Real-Time Simulated Avatar from Head-Mounted Sensors

Zhengyi Luo[1,2] Jinkun Cao[2] Rawal Khirodkar[1] Alexander Winkler[1] Jing Huang[1]
Kris Kitani[1,2,*] Weipeng Xu[1,*]
[1]Reality Labs Research, Meta; [2]Carnegie Mellon University

https://zhengyiluo.github.io/SimXR/

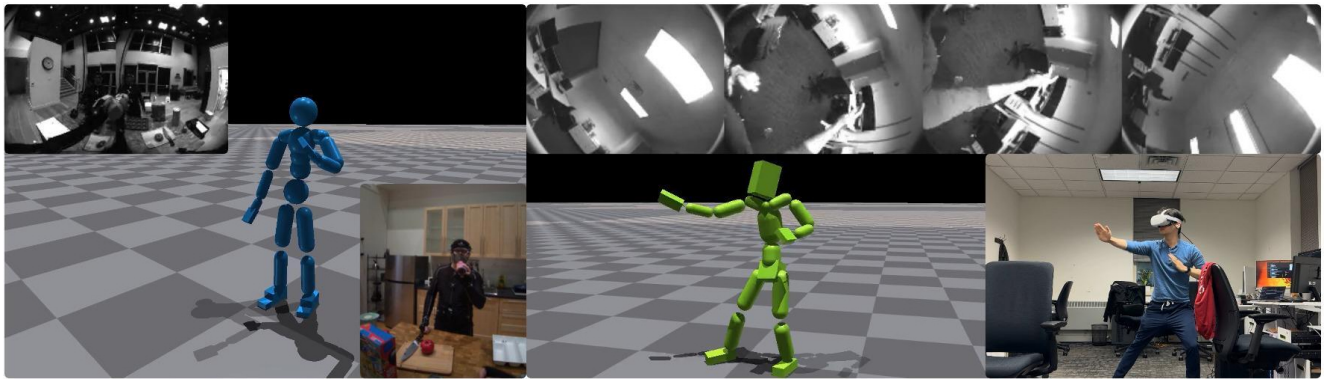

Figure 1. Avatar control using `SimXR` on real world AR/VR headsets. (*Left*): An indoor kitchen setting using AR headset. `SimXR` controls the humanoid using headset pose and visual input from two front-facing cameras. (*Right*): An office setting using VR headset (Quest 2). Humanoid motion is driven by the headset pose, two side-facing and two up-facing cameras.

Abstract

*We present **SimXR**, a method for controlling a simulated avatar from information (headset pose and cameras) obtained from AR / VR headsets. Due to the challenging viewpoint of head-mounted cameras, the human body is often clipped out of view, making traditional image-based egocentric pose estimation challenging. On the other hand, headset poses provide valuable information about overall body motion, but lack fine-grained details about the hands and feet. To synergize headset poses with cameras, we control a humanoid to track headset movement while analyzing input images to decide body movement. When body parts are seen, the movements of hands and feet will be guided by the images; when unseen, the laws of physics guide the controller to generate plausible motion. We design an end-to-end method that does not rely on any intermediate representations and learns to directly map from images and headset poses to humanoid control signals. To train our method, we also propose a large-scale synthetic dataset created using camera configurations compatible with a commercially available VR headset (Quest 2) and show promising results on real-world captures. To demonstrate the applicability of our framework, we also test it on an AR headset with a forward-facing camera.*

1. Introduction

From the sensor streams captured by a head-mounted device (AR / VR, or XR headsets), we aim to control a simulated humanoid / avatar to follow the wearer's global 3D body pose in real-time, as shown in Fig.1. This could be applied to animating virtual avatars in mixed reality, games, and potentially teleoperating humanoid robots [21]. However, the sensor suite of commercially available head-mounted devices is rarely designed for full-body pose estimation. Their cameras are often facing forward (*e.g.* Aria glasses [52]) or on the side (*e.g.* Meta Quest [4]) and are used mainly for Simultaneous Location and Mapping (SLAM) and hand tracking. Thus, the body is seen from extreme and distorted viewpoints from these cameras.

These challenges have led to research on vision-based egocentric pose estimation to create scenarios with more favorable camera placement (*e.g.* fisheye cameras directly pointing downward [5, 54–56, 64, 69]), where more body parts can be observed. These camera views are often unrealistic and hard to recreate in the real-world: a camera protruding out and facing downward could be out of the budget or break the aesthetics of the product. As there is no standard for these heterogeneous camera specifications, it is difficult to collect large-scale data. Using synthetic data [5, 54]

*Equal advising.

could alleviate this problem to some extent, but real-world recreation of the camera specification used in rendering is still challenging, exacerbating the sim-to-real gap.

Another line of work instead uses head tracking to infer full-body motion. The 6 degree-of-freedom (DoF) headset pose, being considerably lower in dimensionality compared to images, is easily accessible from extended reality (XR) headsets. However, the headset pose alone contains insufficient information on full body movement, so previous work either formulates the task as a generative one [28] or relies on action labels [30] to further constrain the solution space. Adding VR controllers as input can provide additional information on hand movement and can lead to a more stable estimate of full body pose, but the VR controller is not always available [3, 8, 41], especially for light-weight AR glasses with forward-facing cameras. These methods are also primarily kinematics-based [7, 8, 14, 25], focusing on motion estimation without taking into account the underlying forces. As a result, they cause floating and penetration problems, especially because feet are often unobserved.

To combat these issues, physics simulation [59] and environmental cues [27] have been used to create plausible foot motion. Leveraging the laws of physics can significantly improve motion realism and force the simulated character to adopt a viable foot movement. However, incorporating physics introduces the additional challenge of humanoid control–humanoids need to be balanced and track user movement at all times. Most physics-based methods are learned using reinforcement learning (RL) and require millions (sometimes billions) of environment interactions. If each interaction requires processing of the raw image input, the computational cost would be significant. As a result, approaches that use vision and simulated avatar often use a low-dimensional intermediate representation, such as pre-computed image features [66] or kinematic poses [30, 31, 68]. Although a controller can be trained to consume these intermediate features, this approach creates a disconnect between the visual and control components, where the visual component does not receive adequate feedback during training from physics simulation.

In this work, we demonstrate the feasibility of an end-to-end simulated avatar control framework for XR headsets. Our approach, ***SIM**ulated Avatar from **XR** sensors* (`SimXR`), directly maps the input signals to joint actuation without relying on intermediate representations such as body pose or 2D keypoints. Our key design choice is to distill from a pre-trained motion imitator to learn the mapping from input to control signals, which enables efficient learning from vision input. Due to its simplistic design and learning framework, our approach is compatible with a diverse selection of smart headsets, ranging from VR goggles to lightweight AR headsets (as shown in Fig.2). Since no dataset exists for the camera configurations of commercially available VR headsets, we also propose a large-scale synthetic dataset (2216k frames) and a real-world dataset (40k frames) for testing and show that our method can be applied to real-world and real-time use cases.

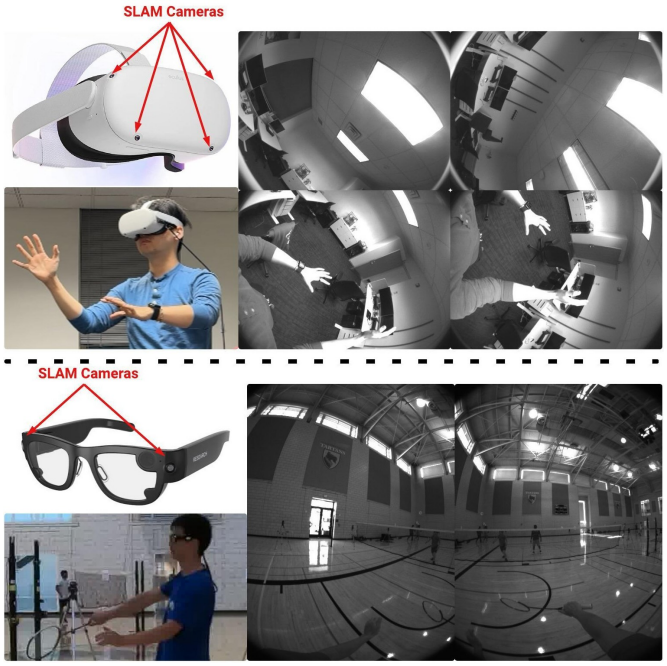

Figure 2. SimXR framework applied to two AR/VR devices. (*Top*): Quest 2 [2] headset with 4 SLAM cameras, two facing upward and two downward. (*Bottom*): Aria glass [1] with two forward-facing SLAM cameras. Both devices provides 6DoF headset tracking in real-time.

To summarize, our contributions are: (1) we design a method to use simulated humanoids to estimate global full-body pose using images and headset pose from an XR headset (front-facing AR cameras or side-facing VR ones); (2) we demonstrate the feasibility of learning an end-to-end controller to directly map from input sensor features to control signals through distillation; (3) we contribute large-scale synthetic and real-world datasets with commercially available VR headset configuration for future research.

2. Related Work

Pose Estimation from Head-Mounted Sensors. Due to the lack of dataset using commercially available devices, egocentric pose estimation has been studied using synthetic data [5, 54], and small-scale real-world data captured from custom camera rigs [24, 44, 55, 64, 65]. Among them, EgoCap[44] uses two downward facing cameras protruding from the helmet, while Jiang *et al*. [24] uses a chest-mounted camera. Later, Mo2Cap2 [64] and SelfPose [54] use a head-mounted downward-facing fisheye camera for pose estimation in the camera's coordinate system. All use synthetic data for training, while Mo2Cap2 also captures

a real-world dataset (10k frames) for testing. Wang *et al*. [55, 56] extends a similar setup for global 3D body pose estimation by extracting head movement from video and optimization-based global pose refinement. To combat the lack of large-scale dataset, UnrealEgo [5] uses a dual fisheye camera setup to generate synthetic data using a game engine. All of the above settings have cameras that directly point downward at the human body. However, cameras on commercially available XR headsets often do not have a dedicated body camera and only have monochrome SLAM or hand-tracking cameras. For VR devices, these cameras point to the side and have very limited visibility of the hands and feet. For AR glasses, cameras often point forward and provide only fleeting hand visibility. Due to challenging viewpoints, some work uses head tracking as an alternative [28, 30, 41, 59, 59, 65, 66] for pose estimation. Among them, Egopose [66] estimates locomotion, while KinPoly [30] extends it to action-conditioned pose estimation. EgoEgo [28] proposes the first general-purpose pose estimator that uses only the head pose. While they utilize front-facing images, the images are used to extract the head pose, rather than to provide information about the body pose. In this work, we take advantage of both camera views and headset pose for full-body avatar control.

Simulated Humanoid Motion Imitation. Motion imitation is an important humanoid control task that has seen steady progress in recent years [6, 12, 16, 33, 39, 40, 58, 60, 67]. Since no ground-truth data exist of human joint actuation and physics simulators are often nondifferentiable, a policy / imitator / controller is often trained to track / imitate / mimic human motion using deep reinforcement learning (RL). Although methods such as SuperTrack [16] and DiffMimic [43] explore more efficient ways than RL to train imitators, learning a robust policy to track a large amount of human motion remains challenging. Nonetheless, from policies that can track a single clip of motion [39], to large-scale datasets [33, 60], the applicability of motion imitators to downstream tasks grows. Previously, UHC [30], a motion imitator based on an external non-physical force [67], has been used for egocentric [30] and third-person scene-aware [31] pose estimation. Its follow-up, PHC [33], removes the dependency on the non-physical force.

Simulated Humanoid Control for Pose Estimation. Our work follows recent advances [18, 19, 22, 31, 57, 68] in using the laws of physics as a strong prior to estimating full-body motion. Mapping directly from images to humanoid control signals is quite challenging due to the complex dynamics of a humanoid and the diversity in natural images. Training motion controllers using RL is also notoriously sample inefficient, forcing some vision-based robotic approaches to use distillation [71] or very small images [36]. Therefore, most physics-based methods separate the problem into two distinct components: image-based pose estimation and humanoid motion imitation. First, the pose is estimated from the images using an off-the-shelf body pose [26] or a keypoint detector [17]. The estimated pose is then fed to a pre-trained imitator for further refinement [31, 68], sampling-based control [22], or co-training [18]. Some methods also employ trajectory optimization [42, 50, 51, 63], but the optimization process can be time-consuming, unless certain compromises are assumed (*e.g*. applying external non-physical forces [50, 51]). This disjoint process uses the kinematic body pose as an intermediate representation to communicate movement information to the humanoid controller. However, this communication layer can be fragile and adversely affected by the imitator's sensitivity to the intermediate representation. Thus, we propose to remove the kinematic pose layer and directly learn a mapping from the input to the control signals end-to-end.

3. Approach

At each time step, given the images $\boldsymbol{I}_t$ and the 6DoF pose ${\boldsymbol{q}_t}^{\intercal}$ captured by the headset, our task is to drive a simulated avatar to match the full body pose of the camera wearer $\boldsymbol{q}_t$. We use monochrome SLAM cameras on XR headsets, producing images of dimension $\boldsymbol{I}_t \in \mathbb{R}^{V \times H \times W \times C}$ of V views (2 views for Aria glasses, 4 for Quest) and $C = 1$ channels for monochrome images. Body pose $\boldsymbol{q}_t \triangleq (\boldsymbol{\theta}_t, \boldsymbol{p}_t)$ consists of 3D joint rotation $\boldsymbol{\theta}_t \in \mathbb{R}^{J \times 6}$ (using the 6D rotation representation [70]) and position $\boldsymbol{p}_t \in \mathbb{R}^{J \times 3}$ of all J links on the articulated humanoid. Body velocities $\dot{\boldsymbol{q}}_{1:T}$ are expressed as $\dot{\boldsymbol{q}}_t \triangleq (\boldsymbol{\omega}_t, \boldsymbol{v}_t)$, consisting of angular $\boldsymbol{\omega}_t \in \mathbb{R}^{J \times 3}$ and linear velocities $\boldsymbol{v}_t \in \mathbb{R}^{J \times 3}$. Throughout the paper, we use $\hat{\cdot}$ to denote the ground-truth kinematic motion from Motion Capture (MoCap) and normal symbols without accents for values from the physics simulation. In Sec.3.1, we will first set up the preliminaries for humanoid control. Then, in Sec. 3.2, we briefly describe our synthetic data generation pipeline. In Sec. 3.3, we describe our proposed method `SimXR` and how to learn this controller.

3.1. Preliminaries: Simulated Humanoid Control

Instead of directly regressing full-body pose, we use a simulated humanoid to "act" inside physics simulation and read out its states as pose estimates. Following the general framework of goal-conditioned RL, we aim to obtain a goal-conditioned policy π to control humanoids based on input from the headset. The framework is formulated as a Markov Decision Process (MDP) defined by the tuple $\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R}, \gamma \rangle$ of states, actions, transition dynamics, reward function, and discount factor. Physics simulation determines the state $\boldsymbol{s}_t \in \mathcal{S}$ and the dynamics of the transition $\mathcal{T}$, where a policy computes the action $\boldsymbol{a}_t$. For humanoid control tasks, the state $\boldsymbol{s}_t$ contains the proprioception $\boldsymbol{s}_t^{\text{p}}$ and the goal state $\boldsymbol{s}_t^{\text{g}}$. Proprioception is defined as $\boldsymbol{s}_t^{\text{p}} \triangleq (\boldsymbol{q}_t, \dot{\boldsymbol{q}}_t)$, which contains the 3D body pose $\boldsymbol{q}_t$ and

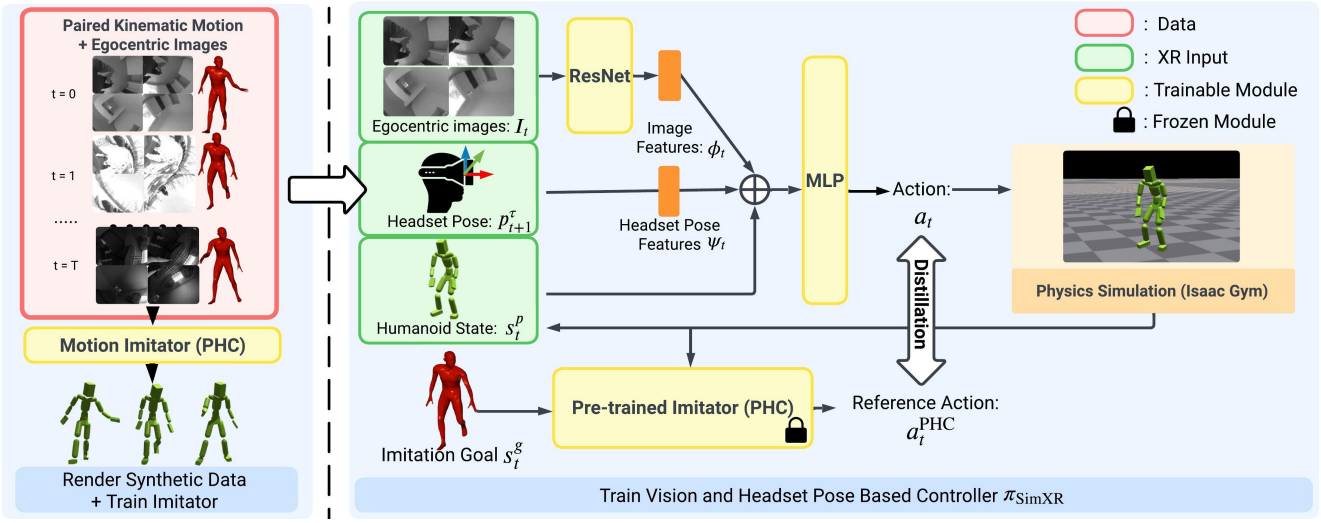

Figure 3. Our proposed SimXR framework. From a large-scale human motion dataset, we first train a motion imitator (PHC [33]) and render synthetic images. Then, we train our vision and headset pose-based controller through distilling from the pretrained imitator.

the velocity $\dot{\boldsymbol{q}}_t$. The goal state $\boldsymbol{s}^{\text{g}}_t$ is defined based on the task. Based on proprioception $\boldsymbol{s}^{\text{p}}_t$ and the goal state $\boldsymbol{s}^{\text{g}}_t$, the reward $r_t = \mathcal{R}(\boldsymbol{s}^{\text{p}}_t, \boldsymbol{s}^{\text{g}}_t)$ is used to train the policy using RL (*e.g.* PPO [49]). If a reference action $\hat{\boldsymbol{a}}_t$ can be provided (often by a pre-trained expert), we can also optimize the policy π through policy distillation [45, 46, 48].

Physics-based Motion Imitation. Motion imitation is defined as the task of controlling a simulated humanoid to match a sequence of kinematic human motion $\hat{\boldsymbol{q}}_{1:T}$. A motion imitator is formulated as follows: given the next-frame reference 3D pose $\hat{\boldsymbol{q}}_{t+1}$, a policy $\pi_{\text{PHC}}(\boldsymbol{a}_t|\boldsymbol{s}^{\text{p}}_t, \boldsymbol{s}^{\text{g-mimic}}_t)$ computes the joint actuation $\boldsymbol{a}_t$ to drive the humanoid to match $\hat{\boldsymbol{q}}_{t+1}$. The goal state for motion imitation is defined as $\boldsymbol{s}^{\text{g-mimic}}_t \triangleq (\hat{\boldsymbol{\theta}}_{t+1} \ominus \boldsymbol{\theta}_t, \hat{\boldsymbol{p}}_{t+1} - \boldsymbol{p}_t, \hat{\boldsymbol{v}}_{t+1} - \boldsymbol{v}_t, \hat{\boldsymbol{\omega}}_t - \boldsymbol{\omega}_t, \hat{\boldsymbol{\theta}}_{t+1}, \hat{\boldsymbol{p}}_{t+1})$, which contains the one frame difference between the reference and the current pose, normalized with respect to the heading of the current humanoid. A trained motion imitator could be used as a teacher and distill its motor skills for downstream tasks [11, 32, 36, 61], and we follow this paradigm and use an off-the-shelf RL-trained motion imitator (PHC [33]) as a teacher to learn the mapping between the XR headset sensor and the control signals.

Avatar Control using head-mounted Sensors. Following the above definition, the task of controlling a simulated humanoid to match egocentric observation from head-mounted sensors can be formulated as using a control policy $\pi_{\text{SimXR}}(\boldsymbol{a}_t|\boldsymbol{s}^{\text{p}}_t, \boldsymbol{I}_t, \boldsymbol{q}^{\tau}_{t+1})$ to compute the joint action $\boldsymbol{a}_t$ based on image $\boldsymbol{I}_t$, headset pose $\boldsymbol{q}^{\tau}_{t+1}$, and humanoid proprioception $\boldsymbol{s}^{\text{p}}_t$ to match the headset wearer's body pose $\hat{\boldsymbol{q}}_{1:T}$.

3.2. Synthetic Data for XR Avatar

As paired motion data and XR cameras & headset tracking is difficult to obtain, we use synthetic data to train our method when real data is not available. Specifically, since no data exist for Quest 2's SLAM camera configuration, we create a large-scale synthetic one. We render human motion from a large-scale internal MoCap dataset using the exact camera placement and intrinsic of the Quest 2 headset in the Unity game engine [9]. The headset moves with the head movement of the wearer as if it is worn. Our motion dataset contains diverse poses ranging from daily activities to jogging, stretching, gesturing, sports, *etc.*, similar to AMASS [34] but uses a different kinematic structure from SMPL [29]. During rendering, we randomize clothing, lighting, and background images (projected onto a sphere) for **every** frame for domain randomization. As a result, the methods trained using our synthetic data can be applied to real-world scenarios without additional image-based domain augmentation during training. We render RGB images and then convert them to monochrome. Fig. 3 shows samples of our synthetic data. For more information on our synthetic data, please refer to the Supplement.

3.3. Simulated Humanoid Control From Head-Mounted Sensors

Sensor Input Processing. At each frame, the head-mounted sensors provide the image $\boldsymbol{I}_{t+1}$ and the 6DoF headset pose $\boldsymbol{p}^{\tau}_{t+1}$ (here we use t+1 to indicate the incoming pose/image for tracking). The input image $\boldsymbol{I}_{t+1}$ (which contains all monochrome views) is first processed with a lightweight image feature extractor $\mathcal{F}$ (*e.g.* ResNet18 [20]) to compute image features: $\boldsymbol{\phi}_t = \mathcal{F}(\boldsymbol{I}_{t+1})$. All V camera views share the same feature extractor (Siamese network). We also replace all batch normalization layers [23] with group normalization [62] for training stability.

For the 6DoF headset pose $\boldsymbol{q}_t{}^{\tau}$, we treat it as a virtual "joint", where $\boldsymbol{q}^{\tau}_{t+1} \triangleq (\boldsymbol{p}^{\tau}_{t+1}, \boldsymbol{\theta}^{\tau}_{t+1})$ contains the global rotation $\boldsymbol{\theta}^{\tau}_{t+1}$ and translation $\boldsymbol{p}^{\tau}_{t+1}$ of the headset. We use the

humanoid head to track the pose of the headset by computing the rotation and translation difference between the head joint $\boldsymbol{q}_t^{\text{Head}}$ and the headset pose $\boldsymbol{q}_{t+1}^{\tau}$. This creates the headset pose feature: $\psi_t = (\boldsymbol{\theta}_t^{\text{Head}} \ominus \boldsymbol{\theta}_{t+1}^{\tau}, \boldsymbol{p}_t^{\text{Head}} - \boldsymbol{p}_{t+1}^{\tau})$. The image feature and the headset pose features are then concatenated with proprioception $\boldsymbol{s}_t^{\text{p}}$ to form the input to an MLP to compute joint actions, as illustrated in Fig.3.

Online Distillation. Learning to control humanoids using RL requires a large number of simulation steps (*e.g.* billions of environment interactions), and our early experiments show that directly training an image-based policy using RL is infeasible (see Sec. 4.2). This is due to the large increase in input and model size (*e.g.* four 120×160 monochrome images versus 938d imitation goal $\boldsymbol{s}_t^{\text{g-mimic}}$) that prohibitively slows down training steps. Therefore, we opt for online distillation and use a pre-trained motion imitator, PHC π_{PHC}, to teach our policy $\pi_{\texttt{SimXR}}$ via supervised learning. In short, we offload the sample-inefficient RL training to a low-dimensional state task (motion imitation) and use sample-efficient supervised learning for the high-dimensional image processing task. This is similar to the process proposed in PULSE [32] with the important distinction that PULSE uses the same input and output for the student and teacher, while ours has drastically different input modalities (images and headset pose vs. 3D pose).

Concretely, we train our pose estimation policy $\pi_{\texttt{SimXR}}$ following the standard RL training framework: for each episode, given paired egocentric images $\boldsymbol{I}_{1:T}$, headset pose $\boldsymbol{q}_{1:T}^{\tau}$, and the corresponding reference full-body pose $\hat{\boldsymbol{q}}_{1:T}$, the humanoid is first initialized as $\hat{\boldsymbol{q}}_0$. Then, the policy $\pi_{\texttt{SimXR}}(\boldsymbol{a}_t | \boldsymbol{s}_t^{\text{p}}, \boldsymbol{I}_t, \boldsymbol{q}_{t+1}^{\tau})$ computes the joint actuation for the forward dynamics computed by physics simulation. By rolling out the policy in simulation, we obtain paired trajectories of $(\boldsymbol{s}_{1:T}^{\text{p}}, \boldsymbol{I}_{1:T}, \boldsymbol{q}_{1:T}^{\tau}, \hat{\boldsymbol{q}}_{1:T})$. Using the ground truth reference pose $\hat{\boldsymbol{q}}_{1:T}$ and simulated humanoid states $\boldsymbol{s}_{1:T}^{\text{p}}$, we can compute the per-frame imitation target for our pre-trained imitator $\boldsymbol{s}_t^{\text{g-mimic}} \leftarrow (\boldsymbol{s}_t^{\text{p}}, \hat{\boldsymbol{q}}_{t+1})$. Then, using paired $(\boldsymbol{s}_t^{\text{p}}, \boldsymbol{s}_t^{\text{g-mimic}})$, we query PHC $\pi_{\text{PHC}}(\boldsymbol{a}_t^{\text{PHC}} | \boldsymbol{s}_t^{\text{p}}, \boldsymbol{s}_t^{\text{g-mimic}})$ to calculate the reference action $\boldsymbol{a}_t^{\text{PHC}}$. This is similar to DAgger [45], where we use an expert to annotate reference actions for the student to learn from. To update $\pi_{\texttt{SimXR}}$, the loss is:

$$\mathcal{L} = \|\boldsymbol{a}_t^{\text{PHC}} - \boldsymbol{a}_t\|_2^2, \tag{1}$$

using standard supervised learning. In this way, our policy is trained end-to-end, where the image feature extractors $\mathcal{F}$ and the policy networks are directly updated using the gradient of $\mathcal{L}$. The learning process is also described in Alg. 1.

Algo 1: Learn `SimXR` via distillation

Input: Ground truth paired XR sensor input and pose dataset $\hat{\boldsymbol{Q}}$, pretrained PHC π_{PHC} ;
while *not converged* **do**
  $\boldsymbol{M} \leftarrow \emptyset$ initialize sampling memory ;
  while $\boldsymbol{M}$ *not full* **do**
    $\hat{\boldsymbol{q}}_{1:T}, \boldsymbol{I}_{1:T}, \boldsymbol{s}_t^{\text{p}}, \leftarrow$ sample motion, images and initial state from $\hat{\boldsymbol{Q}}$;
    for $t \leftarrow 1 \ldots T$ **do**
      $\boldsymbol{a}_t \leftarrow \pi_{\texttt{SimXR}}(\boldsymbol{a}_t | \boldsymbol{s}_t^{\text{p}}, \boldsymbol{I}_t, \boldsymbol{q}_{t+1}^{\tau})$; `// compute humanoid action;`
      $\boldsymbol{s}_{t+1} \leftarrow \mathcal{T}(\boldsymbol{s}_{t+1} | \boldsymbol{s}_t, \boldsymbol{a}_t)$ `// simulation;`
      $\boldsymbol{s}_t^{\text{g-mimic}} \leftarrow (\boldsymbol{s}_t^{\text{p}}, \hat{\boldsymbol{q}}_{t+1})$; `// compute imitation target for PHC;`
      store $\boldsymbol{s}_t^{\text{p}}, \boldsymbol{s}_t^{\text{g-mimic}}, \boldsymbol{I}_t, \boldsymbol{q}_{t+1}^{\tau}$ into memory $\boldsymbol{M}$;
  $\boldsymbol{a}_t^{\text{PHC}} \leftarrow \pi_{\text{PHC}}(\boldsymbol{a}_t^{\text{PHC+}} | \boldsymbol{s}_t^{\text{p}}, \boldsymbol{s}_t^{\text{g-mimic}})$ annotate collected states in $\boldsymbol{M}$ using π_{PHC} ;
  $\pi_{\texttt{SimXR}} \leftarrow$ supervised update for $\pi_{\texttt{SimXR}}$ using pairs of $(\boldsymbol{a}_t, \boldsymbol{a}_t^{\text{PHC}})$ and Eq.1.

Aria Glasses			Quest 2				
Aria Digital Twin [37]			Synthetic			Real-world	
Train	Test	Annot.	Train	Test	Annot.	Test	Annot.
242 / 94k	64 /25k	MoCap	4097 / 1758k	1072/458k	MoCap	10/40k	Mono

Table 1. Dataset statistics and annotation source. MoCap: motion capture; Mono: monocular third-person pose estimation.

4. Experiments

Humanoids. Due to the different annotation formats between our proposed synthetic dataset and public datasets, we use two different humanoids for our experiments, one for the Quest VR headset and one for Aria AR glasses. For Aria, we use a humanoid following the SMPL [29] kinematic structure with the mean shape. It has 24 joints, of which 23 are actuated, resulting in an actuation space of $\mathbb{R}^{23\times3}$. Each degree of freedom is actuated by a proportional derivative (PD) controller, and the action $\boldsymbol{a}_t$ specifies the PD target. For the VR datasets, we use a humanoid that has 25 joints (out of which 24 are actuated). The imitator for SMPL-humanoid is trained on the AMASS [34] dataset, while for the in-house humanoid it is trained on the same in-house motion capture used to create our synthetic dataset.

Datasets. To train our control policies, we require high-quality motion data paired with camera views, as training with low-quality data might lead to the simulated character picking up unwanted behavior. Thus, for pose estimation using the Quest headset, we train solely on synthetic data created using a large-scale in-house motion capture dataset. We randomly split the data using an 8:2 ratio, resulting in 1758k frames for training and 458k frames for testing. To test the performance of our method in real-world scenarios, we also collect a real-world dataset containing 40k frames recorded by three different subjects. This dataset contains paired headset poses, SLAM camera images, and third-person images. We use the third-person images to create pseudo-ground truth using a SOTA monocular pose estimation method [53]. The real-world dataset contains daily activity motion common in VR scenarios, such as hand

Figure 4. Qualitative results on synthetic and real-world AR/VR headset data. We visualize camera images, simulation, rendered mesh from simulation states, and third-person reference views. We show that our method can transfer to real-world data and handle diverse body poses including kicking, kneeling, *etc*. For AR headset results, the third-person view is provided by another subject wearing a headset.

			Synthetic-Test						Real-world			
Method	Size	Physics	Succ ↑	$E_{\text{g-mpjpe}}$ ↓	E_{mpjpe} ↓	$E_{\text{pa-mpjpe}}$ ↓	E_{acc} ↓	E_{vel} ↓	Succ ↑	$E_{\text{pa-mpjpe}}$ ↓	E_{acc} ↓	E_{vel} ↓
UnrealEgo [5]	554.5MB	✗	-	-	**56.9**	47.2	54.5	32.5	-	81.0	46.7	35.1
KinPoly-v [30]	98.9MB	✓	82.2%	66.7	63.5	42.8	**4.4**	**5.8**	9/10	83.0	7.0	11.2
Ours	59.7MB	✓	**94.3%**	**66.4**	62.4	**40.0**	6.5	8.3	**10/10**	**73.0**	**6.8**	**10.5**

Table 2. Pose estimation result on the test split (458k frames) of synthetic data and real-world captures (40k frames). Here, our MPJPE is computed as "device-relative" instead of root-relative.

movements, boxing, kicking, *etc*. For pose estimation using Aria glasses, we use the recently proposed Aria Digital Twin dataset (ADT) [37]. The ADT dataset contains indoor motion sequences captured using MoCap suits and 3D scene scanners. It contains 119k frames that have paired skeleton and AR headset sensor output, recorded in a living room environment. This dataset only contains 3D keypoint annotations (no rotation), and we fit the SMPL body annotation to the 3D keypoints using a process similar to 3D-Simplify [10, 38]. We randomly split the ADT dataset for

	ADT-Test					
Method	Succ ↑	$E_{\text{g-mpjpe}}$ ↓	E_{mpjpe} ↓	$E_{\text{pa-mpjpe}}$	E_{acc} ↓	E_{vel} ↓
KinPoly [30]	98.3%	93.8	80.6	60.8	5.9	9.5
UnrealEgo [5]	-	-	131.5	71.5	26.7	20.4
Ours (headset-only)	**100%**	120.7	120.6	85.8	5.2	9.2
Ours	**100%**	**67.8**	**67.6**	**47.7**	**4.6**	**7.3**

Table 3. Pose estimation results on the ADT test set (25k frames).

training (94k frames) and testing (25k frames). We do not use synthetic data for training the AR controller since there are available real-world ground-truth data.

Metrics. We report both pose- and physics-based metrics to evaluate our avatar's performance. We report the success rate (Succ) as in UHC [30], defined as: at *every point* during imitation, the head joint is < 0.5m from the headset pose. For pose estimation, we report the **device-relative** (instead of root relative) per-joint position error (MPJPE) E_{mpjpe}, global MPJPE $E_{\text{g-mpjpe}}$, and MPJPE after Procrustes analysis $E_{\text{pa-mpjpe}}$. Since $E_{\text{pa-mpjpe}}$ solves for the best matching scale, rotation, and translation, it is more suitable for our real-world dataset, where the scale and global position of the pseudo-ground-truth pose are noisy. To maintain the consistency of evaluation between the two humanoids, we select 11 common joints (head, left and right shoulders, elbows, wrists, knees, ankles) to report joint errors, rather than evaluating all joints as in the prior art [33]. To test physical realism, we include acceleration E_{acc} (mm/frame^2) and velocity E_{vel} (mm / frame) errors.

Baselines. We adopt the SOTA vision-based pose estimation method UnrealEgo [5] as the main vision-based baseline. UnrealEgo uses a U-Net structure to first reconstruct 2D heatmaps from input images. Then, an autoencoder is used to lift the 2D heatmap to 3D keypoints. UnrealEgo predicts the 3D pose in the headset's coordinate system instead of the global one. To compare against a physics-based method, we reimplement KinPoly [30] and add image input to create KinPoly-v. We also remove its dependence on action labels and external forces (replacing UHC [30] with PHC [33]). KinPoly-v uses a two-stage method: first estimate full-body pose from images, and then feed them into a pretrained imitator to drive the simulated avatar. It uses a closed-loop system, where the simulated state is fed into the image-based pose estimator, making it "dynamically regulated". KinPoly-v shares the same overall architecture as our method but uses kinematic pose as an intermediate representation to communicate with a pretrained imitator, instead of an end-to-end approach.

Implementation Details. We use NVIDIA's Isaac Gym [35] for physics simulation. All monochrome images are resized to 120×160 for the Aria and Quest headsets during training and evaluation. All MLPs used are 6-layer with units [2048, 1536, 1024, 1024, 512, 512] and SiLU [15] activation. The networks for Quest and Aria share the same architecture, with the only difference being the first layer for the image feature extractor (processing 2 or 4 monochrome images). Due to the orders-of-magnitude increase in input size and network capacity (ResNet 18 vs. 6 layer MLPs), training image-based methods with simulation is around 10 times more resource intensive even using our lightweight networks. We train $\pi_{\texttt{SimXR}}$ for three days, collecting 0.1B environment interactions. For comparison, PHC trained for three days using RL collect around 2B environment interactions. After training, our estimation network and simulation run at $\sim$30 FPS. We perform all the evaluation with a fixed body shape for both humanoids, as our pipeline does not infer or use any body shape information. For more implementation details, please refer to the supplement.

4.1. Results

As motion is best seen in videos, please refer to our supplement for extended visual results of our proposed method.

VR Headset Pose Estimation. In Table 2 and Fig. 4, we report the results of our synthetic and real-world dataset. Our method achieves comparable or better pose estimation results than both prior vision and vision + physics-based methods, estimating physically plausible full-body poses. Compared to UnrealEgo, we can see that `SimXR` uses fewer parameters by an order of magnitude while achieving a better pose estimation result. While both UnrealEgo and `SimXR` are single-frame methods (using no temporal architectures), `SimXR` has significantly less jittering. This is due to the laws-of-physics as a strong prior, and the inherent temporal information provided by the simulation state. Note that UnrealEgo estimates pose in the device frame, so it provides a better headset-relative pose estimate in terms of E_{mpjpe}. Our method directly estimates pose in the global coordinate system and does not benefit from aligning the head position (as can be seen in the small gap between global $E_{\text{g-mpjpe}}$ and local E_{mpjpe} joint errors. Compared to KinPoly-v, we can see that our method achieves better performance across the board. In KinPoly-v, the imitator, UHC [30], uses an external non-physical force to help balance the humanoid and does not require any reference velocity $\hat{\boldsymbol{q}}_t$ as input. PHC does not use any non-physical forces and does require reference velocities as input, which creates additional difficulty in KinPoly-v. As a result, KinPoly-v does not perform well in both synthetic and real-world test sets, showing the limitations of the two-stage methods. Open-loop methods (where the policy does not have access to the kinematic state) may suffer less from the velocity estimation issue but introduce more disjoint between the kinematic and dynamic processes, as shown in KinPoly [30]. The relatively small gap between real-world and synthetic performance shows that our synthetic dataset is effective in providing a starting point for this challenging task.

Method	Synthetic-Test, $E_{\text{pa-mpjpe}} \downarrow$										
	Head	L_Shoulder	L_Elbow	L_Wrist	R_Shoulder	R_Elbow	R_Wrist	L_Knee	L_Ankle	R_Knee	R_Ankle
UnrealEgo	**28.3**	29.3	40.7	55.8	29.2	38.2	46.9	53.0	74.4	50.9	72.8
Ours	30.2	**25.1**	**31.8**	**46.1**	**24.5**	**31.4**	**43.1**	**43.9**	**61.0**	**43.8**	**60.8**

Table 4. Per-joint error analysis on the $E_{\text{pa-mpjpe}}$ on the synthetic test set.

Synthetic-Test							
Vision	Headset	GN	Distill	Succ ↑	$E_{\text{g-mpjpe}} \downarrow$	$E_{\text{acc}} \downarrow$	$E_{\text{vel}} \downarrow$
✗	✓	✓	✓	73.0%	118.9	8.8	11.6
✓	✗	✓	✓	27.1%	161.5	6.8	**8.3**
✓	✓	✗	✓	93.8%	74.9	7.3	9.3
✓	✓	✓	✗	35.8%	226.1	9.6	9.9
✓	✓	✓	✓	**94.3%**	**66.4**	**6.5**	**8.3**

Table 5. Ablations on components of SimXR: without vision/headset input, not suing group norm, and no distillation.

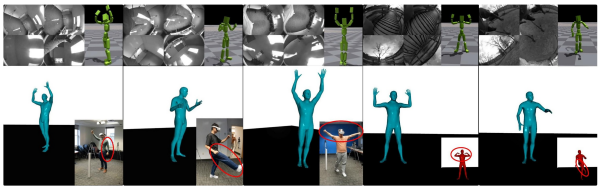

Figure 5. Failure cases of our method: misplaced feet or hands.

AR Headset Pose Estimation. Table 3 shows the test result on the ADT dataset. ADT is a much simpler dataset in terms of motion complexity, as it contains only walking, reaching, and daily activities. However, estimating pose from an AR headset is a harder task, as the viewing angle is much more challenging with the body not seen most of the time. Thus, UnrealEgo performs poorly on the dataset, unable to deal with unseen body parts. Our method can effectively leverage head movement and create plausible full-body motion. Due to similar issues in the VR headset case, KinPoly-v also does not perform well. To show that our method effectively uses the image input, Table 3 also includes a headset-only variant of our approach, where we do not use any vision-based input. Comparing row 3 (R3) and R4, we can see that vision-based input indeed provides valuable information about the movement of the hands.

4.2. Ablations and Analysis

In Table 5, we ablate the components of our method on the synthetic test set. Comparing R1, R2 and R5, we can see the importance of each of the two modalities: the vision signals provide most of the end-effector body movement signals, while the headset guides the body root motion. Without vision signals, the humanoid would achieve poor pose estimation results but can still achieve a reasonable success rate since the headset pose provides a decent amount of movement signals (R1). Without headset pose signals (R2), the humanoid will soon lose track of the head pose. In this case, the method needs to perform both SLAM and pose estimation from images, which requires special architectures. Comparing R3 and R5, we can see that the use of group normalization instead of batch normalization provides some boost in performance. R4 shows that training from scratch using RL for vision-based methods without using distillation would require more efficient algorithms.

To further analyze our pose estimation results, we report the per-joint values for $E_{\text{pa-mpjpe}}$ in Table 4. We can see that SimXR, similar to UnrealEgo, makes the most mistakes in end effectors such as the wrists and ankles. It performs worse in head alignment since it uses a humanoid to track the headset pose, but gains performance on lower-body joints such as ankles. This is due to the fact that SimXR effective uses physics as a prior and can create plausible lower body movement based on input signals.

5. Conclusion and Future Work

Failure Cases. In Fig. 5, we visualize some failure cases of our method. As one of the first methods to control a simulated humanoid to match image observations from commercial headsets, it can misinterpret hand positions when they are not observed. Some kicking motion can also be ignored if the feet observation is blurry (due to clothing color, lighting *etc*.). We also notice that, in some cases, the simulated humanoid movement might lag behind the real-world images, especially when hands come in and out of view. This can be attributed to the humanoid being conservative and not committing to movement until the body part is fully visible. In the videos, we can observe the humanoid stumbling and dragging its feet to remain balanced, especially when the headset moves too quickly.

Conclusions. We propose SimXR to control simulated avatars to match sensor input from commercially available XR headsets. We propose a simple, yet effective, end-to-end learning framework to learn to map from headset pose and camera input from XR headsets to humanoid control signals via distillation. To train our method and facilitate future research, we also propose a large-scale synthetic dataset for training and a real-world dataset for testing, all captured using standardized off-the-shelf hardware. Training only using synthetic data, our lightweight networks can control simulated avatars in real-world data capture with high accuracy in real-time. Future directions include adding auxiliary loss during training to improve the accuracy of pose estimation, incorporating temporal information, and using scene-level information for better pose estimates.

References

[1] Introducing project aria, from meta. https://www.projectaria.com/, . Accessed: 2023-10-17. 2

[2] Meta quest 2: Immersive all-in-one VR headset. https://www.meta.com/ie/quest/products/quest-2/, . Accessed: 2023-11-16. 2

[3] Apple vision pro. https://www.apple.com/apple-vision-pro/, . Accessed: 2023-10-19. 2

[4] Meta quest 3: New mixed reality VR headset – shop now. https://www.meta.com/ie/quest/quest-3/, . Accessed: 2023-10-17. 1

[5] Hiroyasu Akada, Jian Wang, Soshi Shimada, Masaki Takahashi, Christian Theobalt, and Vladislav Golyanik. Unrealego: A new dataset for robust egocentric 3d human motion capture. *arXiv preprint arXiv:2208.01633*, 2022. 1, 2, 3, 6, 7

[6] Mazen Al Borno, Martin de Lasa, Aaron Hertzmann, and Senior Member. Trajectory optimization for full-body movements with complex contacts. 3

[7] Sadegh Aliakbarian, Pashmina Cameron, Federica Bogo, Andrew Fitzgibbon, and Thomas J Cashman. Flag: Flow-based 3d avatar generation from sparse observations. *arXiv preprint arXiv:2203.05789*, 2022. 2

[8] Sadegh Aliakbarian, Fatemeh Saleh, David Collier, Pashmina Cameron, and Darren Cosker. Hmd-nemo: Online 3d avatar motion generation from sparse observations. *arXiv preprint arXiv:2308.11261*, 2023. 2

[9] Beta Program. Unity real-time development platform. https://unity.com/. Accessed: 2023-11-18. 4, 12

[10] 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. *Lect. Notes Comput. Sci.*, 9909 LNCS:561–578, 2016. 6

[11] Steven Bohez, Saran Tunyasuvunakool, Philemon Brakel, Fereshteh Sadeghi, Leonard Hasenclever, Yuval Tassa, Emilio Parisotto, Jan Humplik, Tuomas Haarnoja, Roland Hafner, Markus Wulfmeier, Michael Neunert, Ben Moran, Noah Siegel, Andrea Huber, Francesco Romano, Nathan Batchelor, Federico Casarini, Josh Merel, Raia Hadsell, and Nicolas Heess. Imitate and repurpose: Learning reusable robot movement skills from human and animal behaviors. *arXiv preprint arXiv:2203.17138*, 2022. 4

[12] Nuttapong Chentanez, Matthias Müller, Miles Macklin, Viktor Makoviychuk, and Stefan Jeschke. Physics-based motion capture imitation with deep reinforcement learning. *Proceedings - MIG 2018: ACM SIGGRAPH Conference on Motion, Interaction, and Games*, 2018. 3

[13] Kyunghyun Cho, Bart van Merrienboer, Çaglar Gülçehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder–decoder for statistical machine translation. In *Conference on Empirical Methods in Natural Language Processing*, 2014. 15

[14] Yuming Du. Avatars grow legs: Generating smooth human motion from sparse tracking inputs with diffusion model. 2

[15] Stefan Elfwing, Eiji Uchibe, and Kenji Doya. Sigmoid-weighted linear units for neural network function approximation in reinforcement learning. *arXiv preprint arXiv:1702.03118*, 2017. 7

[16] Levi Fussell, Kevin Bergamin, and Daniel Holden. Supertrack: motion tracking for physically simulated characters using supervised learning. *ACM Trans. Graph.*, 40:1–13, 2021. 3

[17] Zigang Geng, Ke Sun, Bin Xiao, Zhaoxiang Zhang, and Jingdong Wang. Bottom-up human pose estimation via disentangled keypoint regression. *arXiv preprint arXiv:2104.02300*, 2021. 3

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

[19] Erik Gärtner, Mykhaylo Andriluka, Hongyi Xu, and Cristian Sminchisescu. Trajectory optimization for physics-based reconstruction of 3d human pose from monocular video. *arXiv preprint arXiv:2205.12292*, 2022. 3

[20] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *arXiv preprint arXiv:1512.03385*, 2015. 4

[21] Tairan He, Zhengyi Luo, Wenli Xiao, Chong Zhang, Kris Kitani, Changliu Liu, and Guanya Shi. Learning human-to-humanoid real-time whole-body teleoperation, 2024. 1

[22] Buzhen Huang, Liang Pan, Yuan Yang, Jingyi Ju, and Yangang Wang. Neural mocon: Neural motion control for physically plausible human motion capture. *arXiv preprint arXiv:2203.14065*, 2022. 3

[23] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015. 4

[24] Hao Jiang and Kristen Grauman. Seeing invisible poses: Estimating 3d body pose from egocentric video. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3501–3509. IEEE, 2017. 2

[25] Jiaxi Jiang, Paul Streli, Huajian Qiu, Andreas Fender, Larissa Laich, Patrick Snape, and Christian Holz. Avatarposer: Articulated full-body pose tracking from sparse motion sensing. *arXiv preprint arXiv:2207.13784*, 2022. 2

[26] Muhammed Kocabas, Nikos Athanasiou, and Michael J Black. Vibe: Video inference for human body pose and shape estimation. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 5252–5262, 2020. 3

[27] Sunmin Lee, Sebastian Starke, Yuting Ye, Jungdam Won, and Alexander Winkler. Questenvsim: Environment-aware simulated motion tracking from sparse sensors. *arXiv preprint arXiv:2306.05666*, 2023. 2

[28] Jiaman Li, C Karen Liu, and Jiajun Wu. Ego-body pose estimation via ego-head pose estimation. *arXiv preprint arXiv:2212.04636*, 2022. 2, 3

[29] Matthew Loper, Naureen Mahmood, Javier Romero, Gerard Pons-Moll, and Michael J Black. Smpl: A skinned multi-person linear model. *ACM Trans. Graph.*, 34, 2015. 4, 5, 12

[30] Zhengyi Luo, Ryo Hachiuma, Ye Yuan, and Kris Kitani. Dynamics-regulated kinematic policy for egocentric pose estimation. *NeurIPS*, 34:25019–25032, 2021. 2, 3, 6, 7, 13

[31] Zhengyi Luo, Shun Iwase, Ye Yuan, and Kris Kitani. Embodied scene-aware human pose estimation. *NeurIPS*, 2022. 2, 3, 13

[32] Zhengyi Luo, Jinkun Cao, Josh Merel, Alexander Winkler, Jing Huang, Kris Kitani, and Weipeng Xu. Universal humanoid motion representations for physics-based control. *arXiv preprint arXiv:2310.04582*, 2023. 4, 5, 13

[33] Zhengyi Luo, Jinkun Cao, Alexander W. Winkler, Kris Kitani, and Weipeng Xu. Perpetual humanoid control for real-time simulated avatars. In *International Conference on Computer Vision (ICCV)*, 2023. 3, 4, 7, 13

[34] Naureen Mahmood, Nima Ghorbani, Nikolaus F Troje, Gerard Pons-Moll, and Michael J Black. Amass: Archive of motion capture as surface shapes. *Proceedings of the IEEE International Conference on Computer Vision*, 2019-Octob: 5441–5450, 2019. 4, 5

[35] Viktor Makoviychuk, Lukasz Wawrzyniak, Yunrong Guo, Michelle Lu, Kier Storey, Miles Macklin, David Hoeller, Nikita Rudin, Arthur Allshire, Ankur Handa, and Gavriel State. Isaac gym: High performance gpu-based physics simulation for robot learning. *arXiv preprint arXiv:2108.10470*, 2021. 7

[36] Josh Merel, Saran Tunyasuvunakool, Arun Ahuja, Yuval Tassa, Leonard Hasenclever, Vu Pham, Tom Erez, Greg Wayne, and Nicolas Heess. Catch and carry: Reusable neural controllers for vision-guided whole-body tasks. *ACM Trans. Graph.*, 39, 2020. 3, 4

[37] Xiaqing Pan, Nicholas Charron, Yongqian Yang, Scott Peters, Thomas Whelan, Chen Kong, Omkar Parkhi, Richard Newcombe, and Carl Yuheng Ren. Aria digital twin: A new benchmark dataset for egocentric 3d machine perception. *arXiv preprint arXiv:2306.06362*, 2023. 5, 6

[38] Georgios Pavlakos, Vasileios Choutas, Nima Ghorbani, Timo Bolkart, Ahmed A A Osman, Dimitrios Tzionas, and Michael J Black. Expressive body capture: 3d hands, face, and body from a single image. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2019-June:10967–10977, 2019. 6

[39] Xue Bin Peng, Pieter Abbeel, Sergey Levine, and Michiel van de Panne. Deepmimic. *ACM Trans. Graph.*, 37:1–14, 2018. 3

[40] Xue Bin Peng, Michael Chang, Grace Zhang, Pieter Abbeel, and Sergey Levine. Mcp: Learning composable hierarchical control with multiplicative compositional policies. *arXiv preprint arXiv:1905.09808*, 2019. 3

[41] Jose Luis Ponton, Haoran Yun, Carlos Andujar, and Nuria Pelechano. Combining motion matching and orientation prediction to animate avatars for consumer-grade vr devices. In *Computer Graphics Forum*, pages 107–118. Wiley Online Library, 2022. 2, 3

[42] Davis Rempe, Leonidas J Guibas, Aaron Hertzmann, Bryan Russell, Ruben Villegas, and Jimei Yang. Contact and human dynamics from monocular video. *Lect. Notes Comput. Sci.*, 12350 LNCS:71–87, 2020. 3

[43] Jiawei Ren, Cunjun Yu, Siwei Chen, Xiao Ma, Liang Pan, and Ziwei Liu. Diffmimic: Efficient motion mimicking with differentiable physics. *arXiv preprint arXiv:2304.03274*, 2023. 3

[44] Helge Rhodin, Christian Richardt, Dan Casas, Eldar Insafutdinov, Mohammad Shafiei, Hans-Peter Seidel, Bernt Schiele, and Christian Theobalt. Egocap: egocentric marker-less motion capture with two fisheye cameras. *ACM Transactions on Graphics (TOG)*, 35(6):1–11, 2016. 2

[45] Stephane Ross, Geoffrey J Gordon, and J Andrew Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. *arXiv preprint arXiv:1011.0686*, 2010. 4, 5

[46] Andrei A Rusu, Sergio Gomez Colmenarejo, Caglar Gulcehre, Guillaume Desjardins, James Kirkpatrick, Razvan Pascanu, Volodymyr Mnih, Koray Kavukcuoglu, and Raia Hadsell. Policy distillation. *arXiv preprint arXiv:1511.06295*, 2015. 4

[47] Tara N. Sainath, Oriol Vinyals, Andrew W. Senior, and Hasim Sak. Convolutional, long short-term memory, fully connected deep neural networks. *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4580–4584, 2015. 15

[48] Simon Schmitt, Jonathan J Hudson, Augustin Zidek, Simon Osindero, Carl Doersch, Wojciech M Czarnecki, Joel Z Leibo, Heinrich Kuttler, Andrew Zisserman, Karen Simonyan, and S M Ali Eslami. Kickstarting deep reinforcement learning. *arXiv preprint arXiv:1803.03835*, 2018. 4

[49] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms, 2017. 4

[50] Soshi Shimada, Vladislav Golyanik, Weipeng Xu, and Christian Theobalt. Physcap: Physically plausible monocular 3d motion capture in real time. *arXiv preprint arXiv:2008.08880*, 2020. 3

[51] Soshi Shimada, Vladislav Golyanik, Weipeng Xu, Patrick Pérez, and Christian Theobalt. Neural monocular 3d human motion capture with physical awareness. *arXiv preprint arXiv:2105.01057*, 2021. 3

[52] Kiran K. Somasundaram, Jing Dong, Huixuan Tang, Julian Straub, Mingfei Yan, Michael Goesele, Jakob J. Engel, Renzo De Nardi, and Richard A. Newcombe. Project aria: A new tool for egocentric multi-modal ai research. *ArXiv*, abs/2308.13561, 2023. 1

[53] István Sárándi, Alexander Hermans, and Bastian Leibe. Learning 3d human pose estimation from dozens of datasets using a geometry-aware autoencoder to bridge between skeleton formats. *arXiv preprint arXiv:2212.14474*, 2022. 5

[54] Denis Tome, Thiemo Alldieck, Patrick Peluse, Gerard Pons-Moll, Lourdes Agapito, Hernan Badino, and Fernando De la Torre. Selfpose: 3d egocentric pose estimation from a headset mounted camera. *arXiv preprint arXiv:2011.01519*, 2020. 1, 2

[55] Jian Wang, Lingjie Liu, Weipeng Xu, Kripasindhu Sarkar, and Christian Theobalt. Estimating egocentric 3d human pose in global space. *arXiv preprint arXiv:2104.13454*, 2021. 2, 3

[56] Jian Wang, Lingjie Liu, Weipeng Xu, Kripasindhu Sarkar, Diogo Luvizon, and Christian Theobalt. Scene-aware egocentric 3d human pose estimation. *arXiv preprint arXiv:2212.11684*, 2022. 1, 3

[57] Jingbo Wang, Ye Yuan, Zhengyi Luo, Kevin Xie, Dahua Lin, Umar Iqbal, Sanja Fidler, Sameh Khamis, Hong Kong, and Mellon University, Carnegie. Learning human dynamics in autonomous driving scenarios. International Conference on Computer Vision, 2023, 2023. 3

[58] Tingwu Wang, Yunrong Guo, Maria Shugrina, and Sanja Fidler. Unicon: Universal neural controller for physics-based character motion. 2020. 3

[59] Alexander Winkler, Jungdam Won, and Yuting Ye. Questsim: Human motion tracking from sparse sensors with simulated avatars. *arXiv preprint arXiv:2209.09391*, 2022. 2, 3

[60] Jungdam Won, Deepak Gopinath, and Jessica Hodgins. A scalable approach to control diverse behaviors for physically simulated characters. *ACM Trans. Graph.*, 39, 2020. 3

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

[62] Yuxin Wu and Kaiming He. *Group Normalization*, pages 3–19. Springer International Publishing, 2018. 4

[63] Kevin Xie, Tingwu Wang, Umar Iqbal, Yunrong Guo, Sanja Fidler, and Florian Shkurti. Physics-based human motion estimation and synthesis from videos. *arXiv preprint arXiv:2109.09913*, 2021. 3

[64] Weipeng Xu, Avishek Chatterjee, Michael Zollhoefer, Helge Rhodin, Pascal Fua, Hans-Peter Seidel, and Christian Theobalt. Mo2cap2: Real-time mobile 3d motion capture with a cap-mounted fisheye camera. *arXiv preprint arXiv:1803.05959*, 2018. 1, 2

[65] Ye Yuan and Kris Kitani. 3d ego-pose estimation via imitation learning. In *Computer Vision – ECCV 2018*, pages 763–778. Springer International Publishing, 2018. 2, 3

[66] Ye Yuan and Kris Kitani. Ego-pose estimation and forecasting as real-time pd control. *Proceedings of the IEEE International Conference on Computer Vision*, 2019-Octob:10081–10091, 2019. 2, 3

[67] Ye Yuan and Kris Kitani. Residual force control for agile human behavior imitation and extended motion synthesis. *arXiv preprint arXiv:2006.07364*, 2020. 3, 14

[68] Ye Yuan, Shih-En Wei, Tomas Simon, Kris Kitani, and Jason Saragih. Simpoe: Simulated character control for 3d human pose estimation. *CVPR*, 2021. 2, 3, 13

[69] Dongxu Zhao, Zhen Wei, Jisan Mahmud, and Jan-Michael Frahm. Egoglass: Egocentric-view human pose estimation from an eyeglass frame, 2021. 1

[70] Yi Zhou, Connelly Barnes, Jingwan Lu, Jimei Yang, and Hao Li. On the continuity of rotation representations in neural networks. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2019-June:5738–5746, 2019. 3

[71] Ziwen Zhuang, Zipeng Fu, Jianren Wang, Christopher Atkeson, Soeren Schwertfeger, Chelsea Finn, and Hang Zhao. Robot parkour learning. *arXiv preprint arXiv:2309.05665*, 2023. 3, 15

Appendix

Appendices

A. Introduction

In this supplement, we provide additional details about SimXR that are left out of the main paper due to space constraints. Specifically, in Sec. B, we describe the contents of our supplementary site and videos. In Sec. C, we discuss the implementation details of our proposed synthetic dataset (Sec C.1), our proposed method SimXR (Sec C.2), pretrained imitators that act as teachers (Sec C.3), KinPoly-v (Sec C.4), and UnrealEgo (Sec C.5). Finally, in Sec. D, we provide some additional ablations (such as using recurrent networks) and analysis of failure cases. Since motion is best seen in videos, we strongly encourage our readers to view the provided videos for a qualitative analysis of our method. **All data and models will be released.**

B. Supplementary Site & Videos

In the supplement site, we provide an extensive qualitative evaluation of our method. We visualized all three subjects and **full** sequences of our real-world Quest 2 data capture, as well as results from the synthetic dataset. From the videos, we can see that our end-to-end method can follow the headset wearer's body motion closely in a physically plausible fashion. We also show results on AR/Aria glasses and compare with SOTA vision-based and physics-based methods. Last but not least, we visualize failure cases of our method.

C. Implementation Details

C.1. Synthetic Data Generation

Humanoid Kinematic Structure. To create the synthetic egocentric data, we use an internal human mesh model similar to SMPL [29]. Given kinematic body rotations and scale parameters, we can create its corresponding mesh as shown in Fig. A.1. We use a process similar to that of the SMPL humanoid to create an IsaacGym compatible humanoid for the in-house human mesh model.

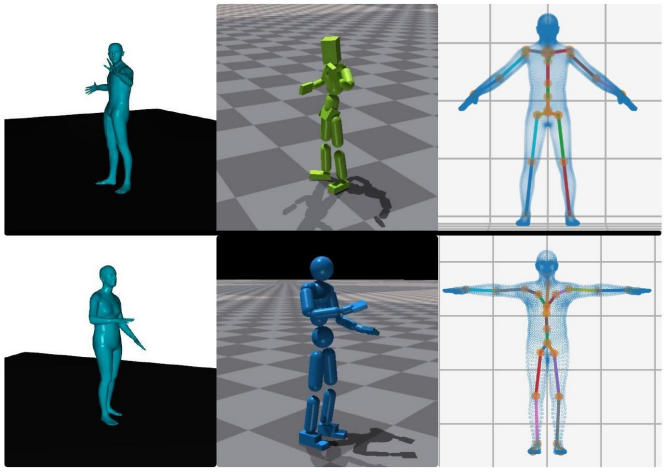

Figure A.1. The two human models we use, their rendered mesh, simulated humanoid, and kinemaitc structure. *(Top)*: our in-house humanoid with 24 DOF. *(Bottom)*: SMPL humanoid with 23 DOF.

MoCap Dataset. To generate synthetic data, we use a large-scale internal MoCap dataset consisting of 130 subjects and $>$ 1300 capture sessions. The motion capture dataset contains a large number of daily activities (walking, running, gesturing, yoga, dancing, balancing, sitting, interaction with objects *etc*.). We remove sequences that contain human-object interactions that are not possible to mimic without simulating the objects (such as sitting on chairs). Since each capture session contains a long sequence of motion, we further divide the sessions into sequences that contain around $\sim$450 frames of motion, resulting in a total of 5169 sequences for training and testing.

Rendering Pipeline. As mentioned in the main paper, rendering is done using the exact placement, intrinsic, and distortion of the cameras as the Quest 2 headset. The Unity [9] game engine is used for rendering. In Fig. A.3, we show examples of raw RGB images rendered using our pipeline. In each frame, we randomize the clothing, lighting, and background of the subjects as domain randomization. The background is rendered using a random image projected onto a skybox. We render each frame of motion from the MoCap dataset in 30 FPS. Each RGB image is rendered in a $640 \times 480 \times 3$ resolution, and we convert the images to monochrome and shrink them to $160 \times 120 \times 1$ for training and testing SimXR.

	Synthetic-Test, $E_{\text{pa-mpjpe}} \downarrow$					
	Head	L_Shoulder	L_Elbow	L_Wrist	R_Shoulder	R_Elbow
In-frame %	0.0%	4.0%	61.3 %	92.4 %	18.1 %	75.8%
In-frame / Out-frame	- / 30.2	28.0 / **25.0**	**30.7** / 33.2	**42.7** / 86.3	25.6 / **24.3**	**30.4** / 34.3
	R_Wrist	L_Knee	L_Ankle	R_Knee	R_Ankle	
In-frame %	96.5 %	99.0 %	98.4 %	99.4%	98.9%	
In-frame / Out-frame	**41.1** / 94.2	43.9 / **39.9**	**60.4** / 74.4	**43.7** / 45.5	**60.3** / 72.7	

Table A.1. Error analysis based on joint in-frame status. "In-frame" is not equivalent to "visible" due to self-occlusion.

Visibility Analysis. Here we conduct a visibility analysis on our generated synthetic data. As accounting for self-

Figure A.2. Self-occlusion visualization; blue dots are keypoints.

occlusion involves reprocessing the synthetic dataset and conducting ray-marching for each joint on the clothed mesh to ascertain visibility, we opt to use the "in-frame" (whether the joint is in one of the camera frames) statistics to approximate visibility. This measure is a reliable visibility indicator for upper body joints, but less so for the lower body, where self-occlusion and extreme viewpoints are more prevalent (see Figure A.2). From Table A.1 we can see that the shoulder and elbow joints are frequently outside the frame, while the wrist and leg joints often remain within the frame. For the shoulder, elbow, and knee joints, their in-frame and out-frame results are similar, since they are closer to the torso. For the body extremities (wrists and ankles), there is a clear gap, as the largest errors occur out of the frame.

C.2. Details about SimXR

Body Shape Used for Evaluation. We conduct all our training and evaluation using a fixed body shape for both of our humanoids (SMPL and in-house). In other words, we use the mean body shape for SMPL and a fixed body shape for our internal humanoid and do not vary bone lengths between different motion sequences or subjects. Since our framework does not involve any intermediate representations such as 3D keypoints or poses, SimXR is scale invariant. When conducting real-world evaluations, we simply adjust the height of the headset pose to match the standing head positions of the mean body shape. This is done in a calibration phase in which the subject is standing still. This process is effective as SimXR can estimate the pose for the three subjects who have different heights. Notice that our imitator can be trained to handle different body shapes, but we opt out of this option as estimating body shape from the distorted egocentric views is still an unsolved problem.

Training Process. The training process for SimXR is similar to training a motion imitator, with the distinction being that we provide images and headset pose as input instead of full-body reference pose. To better learn harder motion sequences, we use the same hard-negative mining process proposed in PHC [33] and PULSE [32]. Concretely, during training, given the full motion and image dataset $\hat{\boldsymbol{Q}}$, we evaluate the current policy on the full dataset and pick the sequences that the policy fails to form $\hat{\boldsymbol{Q}}_{\text{hard}}$. We keep updating $\hat{\boldsymbol{Q}}_{\text{hard}}$ at intervals until the success rate no longer increases. The hyperparameters for training SimXR can be found in Table A.2.

	Batch Size	Learning Rate	# of samples	image size	Image-latent
SimXR	1024	5×10^{-4}	$\sim 10^8$	160×120	512
	Batch Size	Learning Rate	# of samples		
PHC	3072	2×10^{-5}	$\sim 10^{10}$		

Table A.2. Hyperparameters for SimXR and PHC. Due to the increase in input size, SimXR is trained with significantly less samples than PHC and requires distillation.

	Synthetic-Train					
Method	Succ ↑	$E_{\text{g-mpjpe}}$ ↓	E_{mpjpe} ↓	$E_{\text{pa-mpjpe}}$ ↓	E_{acc} ↓	E_{vel} ↓
PHC	99.8%	25.6	20.4	15.5	2.4	3.5

Table A.3. Motion imitation result by the pretrained imitator on the in-house MoCap dataset.

C.3. Details about Pretrained Imitators

For the SMPL humanoid (AR / Aria glass experiments), we use an off-the-shelf motion imitator, PHC [33], trained on the AMASS dataset. We do not make any additional modifications to PHC, since the motion in the ADT dataset is relatively simple. For the in-house humanoid and VR / Quest experiments, we train an imitator using the same training procedure and hyperparameters provided in the PHC implementation. We train the imitator using the training sequences from the internal MoCap dataset, achieving the imitation performance shown in Table A.3. We can see that the imitator has a high success rate and a low joint error on the training data, which means that it is suitable to be used as a teacher for downstream tasks.

C.4. Details about KinPoly-v

We adapt KinPoly [30] to also consume images as input for egocentric pose estimation. In KinPoly, a policy is learned to produce kinematic full-body poses $\tilde{\boldsymbol{q}}_{t+1}$ based on the pose of the headset, which is then fed to an external force based motion imitator (UHC) for physics-based imitation. Comparing KinPoly to previous methods that use both an imitator and a pose estimator (*e.g.* SimPOE [68]), the main difference is whether the pose estimator is aware of the simulated humanoid state. In prior art, the pose estimator is not conditioned on the simulation state and operates independently from the physics simulation. This creates an open-loop system where the pose estimator can estimate a pose that drifts far away from the simulation state, leading the imitator to fall. KinPoly aims to create a closed-loop system where the pose estimator also takes the simulation state into consideration. This methodology is also used in EmbodiedPose [31]. However, the main problem with KinPoly is that, while it is relatively easy to output the correct reference kinematic pose $\tilde{\boldsymbol{q}}_{t+1}$ for the current timestep, it is difficult to compute the correct velocities $\tilde{\dot{\boldsymbol{q}}}_{t+1}$. Due to the ex-

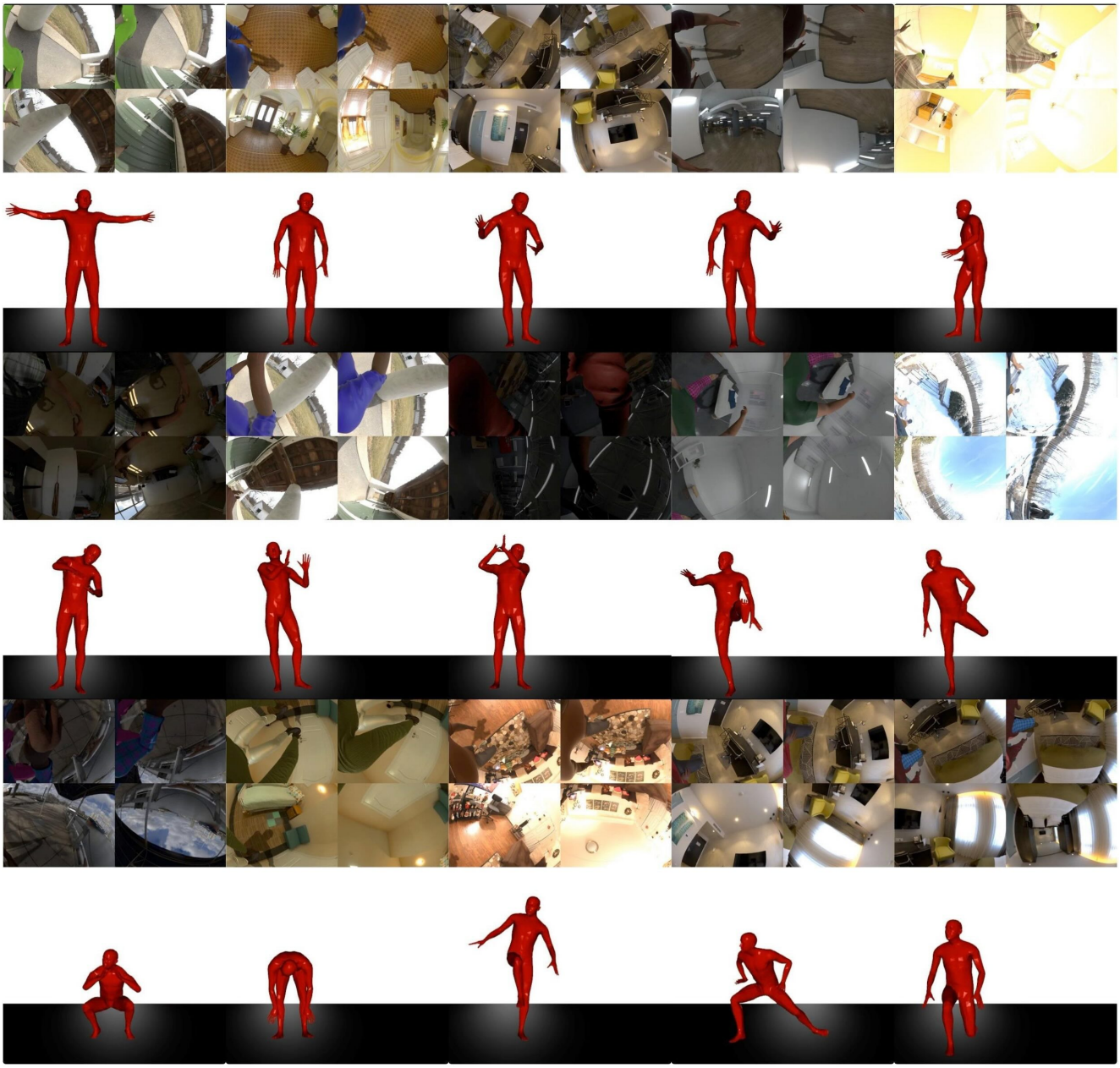

Figure A.3. Sample synthetic data with various poses. Here we include the original rendered RGB images for demonstration purposes. We randomize the actor's clothes, background, lighting at every frame.

ceptional capabilities of the non-physical forces, UHC does not require reference velocities as input. Concretely, UHC's goal state is defined as $\boldsymbol{s}_t^{\text{g-mimic}} \triangleq (\hat{\boldsymbol{\theta}}_{t+1} \ominus \boldsymbol{\theta}_t, \hat{\boldsymbol{p}}_{t+1} - \boldsymbol{p}_t)$, which does not contain any velocity information. As a result, KinPoly's kinematic policy only needs to predict body pose, but not velocity, which simplifies the learning problem. However, since PHC does not use any external forces [67], it requires accurate reference velocities as input.

To remove the dependency on external forces (change from UHC to PHC), we experimented with two forms of velocity prediction. The first approach is to compute the velocity as a finite difference between consecutive frames of the predicted reference poses: $\tilde{\dot{\boldsymbol{q}}}_{t+1} = \tilde{\boldsymbol{q}}_{t+1} - \tilde{\boldsymbol{q}}_t$. This formulation is problematic as large jumps in predicted poses can result in large velocities, which in turn lead to the imitator falling. Another approach is to directly predict the velocity as an output, which turns out to be more stable. We use this version as our implementation for KinPoly-v. However, this approach still suffers from inaccurate velocity prediction, as can be seen in the supplement videos: when the motion becomes faster and more dynamic (such as sports movement or jogging), it becomes difficult to predict the

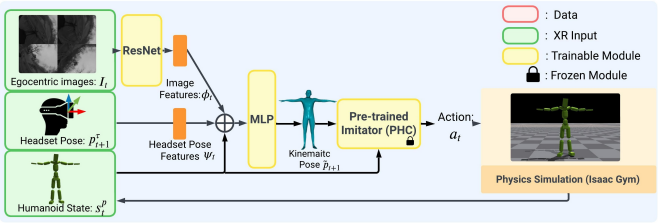

Figure A.4. KinPoly-v's network architecture. Different from distilling from a pretrained imitator, KinPoly-v outputs kinematic pose to the pretrained imitator for physcis-based motion imtiation.

	Synthetic-Test					
Method	Succ ↑	$E_{\text{g-mpjpe}}$ ↓	E_{mpjpe} ↓	$E_{\text{pa-mpjpe}}$	E_{acc} ↓	E_{vel} ↓
Ours (GRU)	91.6%	78.2	73.7	52.8	8.8	10.4
Ours	**96.2%**	**69.0**	**65.2**	**43.2**	**6.8**	**8.7**

Table A.4. Additional ablation on using recurrent architecture

correct velocities for the motion imitator. This can also be a result of the image input, which can be noisy and detrimental to the network learning a good velocity prediction.

Network Architecture. KinPoly-v shares the same input and network architecture as SimXR, with the only distinction being the output: KinPoly-v outputs the kinematic pose $\tilde{\boldsymbol{q}}_{t+1}$ rather than the PD targets $\boldsymbol{a}_t$ for the joints. KinPoly-v's architecture can be found in Fig.A.4

C.5. Details about UnrealEgo

We use the official UnrealEgo implementation, and pick the ResNet18 version with imagenet initialization for a fair comparison with SimXR. To be compatible with monochrome images, we replace the CNN layer with a single-channel convolutional layer and keep the Siamese network structure. We follow the official implementation and first train the 2D heatmap estimation network. Then, using the frozen heatmap estimation network, we train a 3D pose estimator based on the 2D heatmap input. We train the networks for three days to convergence, using a similar compute budget as training SimXR.

D. Additional Ablations and Failure Cases

Recurrent Networks. Currently, SimXR is a per-frame model without using any temporal model architecture. SimXR does rely on temporal information in the form of a simulation state and estimates temporally coherent motion by jointly considering humanoid state and input images. Based on the intuition that incorporating recurrent networks is essential to help robots complete tasks [71], we also tried recurrent architecture in our early experiments. We tested a simple GRU-based [13] architecture with 512 hidden units, and forms a lightweight Conv-LSTM [47]. Table A.4 shows that the use of a recurrent network does not offer an immediate performance increase. We hypothesize that, for a pose estimation task with dense per-frame input, a recurrent network may not be necessary to ensure good performance. Further investigation is needed to better leverage the temporal coherence in videos.

Additional Failure Cases. In our supplementary videos, we visualize the common failure cases of SimXR. Being one of the first methods to drive simulated avatars from images and headset pose input from XR headsets, SimXR shows the feasibility of training such a network, but it is still far from perfect.

- We can observe that the humanoid stumbles and drags its feet to stay balanced when moving around, which is caused by ambiguity in movement signals. Since our humanoid has no information about the future movements of the camera wearer, it adopts the foot-dragging behavior to be cautious and stay balanced.
- Accurate foot movement can still be challenging: while SimXR can estimate kicking and raising feet, it can also miss raised feet due to the challenging viewing angle. The foot can be barely visible even when raised, and the color of the garment can create additional ambiguities.
- We can observe that when the hands are held perfectly still, the humanoid can have micro movements due to inaccuracy in inferring the body poses. Tackling this issue is challenging, as the movement of the headset can cause the hands to move in the camera space but not in the global space, and differentiating between the two requires further investigation.
- The humanoid can also have erroneous hands movement from time to time, erecting the hand quickly and putting them down due to image noise and occlusion.
- Another source of inaccuracy is fast and sporty movement, where the humanoid can lag behind in performing the actions or fall down.

In the future, our aim is to incorporate a larger MoCap and synthetic datasets to improve the robustness of the controller. Introducing auxiliary pose estimation losses during training could also improve SimXR.

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