as well as the location and heading direction of the user according to the current state of the avatar character. These estimations can then be used to update the character's pose directly. We refer to this updating strategy as the *direct mode* of our system.

The direct mode, however, often generates physically implausible results such as unrealistic foot sliding due to the lack of physical constraints. To create a physics-enhanced user experience, we employ a *Full-body Control Policy* module to take the estimated poses as a reference and compute a target pose that will be used to actuate the character to track the user's action. Then, the *Simulation* module is involved to simulate the character. We refer to this process as the *normal mode* of our system.

The user often changes their movement unpredictably, making it hard to track their global position and pose accurately in simulation. For example, the avatar character may generate an excessive speed to match the user's current position without knowing that the user is intending to stop immediately, in which case the momentum of the character may prevent it from stopping quickly and thus cause tracking errors. To deal with this problem, we opt for allowing a small amount of delay in the tracking to enable the system to prepare for the unpredictable changes. Specifically, we let the Full-body Control Policy module to operate at a coarse timescale of 10 Hz. When every three frames of the tracker input are received, the Full-body Pose Predictor is involved to predict a series of poses from the current simulation states of the character in an autoregressive manner. These estimations are then used by the Full-body Control Policy to compute the target pose. We thus refer to the full-body pose predictor and control policy jointly as a *Combined Control Predictor*.

The *Simulation* module performs physics simulation and updates the state of the character. In our system, the character is modeled as an articulated rigid body skeleton with a floating root, where PD-servos are employed to track the target poses provided by the control predictor and compute joint torques to actuate the internal

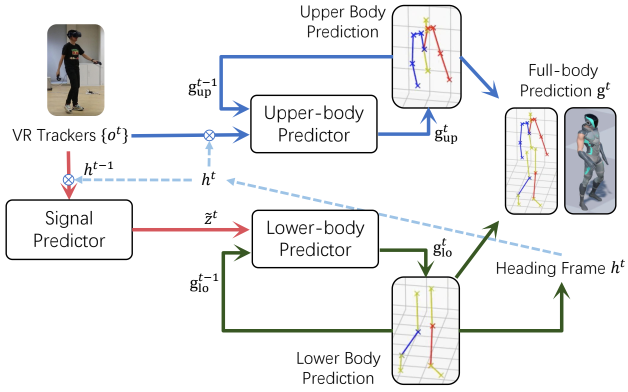


Figure 3: *The architecture of our pose predictor*

degrees of freedom of the character. We run simulation at a relatively high frequency, 120 Hz, to ensure numerical stability. The same target pose is used at every simulation step until the control predictor computes a new target.

Finally, an additional *Tracker Position Correction* module is involved to encourage the simulated character to follow the VR trackers accurately. This module operates at the same timescale as the simulation. It applies virtual forces to the hands of the simulated character, where the force is computed with a PD controller according to the tracking errors. These virtual forces are implemented as additional joint torques computed using Jacobian transpose control and applied to the corresponding arms.

The system is trained using the deep reinforcement learning framework by imitating prerecorded motions. To achieve good performance, we have captured one hour of unorganized performance data, where the subjects were asked to stand or walk in the capture volume while acting as if they were playing a VR game. We find a vanilla end-to-end training hard to converge for such an integrated system with many coupled components. To facilitate the training, we pretrain the full-body pose predictor using supervised learning and the control predictor by learning to track the reference motions, and then fine-tune them jointly using reinforcement learning.

4. Pose Predictor

The *full-body pose predictor* module of our system, represented by $\mathcal{G}$, estimates the user's pose and heading transformation according to the VR tracker input and the past states of the avatar character. Formally, the input to this module is a stream of transformations of the three VR trackers $\boldsymbol{o} = \{\boldsymbol{p}_X, \boldsymbol{q}_X\}$, where $\boldsymbol{p}_X$ and $\boldsymbol{q}_X$ are the location and orientation of tracker X in the world coordinate frame, and $X \in \{\mathrm{H}, \mathrm{L}, \mathrm{R}\}$ corresponds to the HMD and left/right hand-held controllers, respectively. Unless otherwise noted, we use quaternions to represent the orientation of the VR trackers and 3-D rotation vectors for the joint rotations and bone orientations of the character in this paper.

We formulate the pose predictor $\mathcal{G}$ as a recurrent model. Given a sequence of VR tracker input $\{\boldsymbol{o}^t\}, t = 1, \ldots, T$, an initial state of the character $\boldsymbol{g}^0$, and an initial heading transformation $\boldsymbol{h}^0$, the pose predictor generates a series of states $\{\boldsymbol{g}^t, \boldsymbol{h}^t\}, t = 1, \ldots, T$ autoregressively as

$$(\boldsymbol{o}_*^{t+1}, \boldsymbol{g}^t, \boldsymbol{h}^t) = \mathcal{G}(\boldsymbol{o}^t, \boldsymbol{g}^{t-1}, \boldsymbol{h}^{t-1}), \tag{1}$$

where the heading transformation $\boldsymbol{h}$ horizontally moves with the root of the character and has one axis vertically aligned and another aligned with the character's heading direction. We use a 3-tuple (p_x, p_z, θ_y) to represent $\boldsymbol{h}$, where (p_x, p_z) is the planar translation of the coordinate frame and θ_y corresponds to the rotation around the vertical axis. Inspired by [LLL19], we let $\mathcal{G}$ also predict the VR tracker input of the next time step, $\boldsymbol{o}_*^{t+1}$.

When using a VR application, a user's upper body typically exhibits higher range of motion than his lower body, and the movements of the upper body and the lower body are not always strongly correlated. For example, a user can swing his arms in multiple ways while either standing in-place or walking around. To deal with such degrees of freedom, we consider our character as two disjoint sets of joints and treat them separately. As shown in Figure 3, a lower-body pose predictor, $\mathcal{G}_{\mathrm{lo}}$, controls the joints of the character's legs, represented by J_{lo}, while an upper-body pose predictor $\mathcal{G}_{\mathrm{up}}$ handles the set of joints of the upper body and arms of the character, J_{up}. More specifically, given the locations and orientations of the three VR trackers, the lower-body pose predictor $\mathcal{G}_{\mathrm{lo}}$ predicts the global motion of the user in terms of the movement of the heading frame $\boldsymbol{h}$, and computes coordinated leg motions. The upper-body pose predictor $\mathcal{G}_{\mathrm{lo}}$ then computes an upper-body pose that follows the trackers' position in the predicted heading frame.

The state of the character is then represented as $\boldsymbol{g} = \{\boldsymbol{g}_{\mathrm{up}}, \boldsymbol{g}_{\mathrm{lo}}\}$, where $\boldsymbol{g}_{\mathrm{up}} = \{\boldsymbol{p}_j, \boldsymbol{v}_j, \boldsymbol{q}_j\}, j \in J_{\mathrm{up}}$ and $\boldsymbol{g}_{\mathrm{lo}} = \{\boldsymbol{p}_j, \boldsymbol{v}_j, \boldsymbol{q}_j, \boldsymbol{c}, z\}, j \in J_{\mathrm{lo}}$ consist of the position $\boldsymbol{p}_j$, velocities $\boldsymbol{v}_j$, and orientations $\boldsymbol{q}_j$ of the joints in the corresponding joint sets $J_{\mathrm{up/lo}}$, all computed in the reference heading coordinate frame $\boldsymbol{h}$. $\boldsymbol{c} = \{c_{\mathrm{L}}, c_{\mathrm{R}}\}$ contains two scalar variables $c_{\mathrm{L/R}} \in [0,1]$ indicating if the left foot and right foot are in contact with the ground or not, respectively. To lower the coupling between the two pose predictors $\mathcal{G}_{\mathrm{lo}}$ and $\mathcal{G}_{\mathrm{up}}$, we utilize an aggregated state of motion of the upper body, $\boldsymbol{z}$, to convey necessary information to the lower body, where $\boldsymbol{z} = \{\boldsymbol{p}_{\mathrm{up}}, \boldsymbol{v}_{\mathrm{up}}, \boldsymbol{L}_{\mathrm{up}}, \delta\boldsymbol{h}\}$ consists of the centroid position $\boldsymbol{p}_{\mathrm{up}}$, velocity $\boldsymbol{v}_{\mathrm{up}}$, and angular momentum $\boldsymbol{L}_{\mathrm{up}}$ of the character's upper body, as well as the change of heading frame $\delta\boldsymbol{h}$ from the last time step. All these quantities are computed in the current reference heading coordinate frame $\boldsymbol{h}$, except for $\delta\boldsymbol{h}$, which is computed with respect to the heading frame of the previous time step, so that the heading frame can be updated as $\boldsymbol{h}^t = \boldsymbol{h}^{t-1} \otimes \delta\boldsymbol{h}^t$, where $\otimes$ represents the multiplication of two transformations. At runtime, a dedicated *signal predictor* $\mathcal{G}_{\mathrm{sig}}$ is employed to predict this aggregated state $\boldsymbol{z}$ from the input signals of the three VR trackers $\boldsymbol{o}$, allowing the lower-body pose predictor to focus on the global motion of the user and less distracted by the diversity of the upper body motions.

As sketched in Figure 3, the entire pose prediction process of

Equation (1) can then be rewritten as

$$\begin{aligned}\tilde{\boldsymbol{z}}^t &= \mathcal{G}_{\text{sig}}\left(\text{inv}(\boldsymbol{h}^{t-1}) \otimes \boldsymbol{o}^t\right) \\ (\boldsymbol{z}_*^{t+1}, \boldsymbol{g}_{\text{lo}}^t) &= \mathcal{G}_{\text{lo}}(\tilde{\boldsymbol{z}}^t, \boldsymbol{g}_{\text{lo}}^{t-1}) \\ \boldsymbol{h}^t &= \boldsymbol{h}^{t-1} \otimes \delta\boldsymbol{h}^t, \quad \text{where } \delta\boldsymbol{h}^t \in \boldsymbol{g}_{\text{lo}}^t \\ (\boldsymbol{o}_*^{t+1}, \boldsymbol{g}_{\text{up}}^t) &= \mathcal{G}_{\text{up}}\left(\text{inv}(\boldsymbol{h}^t) \otimes \boldsymbol{o}^t, \boldsymbol{g}_{\text{up}}^{t-1}\right) \\ \boldsymbol{g}^t &\leftarrow \{\boldsymbol{g}_{\text{up}}^t, \boldsymbol{g}_{\text{lo}}^t\}\end{aligned} \tag{2}$$

where the operator inv and $\otimes$ represent the inversion and multiplication of transformations, respectively. Specifically, when receiving a set of VR tracker signals $\boldsymbol{o}^t$ at time t, the signal predictor $\mathcal{G}_{\text{sig}}$ first transforms $\boldsymbol{o}^t$ into the current heading frame $\boldsymbol{h}^{t-1}$ and computes a predicted aggregated upper-body state $\tilde{\boldsymbol{z}}^t$ accordingly. The lower-body predictor $\mathcal{G}_{\text{lo}}$ then takes $\tilde{\boldsymbol{z}}^t$ as input and predicts the lower-body state $\boldsymbol{g}_{\text{lo}}^t$. After updating the heading frame using the predicted $\delta\boldsymbol{h}^t$, which is included in $\tilde{\boldsymbol{z}}^t$, the VR tracker input $\boldsymbol{o}^t$ is transformed into the new reference heading coordinate frame $\boldsymbol{h}^t$, and then the upper-body predictor $\mathcal{G}_{\text{up}}$ estimates a new upper-body pose $\boldsymbol{g}_{\text{up}}^t$ according to it. Note that both $\mathcal{G}_{\text{lo}}$ and $\mathcal{G}_{\text{up}}$ also predict $\boldsymbol{z}_*^{t+1}$ and $\boldsymbol{o}_*^{t+1}$, the corresponding input signals of the next time step respectively. Finally, the predicted upper-body and lower-body states $\boldsymbol{g}_{\text{up}}^t$ and $\boldsymbol{g}_{\text{lo}}^t$ are put together to construct the full-body state $\boldsymbol{g}^t$. When a series of tracker input $\{\boldsymbol{o}^t\}$ is given, the pose predictor module repeats the above procedure and generates a sequence of states, forming a motion clip.

4.1. Pose Predictor Training

We implement the three sub-predictors: the signal predictor $\mathcal{G}_{\text{sig}}$, the lower-body predictor $\mathcal{G}_{\text{lo}}$, and the upper-body predictor $\mathcal{G}_{\text{up}}$, as recurrent neural networks, each consisting of three GRU (Gated Recurrent Unit) layers, combined with additional one-layer fully-connected encoder and decode layers. The number of hidden layer units of each network is set as 64, 128 and 128, respectively.

We employ a two-stage training process to train the pose predictor using the motion capture data, where each sub-predictor is pretrained separately with the input/output extracted from the motion data and then fine-tuned jointly while following the combined prediction process described above. Each training episode starts from a batch of starting states randomly chosen from the motion dataset, followed by generating a sequence of states of length T in an auto-regressive manner with the corresponding inputs extracted from the motion data. The objective functions are then evaluated on each generated state, whose gradients are used to update the networks. We use $T = 60$ and a batch size of 32 during the training. Two extra dropout layers with dropout rates of 0.1 and 0.05 respectively are applied after the encoder layer and before the decoder layer of each sub-predictor to prevent over-fitting during training. We use the Adam optimizer [KB14] to perform the gradient update with a learning rate of 5×10^{-4}.

The loss functions for the three sub-predictors are defined as

$$\mathcal{L}_{\text{sig}} = \mathcal{L}_{\tilde{\boldsymbol{z}}}^{\text{mse}} + w_{\boldsymbol{h}}\mathcal{L}_{\boldsymbol{h}} \tag{3}$$

$$\mathcal{L}_{\text{lo}} = \mathcal{L}_{\boldsymbol{g}_{\text{lo}}, \tilde{\boldsymbol{z}}_*}^{\text{mse}} + w_{\boldsymbol{h}}\mathcal{L}_{\boldsymbol{h}} + w_{\text{FK}}\mathcal{L}_{\text{FK}} \tag{4}$$

$$\mathcal{L}_{\text{up}} = \mathcal{L}_{\boldsymbol{g}_{\text{up}}, \tilde{\boldsymbol{o}}_*}^{\text{mse}} + w_{\text{FK}}\mathcal{L}_{\text{FK}}, \tag{5}$$

respectively, where the loss terms in the form of $\mathcal{L}_*^{\text{mse}}$ are weighted MSE (mean squared error) between the quantities represented by the subscripts and their corresponding ground truth. When training the signal predictor $\mathcal{G}_{\text{sig}}$ and the lower-body predictor $\mathcal{G}_{\text{lo}}$, an additional loss term $\mathcal{L}_{\boldsymbol{h}}$ is used to minimize the error of the estimated global heading transformation, computed by accumulating the sequence of predicted change of heading frames. The global heading transformation can change dramatically in a long motion sequence. To avoid the singularity of the rotation angle representation, we compute $\mathcal{L}_{\boldsymbol{h}}$ as

$$\begin{aligned}\mathcal{L}_{\boldsymbol{h}} = &\operatorname*{mse}_t\left[(p_x^t, p_z^t), (\bar{p}_x^t, \bar{p}_z^t)\right] \\ &+ \operatorname*{mse}_t\left[(\sin\theta_y^t, \cos\theta_y^t), (\sin\bar{\theta}_y^t, \cos\bar{\theta}_y^t)\right],\end{aligned} \tag{6}$$

where the symbols with a bar ($\bar{\ }$) indicates ground truth variables. In addition, to ensure consistency between the predicted joint orientations and joint positions, we perform forward kinematics (FK) according to the predicted joint orientations $\{\boldsymbol{q}_j\}, j \in J_{\text{up/lo}}$ and compute the MSE between the resulting joint positions and the predicted ones $\{\boldsymbol{p}_j\}, j \in J_{\text{up/lo}}$. We consider this MSE as an extra loss term $\mathcal{L}_{\text{FK}}$ in the training.

In the fine-tuning process, we train the entire pose predictor while minimizing the combined loss $\mathcal{L} = \mathcal{L}_{\text{sig}} + \mathcal{L}_{\text{lo}} + \mathcal{L}_{\text{up}}$. In practice, we freeze the parameters of the upper body predictor $\mathcal{G}_{\text{up}}$ and only update the signal predictor $\mathcal{G}_{\text{sig}}$ and the lower body predictor $\mathcal{G}_{\text{lo}}$ in this process, which helps stabilize the training and prevent degeneration.

5. Full-body Control Policy

As shown in the system overview of Figure 2, the *Full-body Control Policy* module, represented by π, converts the estimated full-body poses of the user into target poses, which will be used to actuate the simulated character using PD-servos. More specifically, the input to π is a state vector $\boldsymbol{s}^t = \{\boldsymbol{x}^t, \boldsymbol{g}^{t+k}\}, k = 1,2,3$, which contains the current simulation state $\boldsymbol{x}^t = \{\boldsymbol{p}_j^t, \boldsymbol{v}_j^t, \boldsymbol{q}_j^t\}, j \in J$ and three predicted states $\{\boldsymbol{g}^{t+k}\}, k = 1,2,3$ corresponding to 0.1, 0.2, and 0.3 seconds after the current time t respectively. We convert $\boldsymbol{x}^t$ into a reference global heading frame to inform the simulated character about the global tracking errors, which helps the character track the heading of the user accurately. $\{\boldsymbol{g}^{t+k}\}$ are expressed in the same heading frame as well. In the pretraining process, the reference heading is extracted from the reference motion. During the finetuning and at runtime, the reference heading is comuted by the Full-body Pose Predictor. The output of the policy is a target pose $\boldsymbol{u} = \{\boldsymbol{q}_j\}$ consisting of target rotations $\boldsymbol{q}_j$ of every internal joint $j \in J$. We implement the policy π as a simple feedforward neural network consists of four fully connected layers, with 256 units in each of the two hidden layers and ReLU as the activation function.

5.1. Full-body Control Policy Pretraining

We pretrain the full-body control policy using reinforcement learning on our motion capture dataset. Following the standard formulation of a reinforcement learning problem, the training process maximizes the expected cumulative reward

$$\pi^* = \arg\max_{\pi} \mathbb{E}_{\tau\sim\pi}\left[\sum_t \gamma^t R(\boldsymbol{s}^t)\right] \tag{7}$$

over all simulation trajectories $\tau = \{\boldsymbol{s}_0, \boldsymbol{u}_0, \boldsymbol{s}_1, \boldsymbol{u}_1, \dots\}$ induced by π, where $\boldsymbol{s} \in S$ is the state vector, $\boldsymbol{u} \in U$ is the action vector that stacks the joint rotations of a target pose, $R(\boldsymbol{s}^t)$ is the reward of state $\boldsymbol{s}^t$, and γ is the discount factor, which is set to 0.99 in our system.

We train the control policy using the PPO algorithm [SWD*17], which alternates the collection of simulation rollouts and the policy update. During training, each simulation rollout is initialized using a random state extracted from the reference motion and ends when either it is 300 control steps long or the character falls. A batch of 4000 control steps is collected in every update iteration. A separate value network V is used to compute values for each state, which is a feedforward network with two hidden layers of 256 units each. We use the Adam optimizer [KB14] to perform the gradient update. The learning rates for the policy network and the value network are $1\times10^{-5}$and 1×10^{-4}, respectively.

5.1.1. Reward

The goal of the pretraining is to make our simulated character to track the reference motions accurately, thus the reward at each time step is defined as

$$R(\boldsymbol{s}) = \exp(w_{\text{pose}}r_{\text{pose}} + w_{\text{orient}}r_{\text{orient}} + w_{\text{pos}}r_{\text{pos}} + \\ w_{\text{balance}}r_{\text{balance}} + w_{\text{contact}}r_{\text{contact}} + w_{\text{foot}}r_{\text{foot}}). \tag{8}$$

The pose reward r_{pose} encourages the character to match the reference pose locally, which is computed as

$$r_{\text{pose}} = -\frac{1}{|J|}\sum_{j\in J}\|\bar{\boldsymbol{q}}_j\boldsymbol{q}_j^{-1}\|_A + 0.1\|\bar{\boldsymbol{\omega}}_j - \boldsymbol{\omega}_j\|, \tag{9}$$

where $\|\boldsymbol{q}\|_A$ computes the angle of rotation of a quaternion $\boldsymbol{q}$, and $\boldsymbol{\omega}_j$ is the joint's rotational speed. The symbols with a bar ($\bar{}$) indicates the reference.

The link orientation and position reward penalizes the global tracking errors, in terms of the positions and orientations of the character's bones:

$$r_{\text{orient}} = -\frac{1}{|J|}\sum_{j\in J}\|\bar{\boldsymbol{q}}_j \ominus \boldsymbol{q}_j\| \tag{10}$$

$$r_{\text{pos}} = -\frac{1}{|J|}\sum_{j\in J}\|\bar{\boldsymbol{p}}_j - \boldsymbol{p}_j\|, \tag{11}$$

where all the quantities are compared in the global coordinate frame.

The balance reward encourages the relative position between the character's feet and its center of mass to match those from the reference, which helps the character stabilize its pose and maintain balance. We compute this term as

$$r_{\text{balance}} = -\sum_{X\in\{\text{L,R}\}} w_X(\|\bar{\boldsymbol{d}}_X - \boldsymbol{d}_X\| + \|\bar{\boldsymbol{v}}_X - \boldsymbol{v}_X\|), \tag{12}$$

where $\boldsymbol{d}_X = \boldsymbol{c} - \boldsymbol{p}_X$ is the distance between the center of mass $\boldsymbol{c}$ and the corresponding foot, and $\boldsymbol{v}_X$ is the velocity of the foot. We assume that a standing foot contributes more to balance than a moving foot, thus the weight of each reward term is adjusted according to the velocity of the feet as

$$w_X = \frac{\hat{w}_X}{\hat{w}_R + \hat{w}_L} \tag{13}$$

$$\hat{w}_X = \|\boldsymbol{v}_X^{\shortparallel}\| + 5\max(0, h_X - \varepsilon), \tag{14}$$

where $\boldsymbol{v}_X^{\shortparallel}$ represents the planar components of $\boldsymbol{v}_X$, and h_X is the height of the foot.

The contact reward penalizes the difference between the position of the character's foot and its reference when the foot is in contact with the ground:

$$r_{\text{contact}} = -\sum_{X\in\{\text{L,R}\}} c_X\|\bar{p}_X - p_X\|, \tag{15}$$

where c_X is the contact label of the foot predicted by the pose predictor.

At last, the foot height reward encourage the character to lift its feet to clear the ground when moving, which is computed as

$$r_{\text{foot}} = -\sum_{X\in\{\text{L,R}\}} \|\bar{h}_X - h_X\|. \tag{16}$$

In training, all of the weights in Equation (8) are set as 10 except that w_{pose} is set as 5.

5.1.2. Adaptive State Initialization

As also observed by previous works [PRL*19; CMM*18], when training a tracking policy to perform a diverse set of motions, choosing the random initial states uniformly can cause the training to overfit to the motions that are easy to learn. To mitigate this problem, we employ an adaptive state initialization strategy which encourages the training process to start a rollout from a less visited state. Specifically, we draw initial states from a multinomial distribution, with the probability periodically updated so that a state is chosen based on its value. A state with lower value will have higher chance to be selected as a starting state. Specifically, the probability is updated by

$$P(\bar{\boldsymbol{s}}) = \frac{\exp(-V(\bar{\boldsymbol{s}})/T_w)}{\sum_{\bar{\boldsymbol{s}}}\exp(-V(\bar{\boldsymbol{s}})/T_w)}, \tag{17}$$

where V is the value function of the RL problem. The temperature parameter T_w is empirically set to 5. During the training, we update this distribution of the initial states every 100 training iterations.

5.1.3. Pre-Pretraining

We jump-start the training by initializing the tracking policy π using supervised learning, which significantly accelerate the training at the early stage. The training data of this pre-pretraining is created by constructing open-loop control trajectory for our motion capture data using the SAMCON algorithm [LYv*10; LYG15] and extracting corresponding state-action pairs from the simulation. To prevent overfitting, we apply dropout before each fully connected layers of π with the dropout rate of (0.1, 0.1, 0.05, 0.02) respectively. These dropout layers are disabled in the above reinforcement learning process.

6. Combined Control Predictor

As depicted in Figure 2, the two core components of our system, the Full-body Pose Predictor $\mathcal{G}$ and the Control Policy π, run jointly as a *Combined Control Predictor.* To prepare for unpredictable movement from users, a little delay in the tracking is allowed. Specifically, the Full-body Pose Predictor module operates at a coarse timescale of 10 Hz. When every three frames of the tracker input are received, a *signal forecasting* strategy is involved to predict six future frames of the input as:

$$\boldsymbol{o}^t = \delta\boldsymbol{o} + \boldsymbol{o}^{t-1} \quad t > 3 \tag{18}$$

where $\delta\boldsymbol{o}$ is the average offset between consecutive input frames. The Full-body Pose Predictor then takes these extended tracker inputs and predicts a short motion clip of nine frames recursively. The initial pose of this prediction is extracted from the current state of the simulated character, while the previous prediction of the user's heading is employed as the the initial heading. We further apply inverse kinematics in the way similar to [ZSKS18] to ensure accurate tracking of the input trackers. The Control Policy module then compute a target pose according to this reference motion clip. At last, the simulation advances at 120 Hz until the next 0.1-second interval starts.

6.1. Combined Control Fine-tuning

After pretraining the full-body pose predictor and the control policy, our system fine-tunes them together as the combined control predictor using an additional reinforcement learning process. During the training, we extract transformations of the VR trackers from the motion capture data, and encourages the character to match the reference motions as closely as possible. The same rewards as the pretraining of the full-body control policy are used in this fine-tuning process. We use PPO again to train the combined policy, and a progressive learning approach is adopted to facilitate the training. The tracking position correction and the signal forecasting components are disabled at the beginning until the training process has plateaued. Then, the training continues with all the components enabled, while the motion generated by the pose predictor is used as the reference for the rewards. We find that the pose predictor $\mathcal{G}$ is prone to degeneration in the training without effective regularization. To mitigate this issue, the parameters of $\mathcal{G}$ is frozen in the fine-tuning.

6.2. Tracker Position Correction

To ensure accurate tracking of the VR trackers, we employ an additional Tracker Position Correction module in our system. This module applies virtual forces on the avatar's hands to correct their position. Each force is computed using PD control:

$$F_{virtual} = k_p * (\bar{\boldsymbol{p}} - \boldsymbol{p}) - k_d * \dot{\boldsymbol{p}}, \tag{19}$$

where $\boldsymbol{p}$ is the position of a hand and $\dot{\boldsymbol{p}}$ is its linear velocity. $\bar{\boldsymbol{p}}$ is the position of the corresponding VR tracker. k_p and k_d here are set to 1000 and 10, respectively. These virtual forces are not directly applied to the hands of the simulated character. Instead, the character tries to realize such virtual forces using joint torques. The joint torques are computed using Jacobians transpose control, thus ensuring zero accumulated external forces and torques so that the simulation is still physically correct.

7. Implementation Details

Our system is implemented in python, where the networks are built and trained with PyTorch. We simulate an avatar character that is 1.75 m tall and weighs 63 kg using a proprietary physics engine based on joint dynamics in generalized coordinates. The character is modeled as an articulated rigid body skeleton with a floating root and actuated by PD-servos. We employ implicit joint damping to stabilize the PD control as suggested by several previous works [TLT11; LvY16; PALv18], which allows stable simulation with a relatively large time step at 120 Hz. The PD-gains of the PD-servos are set to $k_p = 200$ and $k_d = 20$ for all the joints of the character.

We have motion captured one hour of unorganized performance data using an OptiTrack [Opt] motion capture system. The subjects were asked to stand or walk in the capture volume while acting as if they were playing a VR game. The motion are then retargeted to the simulated avatar by copying the rotations of the corresponding joints. We implement our system on a computer with Intel Xeon Gold 6252 CPU (24 cores, 2.10 GHz). The pre-training of the pose predictor and the tracking policy takes about 72 hours. And then fine-tuning the pose predictor takes about 48 hours. Furthermore, it requires about 120 hours for combined control fine-tuning. In conclusion, the overall training procedure needs 240 hours.

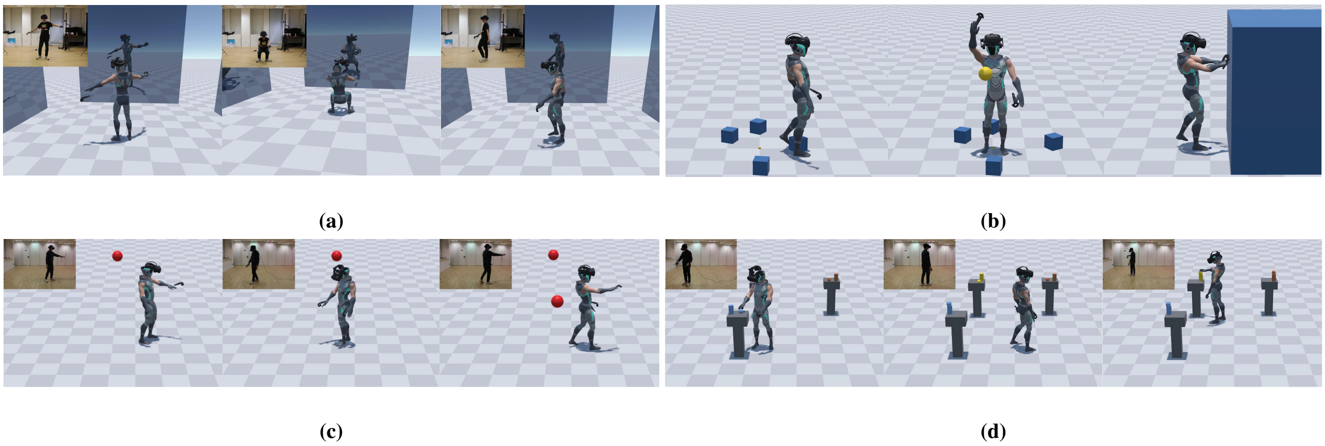
The VR environments are built with Unity and SteamVR plugin to communicate with a HTC Vive VR system. The VR applications and our framework in python run in separate processes on a modern computer with a multicore CPU and communicate with each other through a TCP connection. The entire system runs faster than real time, ensuring a smooth user experience without lagging.

To allow our framework to work with different VR systems with HMD and hand-held controllers of various sizes, we consider the input to the full-body pose predictor as the location of the user's wrists and neck. A calibration processes is implemented to convert the true transformations of the VR trackers into the corresponding input signals, which is achieved by asking the user to perform a T-pose and measuring the distance between the VR devices and the corresponding joints.

The pose predictor $\mathcal{G}$ can be used as a standalone module where the output full-body state is used to drive the avatar character directly, which we refer to as the *direct mode* of our system, as oppose to the *normal* model where the control and simulation are involved. We adopt an extra inverse kinematics approach similar to that was used in [SZKZ20] to enforce foot contacts and achieve accurate tracker positions in this mode.

8. Results

We demonstrate a variety of examples where a user plays in VR with our system. These are best seen in the accompany video. Snapshots of these examples are available in Figure 1, 4a, 4b, 4c, 4d and 5. We also conduct ablation studies to validate our design of the system.



(a) (b) (c) (d)

Figure 4: *Four test VR scenes. (a) A user stands, walks, and squats while swinging arms in the virtual environment. Mirrors are placed around the user to allow them to check the action of the virtual avatar. (b) A participant interacts with simulated objects by stepping on or pushing them while being shot by balls from random directions. (c) A player walks around and eliminates balls randomly spawned in the scene. (d) A participant rearranges the objects with the same color onto the same tables.*

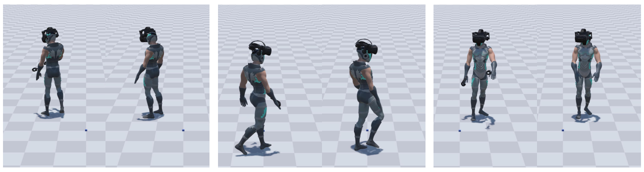


Figure 5: *Comparison between one-point tracking and three-point tracking. The character on the left of each figure uses the input of all three trackers. The character on the right uses the same set of input, but only the head tracker is considered.*

8.1. Three-Point Tracking in VR Scenes

We first test our full-body three-point tracking system for a wide range of motion performed by a participant in a VR environment. Before playing, the participant is required to perform a T-pose, when the distance between the VR trackers and the corresponding joints are measured as described in Section 7.

We have implemented four VR scenes for this test. In the first VR scene, the user can move freely in the VR environment. Several virtual mirrors are placed around the user so that they can check the action of his virtual avatar in real-time. In the second scene, we put several simulated objects around the character and shoot it using balls from random directions. The user can interact with these objects by stepping on or pushing them, while their avatar responds to the environment changes or when hit by the balls. The third VR scene is a mini-game, where the player needs to move and eliminate randomly spawned objects by touching them. In the last VR environment, the participant is asked to rearrange objects of different shapes and move objects with the same color together.

We test the performance of the system in both its normal model and the direct mode as described in Section 7. In all the experiments, our system successfully reconstruct the full-body movement of the user from the input of the three VR trackers. The direct mode has better tracking accuracy while suffering from unstable foot stepping and foot sliding occasionally. In contrast, the normal mode is more robust to unseen input and has no foot sliding. The motion generated in the normal mode is more natural and has realistic physical details. For example, when the character is standing and waving his hand, the upper body in the normal mode will sway slightly with the waving hand, which can not been generated in the direct mode. However, in the normal mode, the user can experience a short delay and less tracking accuracy. The tracking accuracy can be referred in the first-person perspective scene placed in the lower-left corner of each result in our supplementary video.

8.2. One-Point Tracking with HMD

We train our full-body pose predictor model to also predict the VR tracker input of the next time step, as indicated in Equation (1), which allows our system to reconstruct a full-body motion with fewer or even a single VR tracker. It is often the case when a VR system provides only a head-mounted display (HMD) in its default configuration, such as the Google Daydream. We demonstrate this capacity of our system by excluding the signals of the two hand-held controllers from the VR tracker input and only use the head tracker to reconstruct the full-body motion. Note that we have made our full-body pose predictor to predict the transformations of the trackers of the next frame in Equation (2). Our system thus considers these estimations as the input of the missing hand-held controllers. As shown in Figure 5 and also in the supplementary video, our system successfully reconstructs plausible full-body poses using only the HMD input.

8.3. Choice of Network architecture

We employ a decoupled network architecture for the full-body pose predictor module $\mathcal{G}$, where an aggregated representation of the upper-body state and the global motion of the user is used to convey necessary information among sub-predictors. We believe this architecture helps reduce the coupling between each individual component and improves the robustness of the predictor with respect to unseen inputs.

To validate this design, we compare the performance of our network architecture with two baseline networks: a RNN-based full-body pose generator $\mathcal{G}_{\text{full-body}}$ and a phase-functioned neural network $\mathcal{G}_{\text{PFNN}}$ [HKS17]. $\mathcal{G}_{\text{full-body}}$ is implemented to replace the decoupled networks $\mathcal{G}_{\text{lo}}$ and $\mathcal{G}_{\text{up}}$. It takes the same input but computes the full-body motions directly. The network structure of $\mathcal{G}_{\text{full-body}}$ is the same as $\mathcal{G}_{\text{lo}}$, except for the dimension of the input and output layers. This structure is inspired by [LLL19], but we employ GRUs instead of the LSTM as the recurrent units. The phase-functioned neural network $\mathcal{G}_{\text{PFNN}}$ is implemented similarly to [HKS17], where four expert networks are blended using the phase parameter computed according to the contact labels. We model these experts as feed-forward neural networks consisting of three full-connected layers, and the number of hidden layer units is set to 512. The input to $\mathcal{G}_{\text{PFNN}}$ is a sequence of recorded tracker signals $\{\boldsymbol{o}^{t+k}\}$ converted into the current heading frame, where t represent the current frame, and $k \in [-0.5\,s, 0.3\,s]$ sampled every 0.1 seconds.

We compare the performances of these three network architectures, $\mathcal{G}_{\text{ours}}$, $\mathcal{G}_{\text{full-body}}$, and $\mathcal{G}_{\text{PFNN}}$, in the *direct mode* of our system. Inspired by [SZKZ20], the assessment is based on the contact accuracy measured as the amount of foot skating, or the average horizontal velocity of the foot when it is considered to be in contact with the ground. In this experiment, we consider both the contact label predicted by the network and that computed based on the height of the foot to calculate the amount of the foot skating.

We train each network model on a small motion dataset of 36900 frames, 30fps. The assessment is performed on four test motion clips, labeled as test set 1, 2, 3 and 4, respectively. Test set 1 is one of the training motions, where the character walks around randomly. Test set 2 is a similar motion to test set 1 but is not used in the training. Test set 3 is a synthesized motion, where the lower-body motion is taken from test set 1 but the upper-body motion are replaced by another motion clip with dramatic arm movement, which is not used in the training. Test set 4 is the test set 2 augmented in the same way as test set 3. All these test motion clips are clipped to 30 seconds long. The results are reported in Figure 6 and 7, which shows that our pose predictor generates motions with more stable foot contacts than the baselines, indicating that our decoupled network architecture is more robust to unseen upper-body input. As a reference, Figure 6 and 7 also show the performance of the normal mode of our system on the same test sets. It can be seen clearly that foot skating is effectively eliminated in the normal mode using the physics-based simulation. Note we also measure the stability of the foot contacts of the ground-truth motion in these figures, where the small amount of foot skating is due to the errors accumulated during motion capturing and retargeting.

8.4. Effectiveness of the Tracker Position Correction

We employ the Tracker Position Correction module in our system to improve the accuracy of tracking the positions of the hand-held trackers. To evaluate the effectiveness of this component, we employ a 30-second test sequence of the VR tracker signals recorded at 30 fps, where the user walks around and waves their hand randomly. We then evaluate the tracking results with and without this component both visually and quantitatively. While the generated lower-body movements are similar in both the settings, the generated hand positions match the corresponding input trackers better with this module on than turning it off. This can be seen quantitatively in Table 1. Note that considering that our system bears a 0.1-second delay in the normal mode, we shift the generated motion forward by 0.1 seconds for a better evaluation of the accuracy. The results are shown under the *Delay Removed* column of Table 1.

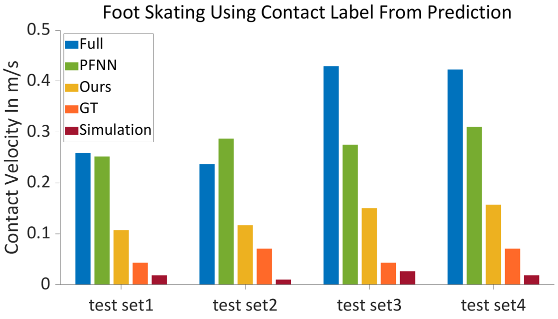


Figure 6: *Foot contact stability for different network architectures using contact labels predicted by the network.*

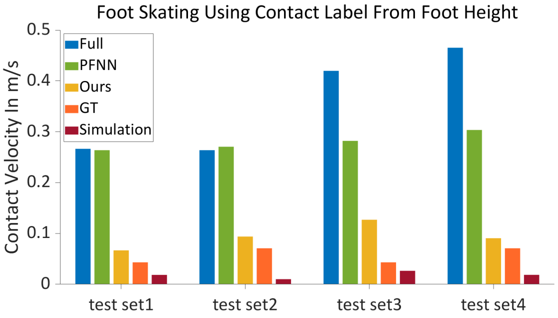


Figure 7: *Foot contact stability for different network architectures using contact labels calculated from foot height threshold.*

8.5. Validation on Full-body Tracking Accuracy

To further validate our system, we evaluate the accuracy of the full-body tracking on a 30-second test sequence extracted from our mocap dataset. The test sequence, where the character walks around while waving hands randomly, is not used in the training process. We use MPJPE [IPOS13] as our evaluation metric, which is wildly used in the human pose estimation problem. We compute this metric based on the global positions of the joints in both the reconstructed motion and the reference. Considering that the normal mode has a time delay of 0.1 seconds, we shift the generated motion forward by 0.1 seconds for better evaluation. The quantity results are shown in Table 2. Note that there are no explicit control of the character's lower body in the decoupled design of our pose predictor. The generated lower-body motion can be slightly

Table 1: *Ablation study of the Tracker Position Correction component. In the Delay Removed configuration, we shift the generated motion forward by 0.1 seconds to compensate the delay of our system, which provides a more accurate evaluation of the tracking results.*

Tracker Position Correction	Delay Removed	Positional Error (cm)
✓	✗	9.95 ± 4.71
✗	✗	15.09 ± 6.38
✓	✓	8.86 ± 3.56
✗	✓	12.21 ± 5.84

Table 2: *Tracking accuracy in different modes. The upper-body and lower-body accuracy are computed as the MPJPE of the joint groups J_{up} and J_{lo} discussed in the Section 4, respectively. The generated motion is shifted forward by 0.1 seconds in the Delay Removed mode to allow a more accurate evaluation.*

Tracking Accuracy	Mode Type	Delay Removed	MPJPE (cm)
full-body	Normal	✓	9.03 ± 4.10
upper-body	Normal	✓	7.66 ± 4.09
lower-body	Normal	✓	12.52 ± 5.68
full-body	Normal	✗	11.81 ± 5.11
upper-body	Normal	✗	10.68 ± 5.29
lower-body	Normal	✗	14.75 ± 6.38
full-body	Direct	-	5.52 ± 4.73
upper-body	Direct	-	3.76 ± 2.51
lower-body	Direct	-	7.88 ± 5.85

different from the user's actual motion, causing a relatively large full-body tracking errors.

9. Discussion

In this paper, we have presented a novel data-driven physics-based system for reconstructing full-body motions using a very sparse set of up to three VR trackers. Our prototype system can works with a typical VR system with its out-of-the-box functions to offer an enhanced immersive experience in virtual reality applications. We have developed a full-body motion predictor module with decoupled upper-body and lower-body pose predictors to achieve a robust pose estimation, where the two components are combined via an aggregated representation of the state of the character. We find this network architecture outperforms the baseline methods that directly predict the full-body movement and is more robust with respect to unseen upper-body motions. We have trained a full-body control policy that controls a simulated character to mimic the user's action based on the prediction of the pose predictor module, which generates physically plausible motions with enriched details and allows the user's avatar to interact with the simulated environment and respond to perturbations. To the best of our knowledge, we are the first deep-learning based three-point tracking system that achieves real-time tracking and simulation of full-body motions using such a small number of positional sensors.

Our method has several limitations. First, our system only predicts lower-body motions that are the most probable according to the transformations of the upper-body VR trackers. Special lower-body motions, such as kicking and standing while swinging legs, are hard to predict using such limited information. It would be an interesting future work to include the information of the virtual/real environment as a part of the input to help determine the correct motion.

Second, while our decoupled pose predictor module is robust to unseen upper-body motions, the performance of the lower-body pose predictor is limited by the training data. We find that our system can generate excessive foot skating in the direct mode when the user turns too fast, steps back quickly while turning, or acts with complex leg-crossing. The control policy and simulation in the normal mode of our system can remove the foot skating, but the character can perform unstably and may fall over, As a fallback, we can reset the simulated character to the state predicted by the pose predictor until the system resumes stable tracking. Including additional data with dynamic motions would be helpful to mitigate this problem, while it remains a future work to develop a robust motion generator that can generalize to control signals with a different distribution from that of the training data.

Third, the normal mode of our system has a small amount of time delay, which may affect the user experience. While this time delay is intended in our system to allow additional information to be collected to track the user's motion accurately, reducing it to a more acceptable level will be a goal for future research.

And lastly, our full-body pose-predictor and control policy are trained based on the same simulated character. While it is relatively easy to scale the input and output accordingly to support users of different heights, generalizing our system to support users with different body ratios and simulate non-human avatars remains a future work.

Acknowledgment

This work is partially supported by the National Key R&D Program of Science and Technology for Winter Olympics (No.2020YFF0304701) and the National Natural Science Foundation of China (No.61772499).

References

[BC15] BUTTNER, MICHAEL and CLAVET, SIMON. "Motion Matching - The Road to Next Gen Animation". *Proc. of Nucl.Ai*. 2015 2.

[BCHF19] BERGAMIN, KEVIN, CLAVET, SIMON, HOLDEN, DANIEL, and FORBES, JAMES RICHARD. "DReCon: Data-Driven Responsive Control of Physics-Based Characters". *ACM Transactions on Graphics* 38.6 (Nov. 2019), 1–11 3.

[CBv10] COROS, STELIAN, BEAUDOIN, PHILIPPE, and VAN DE PANNE, MICHIEL. "Generalized Biped Walking Control". *ACM Transactions on Graphics* 29.4 (July 2010), 130:1–130:9 3.

[CMM*18] CHENTANEZ, NUTTAPONG, MÜLLER, MATTHIAS, MACKLIN, MILES, et al. "Physics-Based Motion Capture Imitation with Deep Reinforcement Learning". *Proceedings of the 11th Annual International Conference on Motion, Interaction, and Games*. MIG '18. New York, NY, USA: Association for Computing Machinery, Nov. 2018, 1–10 3, 6.

[DDC*21] DITTADI, ANDREA, DZIADZIO, SEBASTIAN, COSKER, DARREN, et al. "Full-Body Motion From a Single Head-Mounted Device: Generating SMPL Poses From Partial Observations". *International Conference on Computer Vision 2021*. Oct. 2021 2.

[Dee] DEEPMOTION. *DEEPMOTION - VR Tracking*. https://www.deepmotion.com/virtual-reality-tracking 2.

[EHSN19] EOM, HAEGWANG, HAN, DASEONG, SHIN, JOSEPH S., and NOH, JUNYONG. "Model Predictive Control with a Visuomotor System for Physics-Based Character Animation". *ACM Transactions on Graphics* 39.1 (Oct. 2019), 3:1–3:11 3.

[HAB20] HENTER, GUSTAV EJE, ALEXANDERSON, SIMON, and BESKOW, JONAS. "MoGlow: Probabilistic and Controllable Motion Synthesis Using Normalising Flows". *ACM Transactions on Graphics* 39.6 (Nov. 2020), 236:1–236:14 2.

[HG07] HECK, RACHEL and GLEICHER, MICHAEL. "Parametric Motion Graphs". *Proceedings of the 2007 Symposium on Interactive 3D Graphics and Games*. I3D '07. Seattle, Washington: Association for Computing Machinery, 2007, 129–136 2.

[HHC*19] HONG, SEOKPYO, HAN, DASEONG, CHO, KYUNGMIN, et al. "Physics-Based Full-Body Soccer Motion Control for Dribbling and Shooting". *ACM Transactions on Graphics* 38.4 (July 2019), 74:1–74:12 3.

[HKA*18] HUANG, YINGHAO, KAUFMANN, MANUEL, AKSAN, EMRE, et al. "Deep Inertial Poser: Learning to Reconstruct Human Pose from Sparse Inertial Measurements in Real Time". *ACM Transactions on Graphics* 37.6 (Dec. 2018), 185:1–185:15 2.

[HKPP20] HOLDEN, DANIEL, KANOUN, OUSSAMA, PEREPICHKA, MAKSYM, and POPA, TIBERIU. "Learned Motion Matching". *ACM Transactions on Graphics* 39.4 (July 2020), 53:53:1–53:53:12 2.

[HKS17] HOLDEN, DANIEL, KOMURA, TAKU, and SAITO, JUN. "Phase-Functioned Neural Networks for Character Control". *ACM Transactions on Graphics* 36.4 (July 2017), 42:1–42:13 9.

[HSK16] HOLDEN, DANIEL, SAITO, JUN, and KOMURA, TAKU. "A Deep Learning Framework for Character Motion Synthesis and Editing". *ACM Transactions on Graphics* 35.4 (July 2016), 138:1–138:11 2.

[HWBO95] HODGINS, JESSICA K., WOOTEN, WAYNE L., BROGAN, DAVID C., and O'BRIEN, JAMES F. "Animating Human Athletics". *Proceedings of the 22nd Annual Conference on Computer Graphics and Interactive Techniques*. SIGGRAPH '95. New York, NY, USA: Association for Computing Machinery, Sept. 1995, 71–78 3.

[HYNP20] HARVEY, FÉLIX G., YURICK, MIKE, NOWROUZEZAHRAI, DEREK, and PAL, CHRISTOPHER. "Robust Motion In-Betweening". *ACM Transactions on Graphics* 39.4 (July 2020), 60:60:1–60:60:12 2.

[IPOS13] IONESCU, CATALIN, PAPAVA, DRAGOS, OLARU, VLAD, and SMINCHISESCU, CRISTIAN. "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 (2013), 1325–1339 9.

[KB14] KINGMA, DIEDERIK and BA, JIMMY. "Adam: A Method for Stochastic Optimization". *International Conference on Learning Representations* (Dec. 2014) 5, 6.

[KGP02] KOVAR, LUCAS, GLEICHER, MICHAEL, and PIGHIN, FRÉDÉRIC. "Motion Graphs". *ACM Transactions on Graphics* 21.3 (July 2002), 473–482 2.

[KH17] KWON, TAESOO and HODGINS, JESSICA K. "Momentum-Mapped Inverted Pendulum Models for Controlling Dynamic Human Motions". *ACM Transactions on Graphics* 36.1 (Jan. 2017), 10:1–10:14 3.

[KSL12] KIM, JONGMIN, SEOL, YEONGHO, and LEE, JEHEE. "Realtime Performance Animation Using Sparse 3D Motion Sensors". *Motion in Games*. Ed. by KALLMANN, MARCELO and BEKRIS, KOSTAS. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer, 2012, 31–42 2.

[KSL13] KIM, JONGMIN, SEOL, YEONGHO, and LEE, JEHEE. "Human Motion Reconstruction from Sparse 3D Motion Sensors Using Kernel CCA-based Regression". *Computer Animation and Virtual Worlds* 24.6 (2013), 565–576 2.

[LH17] LIU, LIBIN and HODGINS, JESSICA. "Learning to Schedule Control Fragments for Physics-Based Characters Using Deep Q-Learning". *ACM Transactions on Graphics* 36.4 (June 2017), 42a:1 3.

[LKL10] LEE, YOONSANG, KIM, SUNGEUN, and LEE, JEHEE. "Data-Driven Biped Control". *ACM Transactions on Graphics* 29.4 (July 2010), 129:1–129:8 3.

[LLL19] LEE, KYUNGHO, LEE, SEYOUNG, and LEE, JEHEE. "Interactive Character Animation by Learning Multi-Objective Control". *ACM Transactions on Graphics* 37.6 (Jan. 2019), 1–10 2, 4, 9.

[LSCC20] LUO, YING-SHENG, SOESENO, JONATHAN HANS, CHEN, TRISTA PEI-CHUN, and CHEN, WEI-CHAO. "CARL: Controllable Agent with Reinforcement Learning for Quadruped Locomotion". *ACM Transactions on Graphics* 39.4 (July 2020), 38:38:1–38:38:10. arXiv: 2005.03288 3.

[LvY16] LIU, LIBIN, VAN DE PANNE, MICHIEL, and YIN, KANGKANG. "Guided Learning of Control Graphs for Physics-Based Characters". *ACM Transactions on Graphics* 35.3 (May 2016), 29:1–29:14 3, 7.

[LWB*10] LEE, YONGJOON, WAMPLER, KEVIN, BERNSTEIN, GILBERT, et al. "Motion Fields for Interactive Character Locomotion". *ACM Transactions on Graphics* 29.6 (Dec. 2010), 138:1–138:8 2.

[LWC*11] LIU, HUAJUN, WEI, XIAOLIN, CHAI, JINXIANG, et al. "Real-time Human Motion Control with a Small Number of Inertial Sensors". *Symposium on Interactive 3D Graphics and Games*. I3D '11. New York, NY, USA: Association for Computing Machinery, Feb. 2011, 133–140 2.

[LWH*12] LEVINE, SERGEY, WANG, JACK M., HARAUX, ALEXIS, et al. "Continuous Character Control with Low-Dimensional Embeddings". *ACM Transactions on Graphics* 31.4 (July 2012), 28:1–28:10 2.

[LYG15] LIU, LIBIN, YIN, KANGKANG, and GUO, BAINING. "Improving Sampling-Based Motion Control". *Computer Graphics Forum* 34.2 (May 2015), 415–423 6.

[LYRK21] LI, RUILONG, YANG, SHAN, ROSS, DAVID A., and KANAZAWA, ANGJOO. "Learn to Dance with AIST++: Music Conditioned 3D Dance Generation". *arXiv:2101.08779 [cs]* (Feb. 2021). arXiv: 2101.08779 [cs] 2.

[LYv*10] LIU, LIBIN, YIN, KANGKANG, VAN DE PANNE, MICHIEL, et al. "Sampling-Based Contact-Rich Motion Control". *ACM Transactions on Graphics* 29.4 (July 2010), 128:1–128:10 6.

[LYvG12] LIU, LIBIN, YIN, KANGKANG, VAN DE PANNE, MICHIEL, and GUO, BAINING. "Terrain Runner: Control, Parameterization, Composition, and Planning for Highly Dynamic Motions". *ACM Transactions on Graphics* 31.6 (Nov. 2012), 1–10 3.

[LZCv20] LING, HUNG YU, ZINNO, FABIO, CHENG, GEORGE, and VAN DE PANNE, MICHIEL. "Character Controllers Using Motion VAEs". *ACM Transactions on Graphics* 39.4 (July 2020), 40:40:1–40:40:12 2.

[LZWM06] LIU, GUODONG, ZHANG, JINGDAN, WANG, WEI, and MCMILLAN, LEONARD. "Human Motion Estimation from a Reduced Marker Set". *Proceedings of the 2006 Symposium on Interactive 3D Graphics and Games*. I3D '06. New York, NY, USA: Association for Computing Machinery, Mar. 2006, 35–42 2.

[MAP*19] MEREL, JOSH, AHUJA, ARUN, PHAM, VU, et al. "Hierarchical Visuomotor Control of Humanoids". *arXiv:1811.09656 [cs]* (Jan. 2019). arXiv: 1811.09656 [cs] 3.

[MC12] MIN, JIANYUAN and CHAI, JINXIANG. "Motion Graphs++: A Compact Generative Model for Semantic Motion Analysis and Synthesis". *ACM Transactions on Graphics* 31.6 (Nov. 2012), 153:1–153:12 2.

[MCC09] MIN, JIANYUAN, CHEN, YEN-LIN, and CHAI, JINXIANG. "Interactive Generation of Human Animation with Deformable Motion Models". *ACM Transactions on Graphics* 29.1 (Dec. 2009), 1–12 2.

[MdH10] MORDATCH, IGOR, DE LASA, MARTIN, and HERTZMANN, AARON. "Robust Physics-Based Locomotion Using Low-Dimensional Planning". *ACM Transactions on Graphics* 29.4 (July 2010), 71:1–71:8 3.

[MHG*19] MEREL, JOSH, HASENCLEVER, LEONARD, GALASHOV, ALEXANDRE, et al. "Neural Probabilistic Motor Primitives for Humanoid Control". *arXiv:1811.11711 [Cs]*. New Orleans, Louisiana, USA, Jan. 2019. arXiv: 1811.11711 [cs] 3.

[MLPP09] MUICO, ULDARICO, LEE, YONGJOON, POPOVIĆ, JOVAN, and POPOVIĆ, ZORAN. "Contact-Aware Nonlinear Control of Dynamic Characters". *ACM Transactions on Graphics* 28.3 (July 2009), 81:1–81:9 3.

[MTA*20] MEREL, JOSH, TUNYASUVUNAKOOL, SARAN, AHUJA, ARUN, et al. "Catch & Carry: Reusable Neural Controllers for Vision-Guided Whole-Body Tasks". *ACM Transactions on Graphics* 39.4 (July 2020), 39:39:1–39:39:12 3.

[MTT*17] MEREL, JOSH, TASSA, YUVAL, TB, DHRUVA, et al. "Learning Human Behaviors from Motion Capture by Adversarial Imitation". *arXiv:1707.02201 [cs]* (July 2017). arXiv: 1707.02201 [cs] 3.

[Opt] OPTITRACK. *OptiTrack - Motion Capture Systems*. https://www.optitrack.com/ 7.

[PALv18] PENG, XUE BIN, ABBEEL, PIETER, LEVINE, SERGEY, and VAN DE PANNE, MICHIEL. "DeepMimic: Example-Guided Deep Reinforcement Learning of Physics-Based Character Skills". *ACM Transactions on Graphics* 37.4 (July 2018), 143:1–143:14 3, 7.

[PBYv17] PENG, XUE BIN, BERSETH, GLEN, YIN, KANGKANG, and VAN DE PANNE, MICHIEL. "DeepLoco: Dynamic Locomotion Skills Using Hierarchical Deep Reinforcement Learning". *ACM Transactions on Graphics* 36.4 (July 2017), 41:1–41:13 3.

[PCZ*19] PENG, XUE BIN, CHANG, MICHAEL, ZHANG, GRACE, et al. "MCP: Learning Composable Hierarchical Control with Multiplicative Compositional Policies". *CoRR* abs/1905.09808 (2019). arXiv: 1905.09808 3.

[PRL*19] PARK, SOOHWAN, RYU, HOSEOK, LEE, SEYOUNG, et al. "Learning Predict-and-Simulate Policies from Unorganized Human Motion Data". *ACM Transactions on Graphics* 38.6 (Nov. 2019), 1–11 3, 6.

[RTI*14] RHODIN, HELGE, TOMPKIN, JAMES, IN KIM, KWANG, et al. "Interactive Motion Mapping for Real-Time Character Control". *Computer Graphics Forum* 33.2 (2014), 273–282 2.

[SGXT20] SHIMADA, SOSHI, GOLYANIK, VLADISLAV, XU, WEIPENG, and THEOBALT, CHRISTIAN. "PhysCap: Physically Plausible Monocular 3D Motion Capture in Real Time". *ACM Transactions on Graphics* 39.6 (Nov. 2020), 1–16. arXiv: 2008.08880 3.

[SHP04] SAFONOVA, ALLA, HODGINS, JESSICA K., and POLLARD, NANCY S. "Synthesizing Physically Realistic Human Motion in Low-Dimensional, Behavior-Specific Spaces". *ACM Transactions on Graphics* 23.3 (Aug. 2004), 514–521 2.

[SOL13] SEOL, YEONGHO, O'SULLIVAN, CAROL, and LEE, JEHEE. "Creature Features: Online Motion Puppetry for Non-Human Characters". *Proceedings of the 12th ACM SIGGRAPH/Eurographics Symposium on Computer Animation*. SCA '13. New York, NY, USA: Association for Computing Machinery, July 2013, 213–221 2.

[SWD*17] SCHULMAN, JOHN, WOLSKI, FILIP, DHARIWAL, PRAFULLA, et al. "Proximal Policy Optimization Algorithms". *arXiv:1707.06347 [cs]* (July 2017). arXiv: 1707.06347 [cs] 6.

[SZKZ20] STARKE, SEBASTIAN, ZHAO, YIWEI, KOMURA, TAKU, and ZAMAN, KAZI. "Local Motion Phases for Learning Multi-Contact Character Movements". *ACM Transactions on Graphics* 39.4 (July 2020), 54:54:1–54:54:13 2, 7, 9.

[TGLT14] TAN, JIE, GU, YUTING, LIU, C. KAREN, and TURK, GREG. "Learning Bicycle Stunts". *ACM Transactions on Graphics* 33.4 (July 2014), 50:1–50:12 3.

[TLP07] TREUILLE, ADRIEN, LEE, YONGJOON, and POPOVIĆ, ZORAN. "Near-Optimal Character Animation with Continuous Control". *ACM SIGGRAPH 2007 Papers*. SIGGRAPH '07. San Diego, California: Association for Computing Machinery, July 2007, 7–es 2.

[TLT11] TAN, JIE, LIU, KAREN, and TURK, GREG. "Stable Proportional-Derivative Controllers". *IEEE Computer Graphics and Applications* 31.4 (July 2011), 34–44 7.

[Vic] VICON. *Vicon | Award Winning Motion Capture Systems*. https://www.vicon.com/ 2.

[vRBP17] VON MARCARD, T., ROSENHAHN, B., BLACK, M. J., and PONS-MOLL, G. "Sparse Inertial Poser: Automatic 3D Human Pose Estimation from Sparse IMUs". *Computer Graphics Forum* 36.2 (May 2017), 349–360 2.

[WAH*10] WAMPLER, KEVIN, ANDERSEN, ERIK, HERBST, EVAN, et al. "Character Animation in Two-Player Adversarial Games". *ACM Transactions on Graphics* 29.3 (July 2010), 26:1–26:13 2.

[WCX21] WANG, ZHIYONG, CHAI, JINXIANG, and XIA, SHIHONG. "Combining Recurrent Neural Networks and Adversarial Training for Human Motion Synthesis and Control". *IEEE Transactions on Visualization and Computer Graphics* 27.1 (Jan. 2021), 14–28 2.

[WFH08] WANG, JACK M., FLEET, DAVID J., and HERTZMANN, AARON. "Gaussian Process Dynamical Models for Human Motion". *IEEE Transactions on Pattern Analysis and Machine Intelligence* 30.2 (Feb. 2008), 283–298 2.

[WGH20] WON, JUNGDAM, GOPINATH, DEEPAK, and HODGINS, JESSICA. "A Scalable Approach to Control Diverse Behaviors for Physically Simulated Characters". *ACM Transactions on Graphics* 39.4 (July 2020), 33:33:1–33:33:12 3.

[WGR*19] WOUDA, FRANK J., GIUBERTI, MATTEO, RUDIGKEIT, NINA, et al. "Time Coherent Full-Body Poses Estimated Using Only Five Inertial Sensors: Deep versus Shallow Learning". *Sensors* 19.17 (Jan. 2019), 3716 2.

[WHDK12] WANG, JACK M., HAMNER, SAMUEL R., DELP, SCOTT L., and KOLTUN, VLADLEN. "Optimizing Locomotion Controllers Using Biologically-Based Actuators and Objectives". *ACM Transactions on Graphics* 31.4 (July 2012), 25:1–25:11 3.

[Xse] XSENS. *Xsens 3D Motion Tracking*. https://www.xsens.com/ 2.

[XWI*21] XIE, KEVIN, WANG, TINGWU, IQBAL, UMAR, et al. "Physics-Based Human Motion Estimation and Synthesis From Videos". *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2021, 11532–11541 3.

[YKL21] YANG, DONGSEOK, KIM, DOYEON, and LEE, SUNG-HEE. "LoBSTr: Real-time Lower-body Pose Prediction from Sparse Upper-body Tracking Signals". *Computer Graphics Forum* 40.2 (2021), 265–275 2.

[YLv07] YIN, KANGKANG, LOKEN, KEVIN, and VAN DE PANNE, MICHIEL. "SIMBICON: Simple Biped Locomotion Control". *ACM Transactions on Graphics* 26.3 (July 2007), 105–es 3.

[YPL21] YU, RI, PARK, HWANGPIL, and LEE, JEHEE. "Human Dynamics from Monocular Video with Dynamic Camera Movements". *ACM Trans. Graph.* 40.6 (Dec. 2021) 3.

[ZSKS18] ZHANG, HE, STARKE, SEBASTIAN, KOMURA, TAKU, and SAITO, JUN. "Mode-Adaptive Neural Networks for Quadruped Motion Control". *ACM Transactions on Graphics* 37.4 (Aug. 2018), 1–11 2, 7.