ABSTRACT
Real-time tracking of human body motion is crucial for interactive and immersive experiences in AR/VR. However, very limited sensor data about the body is available from standalone wearable devices such as HMDs (Head Mounted Devices) or AR glasses. In this work, we present a reinforcement learning framework that takes in sparse signals from an HMD and two controllers, and simulates plausible and physically valid full body motions. Using high quality full body motion as dense supervision during training, a simple policy network can learn to output appropriate torques for the character to balance, walk, and jog, while closely following the input signals. Our results demonstrate surprisingly similar leg motions to ground truth without any observations of the lower body, even when the input is only the 6D transformations of the HMD. We also show that a single policy can be robust to diverse locomotion styles, different body sizes, and novel environments.

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).

SA '22 Conference Papers, December 6–9, 2022, Daegu, Republic of Korea
© 2022 Copyright held by the owner/author(s).
ACM ISBN 978-1-4503-9470-3/22/12.
https://doi.org/10.1145/3550469.3555411

1 INTRODUCTION
A promise of AR/VR (Augmented, Virtual Reality) is that it will enable richer forms of self-expression and social experience compared to 2D video. This could be achieved through avatars that accurately capture a user’s movement and body language. To enable this, we need sensors and methods to faithfully reproduce the full body motion of a user in real-time.

Optical marker-based solutions [Vicon 2022] are often used in industries or research labs that require high precision. However, the setup is complex: It requires multiple cameras placed throughout the room, as well as attaching and calibrating markers on the users to be captured. A solution with less friction is markerless motion...
2 RELATED WORK

We categorize approaches to human motion tracking based on the type of sensor used as input. As the input signal becomes sparser, more of the pose reconstruction must be synthesized. Physics is a sensible prior when reconstructing those aspects of the pose which are not observed by any sensors. Therefore we end this section by reviewing physics-based approaches using Reinforcement Learning (RL), and contrast them to kinematic-based approaches.

2.1 Vision Sensors

Generating 3D full body poses based on camera images can be used for motion tracking. This image can be used to predict parameters for a parameterized and differentiable statistical human body model such as SMPL [Kanazawa et al. 2019; Loper et al. 2015; Rong et al. 2021; Xu et al. 2019]. Oftentimes, camera pixels are preprocessed to extract body keypoints [Cao et al. 2019] or body correspondences from the image [Güler et al. 2018]. One of the difficulties when using monocular cameras is the depth ambiguity, e.g. a short person closer to the camera looks identical to a taller person further away. This depth ambiguity can cause reconstructed 3D poses to accurately match the camera image when viewed from the same angle, whereas different views reveal unnatural leaning, scale and pose. Furthermore, if every frame is reconstructed individually, continuity between frames can be difficult and can cause poses to jitter. Constraining the poses through physics-based priors can help to mitigate these issues [Rempe et al. 2021].

2.2 IMU Sensors

Another popular approach is to use sensors mounted on a user’s body, for example inertial measurement units (IMUs). They are small, lightweight and don’t exhibit occlusion, since they are based on acceleration signal, versus vision. This allows them to be used across rooms, as well as outdoors and makes them agnostic to light and weather conditions. An offline method using IMUs was proposed by Marcard et al. [2017], where the system optimizes the pose parameters of the SMPL model so that it matches the sensor’s signal. The system performs best when it can access the entire sensor signal trajectory. In order to overcome this offline constraint, deep learning based approaches using mocap data paired with IMU signals were proposed. These models produce a local joint angle pose in realtime [Huang et al. 2018; Nagaraj et al. 2020]. Methods that estimate the full pose, including global 6D root from IMU signal were explored by [Jiang et al. 2022b; Yi et al. 2021]. Since kinematic pose prediction can be jittery or drift, these methods use predicted foot contacts, Stationary Boundary Points, or terrain prediction to improve the synthesized poses. Apart from IMUs, sensors based on electromagnetic (EM) fields have also been used to reconstruct full-body poses [Kaufmann et al. 2021]. However, most of the above approaches require sensors attached to the lower body.

2.3 HMD Sensors

As Head Mounted Devices (HMDs) for AR/VR are becoming more widely available, methods that generate full-body poses only from HMD and controllers are being explored. Dittadi et al. [2021] proposed a framework based on a variational autoencoder (VAE), where a VAE is first learned with all poses in the dataset, then the decoder is combined with another encoder which uses IMU signals as input. Aliakbarian et al. [2022] utilized a flow-based model instead of VAEs, where the invertible nature of flow-based models enables learning a shared latent space in a single learning phase. Jiang et al. [2022a] uses a Transformer architecture to predict the global skeleton state from only the HMD and controller poses. However, since these approaches are kinematic and don’t enforce physical constraints, they can suffer from artifacts such as foot-skating and jitter.

2.4 Physics-based Approaches

Many of the previous approaches are kinematic, meaning that the model has no notion of masses, inertias or forces as it synthesizes the poses. However, obeying physical laws makes synthesized motions more believable. This physics prior is especially important when the problem is under-constrained, where many solutions fulfill the sensor constraints but only a few are desirable. This is the case when using sparse sensor signal (head and controllers) to reconstruct physically accurate full-body poses.

Shimada et al. [2020] optimized the output kinematic motion predicted from a monocular video where physics laws are used as soft constraints. Yi et al. [2022] added a physics-based optimizer.
as a refinement process to further improve motions generated by kinematic methods such as TransPose [Yi et al. 2021], showing impressive results for IMU-based pose reconstruction. We show that we can generate motions of comparable quality without sensors on the lower body.

However, many methods still allow physics to be slightly violated, e.g., they are not enforced as hard-constraints. This can cause the posture of the characters to be abnormally tilted or for generated motions to appear to be floating in the air. We build much of our work on RL-based imitation learning, which adds a physics simulator as the final step before the pose generation, thereby enforcing physics as hard-constraints that cannot be violated [Peng et al. 2018a]. The controller outputs either target joint angles or torques which are used by the physics simulator to generate the next pose of the simulated character. This stream of research traditionally focused on imitating motions from dense, full-body user observations. Here, a variety of methods have explored how a single policy can imitate diverse motions, such as walking, jogging, break-dancing etc [Bergamin et al. 2019; Chentanez et al. 2018; Russell et al. 2021; Park et al. 2019; Won et al. 2020]. This becomes especially relevant for motion tracking, where we cannot know in advance what type of motion the user will perform, but require the policy to be able to track it. Another line of research is tracking on large mocap datasets to generate fundamental “building blocks” of motions, that can later be reused to achieve other downstream tasks [Merel et al. 2020; Peng et al. 2019, 2021; Won et al. 2021]. In the context of motion tracking these insights could be used to track the motion using a specific style.

Reinforcement Learning has also been used to imitate kinematic motions generated from sparser observations, such as monocular video [Peng et al. 2018b; Yu et al. 2021]. It has been successfully combined with egocentric video to synthesize poses and object interaction [Luo et al. 2021] and predict future motion [Yuan and Kitani 2019]. Luo et al. [2021] learned a universal controller and a kinematic policy and used it to generate simulated motions from a single-view egocentric video. In contrast to these video-based approaches, we show motion synthesis using only the 6 DoF state of the head and controllers. We use a simple architecture, consisting of a single MLP trained end-to-end. Since our physics simulation doesn’t use additional non-physical forces, our approach generates high-quality and believable motions.

3 METHOD

Traditional RL-imitation frameworks often assume that full-body, noise-free observations of the reference are available, which is not the case when tracking a user from only a HMD. In the following we detail our architecture based on Reinforcement learning, with important modifications for motion-tracking of users from sparse and real-world sensors.

3.1 RL background

We use reinforcement learning to train the character. The goal is to find a policy \( \pi_\theta \) to maximise the expected discounted return

\[
J(\pi_\theta) = \mathbb{E}_{\tau \sim \pi_\theta} \left[ \sum_{t=0}^T \gamma^t r_t \right],
\]

where \( \theta \) represents the weights of a neural network, \( \gamma \in [0, 1] \) is the discount factor, and \( \tau = (s_0, a_0, \ldots, s_{T+1}) \) represents trajectories collected by the policy interacting with the environment. The probability for a specific trajectory \( \tau \) when using the current policy \( \pi_\theta \) is given by \( P(\tau|\theta) = P(s_0) \prod_{t=0}^T P(s_{t+1}|s_t, a_t) \pi_\theta(a_t|s_t) \), where \( P(s_0) \) is the initial state distribution and \( P(s_{t+1}|s_t, a_t) \) represents the dynamics of the physics simulator. This probability is used to calculate the expectation \( \mathbb{E}_{\tau \sim \pi_\theta} \) which quantifies the expected return. To learn the policy weights \( \theta \), we use the proximal policy optimization (PPO) algorithm [Schulman et al. 2017].

3.2 Overview

The goal is to reconstruct the full body pose of a user from sparse user observations \( o_{user} \) as shown in Figure 2. This output pose is generated by a physics simulator. The user is driven by joint torques, which are produced by a neural network termed “Policy”. We use Reinforcement Learning, together with an imitation objective [Peng et al. 2018a], to train the policy to produce torques that track a user. During training we use a scalar reward \( r_t \) that captures the goal of producing a pose \( s_t \) that is as close as possible to the corresponding ground-truth pose \( s_{gt,t} \) from the motion database.

We hypothesize this architecture works well with sparse input data because of the high quality full body supervision signal during training, as well as because of the feedback of the simulated state back into the policy. Since the user observations are sparse and don’t provide much information about the user, considering the current state of the physically simulated character significantly reduces ambiguity on possible next poses. This allows the policy to use this internal state to decide on optimal actions. For kinematic tracking approaches, this feedback loop is conceptually similar to Recurrent Neural Networks (RNNs) or ‘Pose Priors’. However, complex architectures such as RNNs or Transformers are not required in our case and a 3-layer MLP policy is sufficient.

The policy requires observations as input in order to determine the torque at every simulation step. The observations can be split into three sections discussed in the following: the observations coming from the simulated character, the sparse observations coming from the sensors worn by the user, and the scale of the user.

Figure 2: Overview of our Imitation Learning architecture to track users from sparse sensor data. The dotted paths are only needed to train the policy network (MLP). During inference, this network produces torques for a physics simulator that generates the pose.
3.3 Simulated Character Observations

The simulated avatar has 33 degrees-of-freedom (DoF). It is fully observable, allowing the policy access to any values that are helpful for determining the torques. We use the joint angles \( \alpha_{\text{sim},q} \in \mathbb{R}^{33} \) and joint angle velocities \( \dot{\alpha}_{\text{sim},q} \in \mathbb{R}^{33} \). Even though redundant, we also give the policy access to the Cartesian positions and orientations of each link, which speeds up training. All the Cartesian positions \( \alpha_{\text{sim},p} \in \mathbb{R}^{16 \times 6} \) are expressed with respect to frame \( S \), which is located on the floor below the avatar and rotates according to its heading direction (see Figure 2). This allows the policy to learn torque mappings independent of the heading direction. The link orientations \( \alpha_{\text{sim},R} \in \mathbb{R}^{16 \times 6} \), also in frame \( S \), are encoded by the first two columns of their rotation matrices. This has been shown to be a favorable orientation representation for neural networks compared to quaternions or angle axis, which can introduce discontinuities. We also observe dynamic quantities such as contact forces of each foot \( \sigma_f \in \mathbb{R}^{2 \times 3} \), which allows the policy to reason about contact states when determining the torques. The policy also observes the simulated avatar’s linear and angular velocity of each of the links, which is necessary in order to take the inertia of the character into account when producing torques. In total the simulated avatar is observed by \( \alpha_{\text{sim}} \in \mathbb{R}^{312} \).

3.4 Synthetic Training Data

Training the policy requires sensor data of the user paired with ground-truth poses \( s_{\text{gt}} \) from which the reward is computed. Instead of recording a new dataset capturing the full-body pose of a user (e.g. with a marker-based setup) while wearing a headset, we synthetically generate this paired data. We offset the ground-truth head and wrist joints to emulate the position and orientation of a headset and left and right controllers as if the subjects were equipped with the devices. Since the real sensor signal is sufficiently clean, noise was not added to the offsets.

Our in-house Mocap data consists of 8 hours of motion clips of 172 subjects. Specifically, the dataset contains 130 minutes of walking, 110 minutes of jogging, 80 minutes of casual conversations with gestures, 90 minutes of whiteboard discussion and 70 minutes of balancing. Existing datasets lacked either subject diversity or motion diversity.

3.5 User Pose Observations

The sensor data, either coming from the real headset or synthetically generated for training, is given the pose and orientation of the headset \( h \), the left controller \( l \) and the right controller \( r \). Like the Cartesian observations of the simulated character, all user positions and orientations are relative to frame \( S \) as shown in Figure 2. The orientations \( S R_i \in \mathbb{R}^6 \) denote the first two columns of the rotation matrix of the head, left and right controller relative to frame \( S \) (see Sec. 3.3 for details).

\[
\alpha_{\text{user}, i} = [h_S, S R_h, l_S, S R_l, r_S, S R_r].
\]

The policy also has access to 6 future user observations. This gives the simulated avatar the ability to better anticipate and thereby track a user’s motion by knowing what they will do next. The trade-off is that relying on future poses introduces a 160ms latency, which makes real-time tracking less responsive. In total the pose observations coming from the user are \( \alpha_{\text{user}} \in \mathbb{R}^{6 \times 3 \times (3 + 6)} \in \mathbb{R}^{162} \).

3.6 User Scale observations

Many imitation learning approaches generate policies that are specific to one character of a particular scale. In order to deliver a general motion tracking solution, we want to be able to track users of any scale (tall or short). One approach is to generate individual policies per user scale and then blend them during inference. The downside is that this requires a unique policy for every user scale and there might not be sufficient mocap training data available for each user. So instead, we learn a single policy which generalizes to various user scales. To achieve this we give the policy access to the user scale in form of a scalar value, which specifies the user’s height in meters.

\[
\alpha_{\text{user}, \text{scale}} \in \mathbb{R}.
\]

This allows the policy to learn to adjust torques based on the user, for instance apply larger torques when tracking taller users with larger masses and inertia.

During training, when imitating a clip recorded by one of the 172 individual subjects, we use the first pose in the clip, which is an A-pose, to extract the height of the subject. This single scale value is used to initialize a simulated character with approximately the same scale. During inference, when a user wears the headset, we require the user to initially stand upright and use the height estimated by the headset to initialize their avatar.

3.7 Reward

The reward \( r_t \) is used to generate a desired behavior of the character. The goal for the simulated character is to imitate a user’s motion as close as possible. We build on the imitation reward introduced by Peng et al. [2018a]. Since during training the sparse observations are synthetically generated from the full-body mocap pose, the corresponding full-body pose \( s_{\text{gt}} \) is known and can be used to formulate this reward. This way the policy has dense supervision during training while requiring only sparse data during inference. Our reward function is given by the dot product

\[
r_t = w \cdot (r(q), \dot{r}(q), r(x), \dot{r}(x), r_f)^T,
\]

where the terms \( r(q) \) and \( \dot{r}(q) \) quantify the difference of joint angles and joint angle velocities between the simulated avatar and the ground truth mocap pose and \( r(x) \) and \( \dot{r}(x) \) quantify the difference in Cartesian positions and velocities of each joint. Each term is expressed using a Gaussian kernel \( r(s) = \exp(-k_s \sum_i \| s_{\text{sim}} - s_{\text{gt}} \|^2) \), where \( s_{\text{sim}} \) and \( s_{\text{gt}} \) are the particular representation of the simulated character and the ground truth pose respectively and \( k_s \) is the sensitivity of the kernel for each representation. Weights \( w \) and kernel sizes \( k \) can be found in the Appendix.

Using only the above terms when no lower-body user observations are available results in the simulated avatar taking short, high-frequency steps, which look unnatural. The additional reward term \( r_f = \exp(k_f \sum_i \max(0, f_g \cdot \text{prev} - f_{\text{prev}})) \) reduces this, by penalizing an abrupt decrease in vertical contact force \( f_g \in \mathbb{R} \), where \( L \) and \( R \) denote the left and right foot, and \( k_f \) the sensitivity of the Gaussian kernel. This encourages the avatar to first naturally
We show that the positions and orientations of Meta’s Quest head-
table. A quantitative evaluation is shown in Table 1 and described
for the last stable timestep. This also facilitated downsampling of the HMD sensor data during in-
ference, which is provided at 72 fps. As a physics simulator we use Nvidia’s PhysX, which has been wrapped by the RL training
framework IsaacGym [Makoviychuk et al. 2021]. This allows us to
simulate physics on the GPU and gives access to the simulation
data through PyTorch tensors [Paszke et al. 2019]. We combine
IsaacGym with the open-source, GPU-ready PPO implementation
rsl_rl [Rudin 2021; Rudin et al. 2021]. This allows us to train 4000
characters in parallel on an NVIDIA GTX 3080, which trains to
14 billion policy steps, equivalent to 13 years of human experi-
ence, in 48 hours. This is roughly 2 orders of magnitude faster than
CPU-based approaches. For a detailed description of RL algorithm
parameters see Appendix.

4 RESULTS

We show that the positions and orientations of Meta’s Quest head-
set and controllers contain enough signal to reasonable estimate the
full-body pose of a user. The framework is able to distinguish be-
 tween various locomotion modes, turning, and their transitions. It
is also able to track motions in which upper and lower body are less
correlated like writing on a whiteboard or boxing. The lower body
pose matches the user surprisingly accurately, so the correct foot
is often in contact at the right time and position. For a qualitative
evaluation, we encourage the readers to compare the reconstructed
simulated poses with the image references in Figure 3 and in the
video. A quantitative evaluation is shown in Table 1 and described
in the following.

Table 1: Pose reconstruction from synthesized and real Meta
Quest HMD (H) and controllers (C). We show that despite not
having sensors on the lower body, our metrics match state-of-
the-art methods like PIP [Yi et al. 2022] that use IMUs
attached to the legs. Limitations are discussed in Sec. 5.4.

| Method       | Ours | Ours | Ours | PIP |
|--------------|------|------|------|-----|
| Sensors      | H+2C | H+2C | H    | 6 IMUs |
| Test data    | Lafan | Real | Real | TC |
| MPJRE [deg]  | 5.7  | -    | -    | -   |
| MPJPE [cm]   | 3.7  | -    | -    | -   |
| RootE [cm]   | 1.8  | -    | -    | -   |
| SIP [deg]    | 12.3 | -    | -    | 12.9 |
| Jitter [km/s²] | 0.3 | 0.1  | 0.2  | 0.2 |
| MHPE [cm]    | 3.7  | 6.3  | 6.2  | -   |
| MHRE [deg]   | 8.4  | 14.3 | 14.7 | -   |

4.1 Quantitative Evaluation

The tracking accuracy for a variety of motions is summarized in
Table 1. We evaluate our trained model on synthetic input gen-
erated from the Lafan dataset [Harvey et al. 2020] as well as on
real data recorded while wearing the headset. We evaluate poses
reconstructed using the HMD and controllers (H+2C), and using
only the 6 DoF headset pose (H). We compare our metrics to a
state-of-the-art solution [Yi et al. 2022], which uses 6 IMUs, two
attached to the legs and evaluated on the TotalCapture (TC) dataset
[Trumble et al. 2017].

As metrics we use MPJRE (Mean Per Joint Rotation Error) in
degrees and MPJPE (Mean Per Joint Position Error) in centimeters.
We also compute the global root error in centimeters. To compare
to [Yi et al. 2022], we calculate the “SIP” error, which represents
the mean orientation error of the upper arms and legs in the global
space in degrees. Jitter represents the smoothness between poses,
where smaller is more smooth. This is quantified by the jerk, the
derivative of the acceleration, and calculated as done in [Yi et al.
2022]. Finally, we compute the MHPE/MHRE (Mean Headset Posi-
tion/Rotation Error), the error between the the simulated avatars
head/hands (+offset) and the global position and orientation of the
Quest devices.

5 DISCUSSION

We discuss three applications of our framework. The main ap-
lication is using sparse sensor information to reconstruct full-body
poses (Sec. 5.1). Then, we demonstrate how the same policy can
track users of different scale (Sec. 5.2). Finally, we demonstrate
how physics simulation can modify tracked motions to generate
believable interactions with the environment (Sec. 5.3). We end
with limitations of this approach (Sec. 5.4).

5.1 Headset and Controller Tracking

When comparing reconstructed poses in Figure 3, we observe that
the framework can distinguish different types of motion. To better
understand how the model is achieving this, we plot the real sensor
input driving these animations in Figure 4. This specific clip con-
tains three different motion types: writing, walking and jogging.
Starting with the writing section, we notice that the vertical head
position dips, hinting that the user is crouching and must be bend-
ing the knees. Other than that there is little correlation between
controllers and head. During the walking section, the controllers
oscillate in the horizontal directions (red and blue) relative to the
head. This might be a pattern the model uses to identify a walking
gait. In contrast to walking, jogging motions have no oscillation in
the x direction (red), but displays a very unique and high frequency
oscillation of the vertical HMD (green).

5.2 Tracking users of different scale

We show that a single policy can track users of different scale.
The user in Figure 3 is 180 cm tall, whereas the user in Figure 5 is
167 cm. The pose reconstruction is done by the same policy, without
retraining. The user scale during inference is determined based on
the initial height of the HMD, which causes the appropriately scaled
avatar to be initialized (see Sec. 3.6 for details).
Figure 3: Qualitative evaluation of pose reconstruction on motion sequences not used during training, corrected for latency. Most of the reconstructed poses match the video reference footage, despite having no lower-body sensor signal available for reconstruction. The four sequences demonstrate writing on a whiteboard, walking, jogging, and backwards walking with turns. Readers are encouraged to view the video for more examples.

Figure 4: Sparse input used to generate the full-body pose: the position of the headset (saturated) and left and right controller (less saturated). Y is the vertical dimension. Orientation information is also used by the model but not visualized here.

Figure 5: In this example a single user controls avatars of two different sizes using the same policy.

Apart from tracking users of different scale, we can also use avatars of different scale to track the same user. Figure 5 shows the same user being tracked by a larger avatar, which is a naive form of retargeting. Since the avatar is larger, but tries to track the position of e.g. the head, it naturally results in a more crouched position than the user. But generally the avatar imitates the users motion fairly well. More importantly, due to the physics simulator, the motion doesn’t violate any physical laws, such as foot skating or floor penetration that often appear when retargeting motions. With this naive baseline, future work could investigate how to further decouple users from avatar representation, allowing them to be embodied by whatever avatar they choose.

5.3 Environment Interaction and Adaptation

Figure 6: Top: The avatar pose is influenced by the environment and it can interact with virtual objects. Bottom: If a user controls an avatar in an environment different than their play space, the motion is adapted to match the virtual world (e.g. rough ground).

Since the characters are physically simulated (compared to kinematically tracked), their pose is influenced by collisions with the simulated environment (floor, objects). This adaptation of the tracked
pose by physics can compensate for missing sensor data and convincingly integrate characters with their virtual world. Two scenarios are shown in Figure 6.

In the top scenario a ball and a rigid object are loaded into the physics simulator. As a user is tracked, we observe a two-way interaction: The avatar influences the external objects (ball kick), and the objects can influence the pose of the avatar (trip hazard). This poses new research questions, for instance, how much should this semi-autonomous avatar be controlled by physics, and when should physics be violated to closely track a user?

The bottom scenario demonstrates an avatar tracking a motion on rough terrain, while the user and sensor signal is recorded on flat ground. This example demonstrates how the reconstructed motion adapts to unseen worlds to create believable animations, for instance the foot adapts to different slopes in the terrain. A similar application would be a user sitting on a chair, in which case the avatar’s mesh should not penetrate the virtual chair. It might be difficult to measure the user to this accuracy with sensors. However, the simulated collisions and gravity can correct the tracked avatar pose to avoid penetration or hovering.

5.4 Limitations
While this method can produce high quality tracking results for some motions, it can also fail to track others entirely. This is different than kinematic-leaning approaches [Yi et al. 2022], which, on difficult motions, might perform worse (e.g. more jitter, less accuracy), but nonetheless still track the motion. But since our motions are simulated without additional non-physical forces, moving the root of the character to a desired position requires a precise sequence of joint torques. Furthermore, physics simulation does not allow teleportation, so as the character drifts further from the user, it can become increasingly difficult to catch up. For these reasons the simulated character can fail when attempting to imitate a dynamic out-of-distribution motion for which it hasn’t yet learned the torque controls (e.g. break-dance, jumping).

Another difficulty comes from uncorrelated upper-lower body motions, resulting in different motions being represented by the same upper body sensor data. In this case, the policy will synthesize a natural and physically-valid lower body pose, but this might not match the user’s pose.

6 CONCLUSION AND FUTURE WORK
We presented a method to track users from sparse sensor data building on approaches from imitation learning. We show that physics simulation compensates for missing sensor information by synthesizing poses in a physically plausible way.

There are a variety of avenues for future work. In terms of motion quality, the simulated avatar can still look stiff and unnatural. Different reward strategies or GAN-based approaches [Peng et al. 2021] might improve the style with the goal of achieving VFX-quality animations from sparse input data. Secondly, due to the reliance on future observations, our real-time system currently has a latency of 160 ms. One way to reduce this and make the tracking more responsive could be to predict future poses while tracking [Yuan and Kitani 2019], maybe using Motion VAEs [Ling et al. 2020]. Next, the scale value (Sec. 3.6) is only a very coarse approximation of the user, since we linearly scale all elements (link length, mass, collision geometry, inertia) of a default skeleton equally. In reality, users of the same height can still have different proportions, resulting in different dynamics [Won and Lee 2019]. In the future we want to supply the policy with more detailed skeleton and body shape information. Finally, we want to increase the diversity of motions the avatars can imitate. This could be achieved using mixture-of-expert policies [Won et al. 2020; Xie et al. 2022], pre-trained low-level controllers that facilitate learning high-level tasks [Peng et al. 2022; Won et al. 2022], or more informative observation representations [Starke et al. 2022].

REFERENCES
Sadegh Ahakarian, Pashmina Cameron, Federica Bogho, Andrew Fitzgibbon, and Tom Cashman. 2022. FLAVG: Flow-based 3D Avatar Generation from Sparse Observations. In 2022 Computer Vision and Pattern Recognition. https://www.microsoft.com/en-us/research/publication/flavg-flow-based-3d-avatar-generation-from-sparse-observations/
Kerwin Bargainig, Simon Clavet, Daniel Holden, and James Richard Forbes. 2019. DBRec: Data-driven Responsive Control of Physics-based Characters. ACM Trans. Graph. 38, 6, Article 206 (2019). http://doi.org/10.1145/3355089.3356536
Z. Cao, G. Ridgele Martinez, T. Simon, S. Wei, and Y. A. Sheikh. 2019. OpenPose: Real-time Multi-Person 2D Pose Estimation using Part Affinity Fields. IEEE Transactions on Pattern Analysis and Machine Intelligence (2019).
Nuttapong Chentanez, Matthias Müller, Miles Macklin, Viktor Makoviychuk, and Stefan Jeschke. 2018. Physics-based motion capture imitation with deep reinforcement learning. In Motion, Interaction and Games, MIG 2018. ACM. 1:1–1:10. https://doi.org/10.1145/3274247.3274506
Andrea Dittadi, Sebastian Driadazio, Darren Cosker, Ben Lundell, Tom Cashman, and Jamie Shotton. 2021. Full-Body Motion From a Single Head-Mounted Device Generating SMPL Poses From Partial Observations. In International Conference on Computer Vision 2021. H. Durrant-Whyte and T. Bailey. 2006. Simultaneous localization and mapping: part I. IEEE Robotics Automation Magazine 13, 2 (2006), 99–110. https://doi.org/10.1109/MA.2006.1638022
Levi Fussell, Kerwin Bargainig, and Daniel Holden. 2021. SuperTrack: Motion Tracking for Physically Simulated Characters using Supervised Learning. ACM Trans. Graph. 40, 6, Article 197 (2021). https://doi.org/10.1145/3478513.348027
Riza Alp Güler, Natalia Neverova, and Iasonas Kokkinos. 2018. Densepose: Dense human pose estimation in the wild. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 7297–7306.
Felix G. Harvey, Mike Yuirec, Derek Novrouzehzahr, and Christopher Pal. 2020. Robust Motion In-betweening. 39, 4 (2020).
Yinghao Huang, Manuel Kaufmann, Emre Aksan, Michael J. Black, Ottmar Hilgues, and Gerard Pons-Moll. 2018. Deep Inertial Pooser: Learning to Reconstruct Human Motion from Sparse Electromagnetic Trackers. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 12906–12915.
Jiaxi Jiang, Paul Strel, Huajian Qiu, Andreas Fender, Larissa Laich, Patrick Snape, and Christian Holz. 2022a. AvatarPose: Articulated Full Body Pose Tracking from Sparse Motion Sensing. https://doi.org/10.48550/ARXIV.2207.13784
Yifeng Jiang, Yuting Ye, Deepak Gopinath, Jungdam Won, Alexander W Winkler, and C Karen Liu. 2022b. Transformer Inertial Pooser: Real-time Human Motion Reconstruction from Sparse IMUs with Simultaneous Terrain Generation. journal = {ACM Trans. Graph.} (2022).
Angjoo Kanazawa, Jiyang Zhang, Panna Felsen, and Itendra Malik. 2019. Learning 3D Human Dynamics from Video. In Computer Vision and Pattern Recognition (CVPR). Manuel Kaufmann, Yi Zhao, Chengcheng Tang, Lingling Tao, Christopher Twigg, Joe Song, Robert Wang, and Ottmar Hilgues. 2021. EM-POSE: 3D Human Pose Estimation from Sparse Electromagnetic Trackers. In International Conference on Computer Vision (ICCV).
Hung Vu Lang, Fabio Zinno, George Cheng, and Michiel Van De Panne. 2020. Character controllers using motion vaes. ACM Transactions on Graphics (TOG) (2020).
Matthew Loper, Naureen Mahmood, Javier Romero, Gerard Pons-Moll, and Michael J. Black. 2015. SMPL: A Skinned Multi-Person Linear Model. ACM TOG 34, 6 (Oct. 2015), 248:1–248:16.
Zhengyi Luo, Ryo Hachiuma, Yu Yuan, and Kris Kitani. 2021. Dynamics-regulated kinematic policy for egocentric pose estimation. Advances in Neural Information Processing Systems 34 2021.
Viktor Makoviychuk, Lukasz Wawrzyniak, Yunrong Guo, Michelle Lu, Kier Storey, Miles Macklin, David Hoeller, Nikita Rudin, Arthur Allahere, Ankur Handa, and Gabriele State. 2021. Isaac Gym: High Performance GPU-Based Physics Simulation For Robot Learning. https://doi.org/10.48550/ARXIV.2108.10470
Josh Merel, Saran Tunyasuvunakool, Arun Abjuh, Yuval Tassa, Leonard Hasenclever, Vu Pham, Tom Erez, Greg Wayne, and Nicolas Heess. 2020. Catch and Carry.
