Physics-based Motion Retargeting from Sparse Inputs

DANIELE REDA, University of British Columbia, Canada
JUNGDAM WON, Seoul National University, South Korea
YUTING YE, Reality Labs Research, Meta, United States of America
MICHIEL VAN DE PANNE, University of British Columbia, Canada
ALEXANDER WINKLER, Reality Labs Research, Meta, United States of America

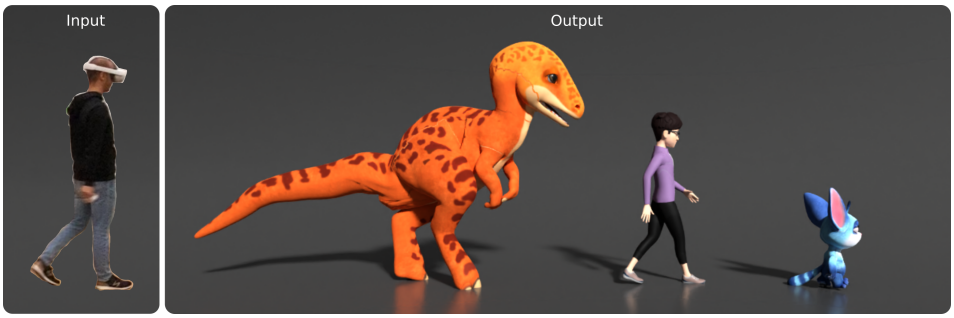

Fig. 1. Our method uses only a headset and controller pose as input to generate a physically-valid pose for a variety of characters in real-time.

Avatars are important to create interactive and immersive experiences in virtual worlds. One challenge in animating these characters to mimic a user's motion is that commercial AR/VR products consist only of a headset and controllers, providing very limited sensor data of the user's pose. Another challenge is that an avatar might have a different skeleton structure than a human and the mapping between them is unclear. In this work we address both of these challenges. We introduce a method to retarget motions in real-time from sparse human sensor data to characters of various morphologies. Our method uses reinforcement learning to train a policy to control characters in a physics simulator. We only require human motion capture data for training, without relying on artist-generated animations for each avatar. This allows us to use large motion capture datasets to train general policies that can track unseen users from real and sparse data in real-time. We demonstrate the feasibility of our approach on three characters with different skeleton structure: a dinosaur, a mouse-like creature and a human. We show that the avatar poses often match the user surprisingly well, despite having no sensor information of the lower body available. We discuss and ablate the important components in our framework, specifically the kinematic retargeting step, the imitation, contact and action reward as well as our asymmetric actor-critic observations. We further explore the robustness of our method in a variety of settings including unbalancing, dancing and sports motions.

CCS Concepts: • **Computing methodologies** → **Reinforcement learning**; **Physical simulation**.

Authors' addresses: Daniele Reda, dreda@cs.ubc.ca, University of British Columbia, Canada; Jungdam Won, Seoul National University, South Korea; Yuting Ye, Reality Labs Research, Meta, United States of America; Michiel van de Panne, University of British Columbia, Canada; Alexander Winkler, winklera@meta.com, Reality Labs Research, Meta, United States of America.

2577-6193/2023/8-ART33 $15.00
https://doi.org/10.1145/3606928

Additional Key Words and Phrases: retargeting, reinforcement learning, physics-based simulation, computer animation

ACM Reference Format:
Daniele Reda, Jungdam Won, Yuting Ye, Michiel van de Panne, and Alexander Winkler. 2023. Physics-based Motion Retargeting from Sparse Inputs. *Proc. ACM Comput. Graph. Interact. Tech.* 6, 3, Article 33 (August 2023), 19 pages. https://doi.org/10.1145/3606928

1 INTRODUCTION

Augmented and Virtual Reality (AR/VR) has the potential to provide rich forms of self-expression. By using human characters it is easier to accurately reflect the motions of a user. However, many users might want to portray themselves via non-human characters. Games with non-human player characters already demonstrate the great appeal of this type of embodiment, albeit one that works within the limited immersion afforded by current gaming input devices and displays. How can we best allow users to embody themselves in non-human characters using current AR/VR systems? Our work seeks to make progress on this question. This entails multiple challenges, in particular: (a) AR/VR systems provide only sparse information regarding the pose of the user, obtained from a head-mounted device (HMD) and two controllers. (b) The target character may have significantly different dimensions and body types, as shown in Figure 1; and (c) Kinematic animation, including that resulting from kinematic retargeting, often lacks physical plausibility, producing movements that lack a feeling of weight.

We propose a method to address these challenges. In particular, we develop an imitation-based reinforcement learning (RL) method that uses the sparse sensor input of a user to drive a physics-based simulation of the target character. This directly takes into account the physical properties of the given character, such as the heavy tail of a dinosaur or the short-legs of a mouse character, as shown in Figure 1. We only require human motion capture data for training, without relying on artist-generated animations for each avatar. This allows us to use large motion capture datasets to train general policies that can track unseen users from real and sparse data in real-time. We identify ingredients as being important to successful retargeting in this setting, including foot contact rewards, sparse mapping of key features for retargeting, and suitable reward terms that offer further style control. Many of the pieces that we rely on exist elsewhere in the literature. Our primary contribution lies with bringing them together in a way that enables a new retargeting capability well-suited to current AR/VR systems. We are the first to show a framework that works with *real* data from *sparse* sensors in *real* time while producing high-quality motions for non-human characters. We validate our design choices through a variety of ablations.

2 RELATED WORK

In this literature review we focus on the most relevant works in motion tracking, retargeting, and physics-based control.

2.1 Human Motion Tracking

Many solutions exist for full-body tracking of human motion, varying in their choice of sensors, the number of sensors, and their placement. Optical marker-based systems with external cameras remain the most common choice for applications requiring high accuracy, e.g., [Vicon 2022]. *Markerless* and *vision-based* approaches rely on cameras alone to generate full body poses. Common approaches leverage human body models such as SMPL as a pose prior [Kanazawa et al. 2019; Loper et al. 2015; Rong et al. 2021; Xu et al. 2019], using extracted keypoints or correspondences from the images [Cao et al. 2019; Güler et al. 2018], or use physics-based priors, e.g., [Rempe et al. 2021] *Wearable* sensors are another common choice, relying on sensors attached on the user's body, such

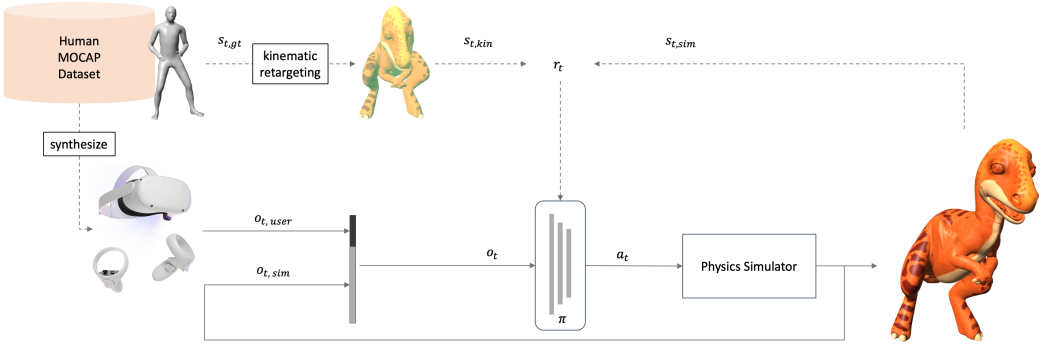

Fig. 2. Overview of our system. The policy π receives the Quest sensor input $o_{t,user}$ and the current state of the simulated character $o_{t,sim}$ as observation and computes torques a_t to apply to a physics simulator. During training, we use human motion capture data $s_{t,gt}$ to estimate a rough pose $s_{t,kin}$ of the simulated character ("kinematic retargeting"). The reward encourages the simulated character $s_{t,sim}$ to imitate this rough kinematic pose $s_{t,kin}$ as best as possible, while respecting all the physical constraints imposed by the simulator. After the policy is trained, full-body data or kinematic retargeting is not required anymore, and the simulated character can be driven purely by the HMD and controller sparse sensor.

as Inertial Measurement Unit (IMU) devices, e.g., [Huang et al. 2018; Jiang et al. 2022; von Marcard et al. 2017].

When using AR/VR devices, systems are further limited by the sparse sensors available. Most commonly available units are comprised of 3 tracker devices: a head-mounted device (HMD) and two controllers, one for each hand. As a human motion tracking device, these are handicapped by the lack of sensory information regarding the lower body and legs, which are essential to synthesizing believable full-body motion. Multiple methods have been proposed to address this, using transformers [Jiang et al. 2022; Vaswani et al. 2017], VAEs [Dittadi et al. 2021] and normalizing flows generative models [Aliakbarian et al. 2022]. Being kinematic-based approaches, however, these methods do not enforce physical properties and thus suffer from motion artifacts such as foot-skating and jitter. Physics-based approaches have also recently been proposed [Winkler et al. 2022; Ye et al. 2022]. These both make use of reinforcement learning and physics to learn general and robust policies that drive full-body avatars, conditioned on input from a VR device. These are closest to the work we present in this paper, and have great promise, although come with their own limitations. The Neural3Points method [Ye et al. 2022] is specific to a single user and uses auxiliary losses and an intermediate full-body pose predictor. Relatedly, Winkler et al. [2022] proposes a more direct approach that is able to control a simulated human avatar and generalizes to users of different heights and multiple type of motions. Our work generalizes the method of Winkler et al. [2022] in two important ways: (1) we learn physics-based retargeting to characters having different morphologies, and (2) we enable real-time retargeting.

2.2 Retargeting Motions

The motion retargeting problem is that of remapping motion from a source character or skeleton, often driven by motion capture data, to another character of possibly different dimensions. This is a long-standing problem for which many solutions have been proposed. Arguably the most challenging version of this problem arises when the source and target characters may differ significantly in terms of their morphology and skeleton, as is also the case for our work.

Kinematic retargeting methods often approach the problem by allowing the user to specify directly, or alternatively to learn via examples, a model for source-to-target pose correpondences, e.g., [Monzani et al. 2000; Seol et al. 2013; Yamane et al. 2010]. This creates a puppetry system, where target motions can be further cleaned to respect contacts with the help of inverse kinematics. Kinematic motion deformation approaches can be used to adapt multiple characters trajectories for motions involving coordination such as moving boxes [Kim et al. 2021]. Recent work proposes a kinematic method to learn how to retarget without requiring any explicit pairing between motions [Aberman et al. 2020], and this is also demonstrated to work on skeletons with very different proportions. Other recent work examines how to learn efficient kinematic motion retargeting for human-like skeletons while preserving contact constraints, such as when hands and arms have self-contact with the body [Villegas et al. 2021].

Physics-based retargeting methods aim to produce a physics-based simulation of the output motion, which results in crisp contacts and physically-plausible motion of the target character. An offline approach to motion retargeting using spacetime trajectory optimization is presented in Al Borno et al. [2018]. The final output uses LQR trees, and thus the given motions can cope with some perturbations. A method is recently proposed for using interactive human motion to drive the motion of a quadruped robot [Kim et al. 2022]. A curated dataset of matching pairs of human-and-robot motions is used to develop relevant kinematic mappings for particular motions or tasks. A deep-RL policy is then learned that can track the target kinematic motions in real time, enabling a form of real-time human-to-real-robot puppetry. In our setting, we assume significantly sparser user input and motion specifications.

2.3 Physics-based Character Simulation

Controllers for physics-based characters have been extensively explored. The ability to imitate reference motions was first demonstrated to varying extents in a number of papers over the past 15 years, e.g., [Coros et al. 2010; Geijtenbeek et al. 2012; Lee et al. 2010; Liu et al. 2010; Ye and Liu 2010; Yin et al. 2007]. These methods often incorporated some iterative optimization to adapt to a specific motion and used a simple control law to provide robust balance feedback. Some of these methods were also adapted to produce motions for non-human characters, e.g., [Geijtenbeek et al. 2013; Wampler et al. 2014].

Neural network policies, trained via deep reinforcement learning (RL), provide new capabilities to learn new skills from scratch, or to imitate artist-provided motions or motion capture clips, e.g., [Peng et al. 2018a, 2017; Won et al. 2017], including demonstrations for non-human characters. More recent methods provide more flexibility in sequencing motions for basketball [Park et al. 2019] or, more generally, to track online streams of motion capture data [Bergamin et al. 2019; Chentanez et al. 2018; Fussell et al. 2021; Won et al. 2020]. Control policies have also been learned which are conditioned on not only the desired motion, but also the specific morphology of a simulated character, which can then even be changed at run time [Won and Lee 2019]. We further refer the reader to a recent survey of RL-related animation methods [Kwiatkowski et al. 2022]. We build on the foundations provided above for our specific problem, namely how to retarget from sparse (and therefore potentially highly ambiguous) input data to a non-human physics-based character with very different dimensions and proportions.

3 METHOD

An overview of our system is shown in Figure 2. We use reinforcement learning to learn a policy that generates torques for a physics simulator. During training, we use human motion capture data to both synthesize HMD and controllers data for the policy, and to build a reward training signal.

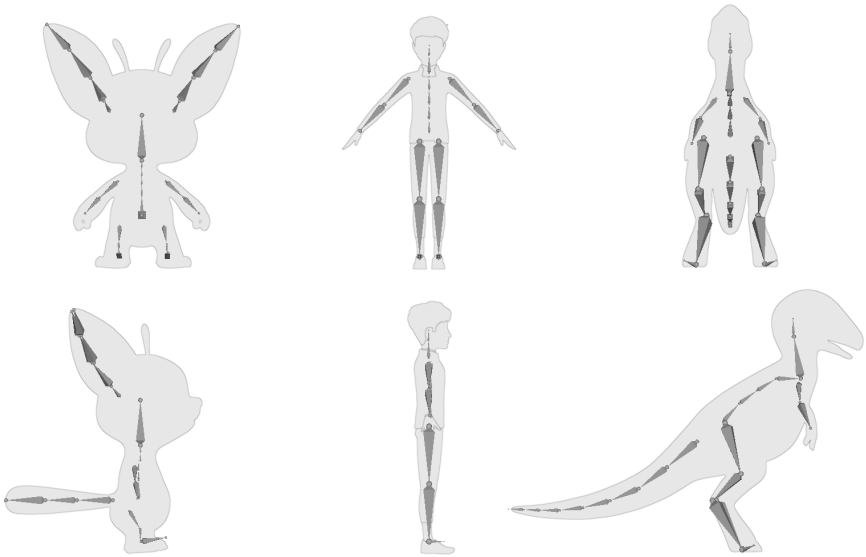

Fig. 3. We demonstrate our retargeting solution on three different characters (from left to right): a mouse-like creature named Oppy, a human named Jesse, and a dinosaur we call Dino.

In the following we give an overview of reinforcement learning and then describe each component in detail.

3.1 Reinforcement Learning

We use deep reinforcement learning (RL) to learn a retargeting policy for each character. In RL, at each time step t, the control policy reacts to an environment state s_t by performing an action a_t. Based on the action performed, the policy receives a reward signal $r_t = r(s_t, a_t)$. In deep RL, the control policy $\pi_\theta(a|s)$ is a neural network. The goal of deep RL is to find the network parameters θ which maximize the expected return defined as follows:

$$J_{RL}(\theta) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t)\right], \tag{1}$$

where $\gamma \in [0, 1)$ is the discount factor. Tuning γ affects the importance we give to future states. We solve this optimization problem using the proximal policy optimization (PPO) algorithm [Schulman et al. 2017], a policy gradient actor-critic algorithm. A review of PPO algorithm is provided in Appendix B.

3.2 Characters

We demonstrate our retargeting solution on three characters with unique features: Oppy [Meta 2023] is a mouse with a short lower body, a big head, big ears and a tail; Dino is a tall dinosaur, with a long and heavy tail and head, and short arms; Jesse is a human-like cartoon character with a skeleton structure similar to the mocap data. Figure 3 shows a visual representation of the characters and Table 1 details the structure of their skeletons.

Table 1. Character details.

Parameter	Oppy	Dino	Jesse
Weight (kg)	7	180	60
Height (cm)	80	250	180
Total links	24	30	16
Total DOF	58	44	32
Upper body joints	7	10	10
Lower body joints	6	8	6
Tail joints	3	8	-
Ear joints (x2)	4	-	-
Max Torque	40	300	300

3.3 Observations

The observation contains two parts: simulated character data $o_{t,sim}$ and user's sparse sensor data $o_{t,user}$.

$$o_t = [o_{t,sim}, o_{t-1,user}, o_{t,user}] \tag{2}$$

$$o_{t,sim} = [o_{sim,q}, o_{sim,\dot{q}}, o_{sim,x}, o_{sim,R}] \tag{3}$$

$$o_{t,user} = [h_t, l_t, r_t, R_{h,t}, R_{l,t}, R_{r,t}] \tag{4}$$

The simulated character's state is fully observable in the simulation. Therefore, even though the sensor signals is sparse, the policy can still rely on the full state of the simulated character. This observation consists of joint angles $o_{sim,q} \in \mathbb{R}^j$ and joint angle velocities $o_{sim,\dot{q}} \in \mathbb{R}^j$ of all degrees of freedom j of the character. We also provide Cartesian positions $o_{sim,x} \in \mathbb{R}^{l\times 3}$ and orientations $o_{sim,R} \in \mathbb{R}^{l\times 6}$ of a subset l of links of the character. The orientations consist of the first two columns of their rotation matrices. All positions and orientations are expressed with respect to a coordinate frame located on the floor below the character which rotates according to the character heading direction. This is useful to make the controller agnostic to the heading direction.

The sensor data, either coming from the real device or synthetically generated from the training data (described in subsection 3.4), consists of the position and orientation of the HMD h, the left controller l and the right controller r. Positions and orientations are expressed in the same coordinate system as the simulated character observations. To allow the policy to infer velocities, we provide it two consecutive sensor observations $[o_{t-1,user}, o_{t,user}]$.

Inspired by Pinto et al. [2018], we use asymmetric observations. At training time we augment the value function observation by providing the full human mocap pose and future human mocap state information. This complete view of the state allows the value function to better estimate the returns. The better the return estimate, the easier it is for the policy to learn. We are allowed to provide this mocap state information, because the value function is required only for training. Real-time inference still only relies on the policy, which uses the sparse sensor input. We ablate this in subsection 5.3 and find that is essential for sparse real time retargeting.

3.4 Synthetic Training Data

During training, we require HMD and controller data for the observation paired with kinematic poses for each character $s_{t,kin}$ from which the reward r_t is computed. To synthetically generate the HMD and controller data we offset the mocap head and wrist joints to emulate the position and orientation of HMD, left and right controllers as if the subjects were equipped with an AR/VR device.

Fig. 4. Training data is generated through kinematic retargeting. The left character is the human mocap data. The middle character shows a rough kinematic retargeting by matching selected joint angles. This pose has many artifacts, such as feet sliding due to different leg lengths, self-collisions, floor collisions, and no motion of the tail and ears. The right character is the closest simulated pose that also respects all physical constraints. Notice how the head does not perfectly follow the human, as it's heavier and takes more time to react, no having access to future information but only past and present.

Importantly, our system does not require artist-generated animations for each specific character as training data, which would be infeasible to create with the diversity and quantity we require. Instead we reuse existing human motion capture data s_{gt} and perform a rough kinematic retargeting s_{kin} to the morphology of the simulated character (Figure 4). In this step, we manually match selected joint angles of the human to conceptually similar joints of the creature. For joints where no correspondence can be found, we just set them to their default pose (e.g. ears and tails). This provides a rough estimate of the creature's motion. However, this motion has many artifacts, such a feet sliding due to different leg lengths, self-collisions, floor collisions, and no motion of the tail and ears. Nonetheless, we can still use it as a reward signal to train our simulated character. The physical constraints imposed by the simulation then remove remaining artifacts. Importantly, after the simulated character is trained, it is driven only by a headset and controllers, without requiring any full-body information of the user or any kinematic retargeting.

3.5 Reward

The goal for the simulated character is to imitate the human motion as closely as possible, while respecting all the constraints imposed by physics. Our reward function includes a component for imitation, contact, and action regularization:

$$r_t = r_t(\text{imitation}) + r_t(\text{contact}) + r_t(\text{action}) \tag{5}$$

$$r_t(\text{imitation}) = r_t(q) + r_t(\dot{q}) + r_t(x) + r_t(\dot{x}) + r_t(\text{orientation}) \tag{6}$$

$$r_t(\text{action}) = r_t(\text{action diff}) + r_t(\text{action min}). \tag{7}$$

Each of the reward terms is expressed using a weighted Gaussian kernel:

$$r_t(s) = w_s \mathrm{e}^{-k_s d(s_{t,sim}, s_{t,kin})} \tag{8}$$

where for each term only the specific component of the state s is considered and $d(s_{sim}, s_{kin})$ represent the distance metric between the simulated and kinematic components of the state, k is the sensitivity of the Gaussian kernel, and w is the weight of the reward component. Parameter values and details of the distance metrics for each term are provided in Appendix A.

3.5.1 Imitation Reward. This reward matches the available information between the simulated character s_{sim} and the kinematically retargeted ground truth pose s_{kin}. The five terms represent a weighted sum of the difference between the matching joint angles (q), joint angle velocities ($\dot{q}$), Cartesian coordinate positions (x) and velocities ($\dot{x}$), and orientation. The imitation reward term captures the degrees of supervision we want to transfer between human motion data and the simulated character. For clarity, Equation 6 is the general form which includes all possible terms, but the way they are used differs according to each character. The less supervision the imitation term provides, the more we rely on physics and the other components to generate a sensible motion.

Depending on the quality of our kinematically retargeted pose, we can choose which of the aspects of the pose we want the simulated character to imitate more closely. The least amount of supervision consists in only tracking their root position, which according to our experiments does not produce high-quality motions. On the other extreme, we also do not want to track every aspect of the kinematically retargeted pose. For example there is no tail motion in the human mocap data, so the kinematically retargeted pose has all tails set to a stiff default pose. However, a simulated character might want to move the tail to achieve balance and smoother motion. So we do not require these parts of the skeleton to imitate the kinematic pose.

Orientations are skeleton independent, so we rely on the actual human mocap data, not the kinematically retargeted pose to formulate the orientation rewards. We always formulate a reward that matches the characters root with the human mocap root, as well as the characters head orientation with the human head orientation. Ablations without these terms are provided in subsection 4.3.

3.5.2 Contact Reward. The contact reward is a boolean value that checks whether the simulated character's foot contact and the human's foot contact coincide. We estimate contact of the mocap data based on a velocity and height threshold. For the simulated character, we can directly access contact forces from the simulator and threshold those. In most cases the kinematically retargeted leg motion has a variety of artifacts, such as feet sliding or penetrating the ground. Imitating this pose is not physically-valid. Since this reward doesn't depend on the skeleton structure, it can be used for all bipedal characters equally and directly computed from human mocap. The contact reward is important to give further training supervision and generate the high-quality motions shown. Ablations are provided.

3.5.3 Action Reward. The action reward is a regularization term to minimize total amount of energy consumed by the character. It consists of two terms that minimize the difference in torque between two subsequent actions and minimize the absolute action value and is defined as:

$$r_t(\text{action diff}) = \frac{1}{N}\sum_{i}^{N}(a_{t-1,i} - a_{t,i})^2 \tag{9}$$

$$r_t(\text{action min}) = \frac{1}{N}\sum_{i}^{N} a_{t,i}^2 \tag{10}$$

where N is the total number of action values which the policy outputs. The purpose of these components is to incentivize overall lower energy movements and to minimize twitching with a smoother movement between poses.

3.6 Termination

As noted in multiple previous works [Peng et al. 2018b; Reda et al. 2020], early termination techniques are important for learning complex motions through reinforcement learning. We reset the environment when one of the following two termination conditions is satisfied: the character

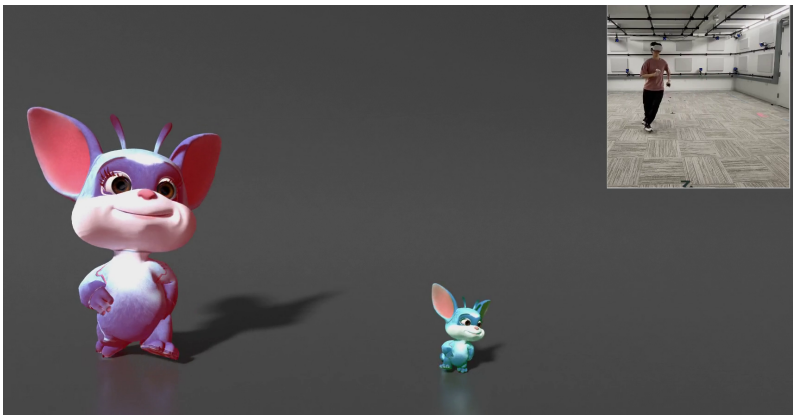

Fig. 5. If the character size matches the user, joint angles and foot contacts between the character and the user are more similar (left). If the simulated character has very different morphology (e.g. here much smaller), the kinematic-retarged pose is less accurate and is mostly ignored by the simulated character in order to generate a physically valid motion. Here the character has to take many steps for a single human step to match the root translation.

enters an unrecoverable state, which we define as falling and touching the ground with the upper body, or when the character root position is more than 30cm apart from the scaled root of the motion capture data. Furthermore, to mitigate the imbalance of visiting and learning to retarget only the early parts of the motion trajectories, we reset the character every 500 steps. We randomly sample a pose from the human data and set the character using the kinematically retargeted pose.

3.7 Learning Control Policies

The policy for each simulated character outputs torque values in the range $[-1, 1]$ which are then rescaled according to minimum and maximum torque values for each joint (provided in Appendix D). We find this to perform better and be more clear with respect to outputting PD target angles, as shown by previous works [Reda et al. 2020]. We train the policy with PPO and PyTorch auto differentiation software [Paszke et al. 2019; Schulman et al. 2017] and simulate physics with NVIDIA PhysX Isaac Gym physics simulator [Makoviychuk et al. 2021]. A complete set of hyperparameter details for reproducibility are summarized in Appendix C.

4 RESULTS

All experiments are performed on a single 12-core machine with one NVIDIA RTX 2070 GPU. All models are trained for 24 hours which translates to approximately 6 billion environment steps.

We demonstrate comparable results with two different motion capture datasets. Our in-house mocap data consists of 4 hours of motion clips of 120 subjects. Specifically, the dataset contains 130 minutes of walking and 110 minutes of jogging. We also demonstrate robust and general results with the Ubisoft La Forge Animation (LaFAN1) dataset [Harvey et al. 2020], an open source motion capture dataset containing 5 subjects and 77 sequences. For the purposes of this work, we only considered actions themed *Walk* and *Run*, which consist of a total of 15 sequences and 74 minutes of data. We note that these motions are very different from the ones in our in-house dataset, containing diverse and hard behaviors and gaiting styles. At inference, we provide input to the policy with a Meta Quest headset and controllers device.

Fig. 6. Right Dino has the orientation reward, while Left Dino does not. As the user turns its head, Right Dino follows more closely.

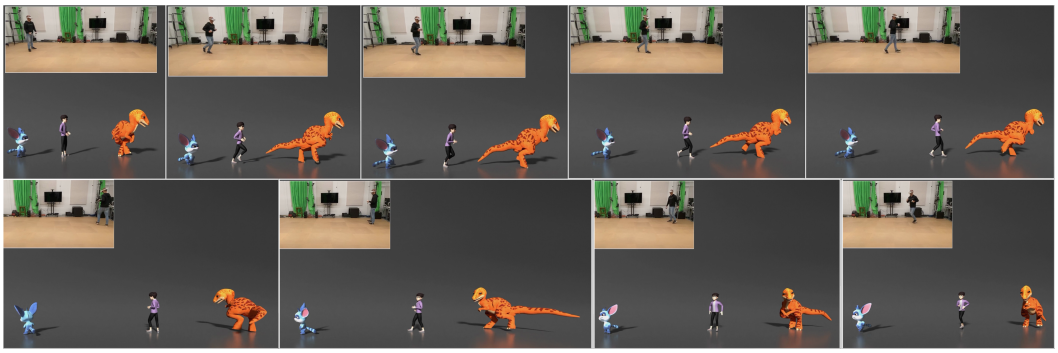

Fig. 7. Sequence of frames showing all three characters being controlled in real time with sparse sensory input. Lower-body motion perfectly matches the one of the user and feet contacts are correctly estimated. Watch the accompanying video for more results.

.

4.1 Real-time Retargeting with Headset and Controller

We thank the QuestSim [Winkler et al. 2022] authors for providing us with testing data and video references. With our method, we are able to control different characters in real time with only headset and controller information. Importantly, we are able to estimate the lower-body pose of the user from only three points in the upper body and correctly match the user action while transferring it to a character with a different morphology. Our virtual characters respect physical behaviors and do not suffer from jittering, foot sliding or penetration. Moreover, we are able to generalize to users not present in the training set and users of different heights. In Figure 7 we show a sequence in which all three characters are controlled by an unseen user.

4.2 Retargeting using only Headset

Some VR systems provide only a head-mounted device (HMD), without the two controllers. This provides an even more challenging domain, requiring the policy to predict a full-body pose and control a virtual character from a one-point input. Nonetheless, our trained models are robust to the lack of controller signal and are able to retarget real time user data from unseen users even from this extremely sparse input, albeit with a lower quality compared to before. We invite the reader to watch the video available in the supplementary material.

Fig. 8. Left Dino's tail has 2 active joints and the remaining 6 passive; Center Dino's tail has 2 active joints and the remaining 6 fixed; Right Dino's tail is completely passive.

4.3 Reward component ablations

Some reward components are essential to get good motions. Here we go through a few interesting examples.

4.3.1 Contact Reward. The contact reward shapes the gait style of the character. Both Oppy and Dino display different locomotion behaviors when using this reward component. Furthermore, as the character size changes, more signal can be transferred to the simulated character. In Figure 5 we show Oppy in two different sizes. When Oppy's size matches the user, it performs the same gait style and distance motions; when it is smaller, by matching the correct gait style it will travel less distance, while it can perform a faster gait to keep up with the user, depending on the weighting of the reward components. Similarly, in Figure 7, the different frames show the matching gaits between the three characters and the user.

4.3.2 Orientation Reward. Providing signal for mimicking head and root orientation is an essential component to support more fidelity in tracking user's head and overall movements. We show in Figure 6 how Dino without the head orientation component is unable to correctly move its head in the same way as the user. As shown in the supplementary video, both Oppy and Dino without head orientation reward component show the head wobbling left and right while walking. These characters have heavy heads needing learned control.

5 DISCUSSION

We discuss different capabilities and components of our system.

5.1 Physics-based control

Physics acts as a powerful helper in driving the motion of components with missing pose information, with the skeleton description as underlying prior. For the tail of Dino, the simulator affords several stylization options, i.e., whether we allow more joint mobility and passively actuate it through a PD controller with fixed-set input as secondary motion or we let the policy make active control decisions. In Figure 8 we show three examples, in which Dino's tail is fixed, passively actuated, or controlled by the learned policy. Tail and ears of Oppy are all treated as secondary dynamics. This stylization would not be possible in a kinematic retargeting setting.

5.2 Controlling the style

Our method is robust to different set of parameters. Once changed, most parameters still output a reasonable motion controller with different styles. As described in subsection 4.3, the contact

reward shapes the gait style of the character, and modified together with the size of the character would produce different gait styles.

The kinematic retargeting described in subsection 3.4 only requires a rough retargeting to produce sensible motions, as the physics dynamics correct the artifacts. Moreover, tuning the key joints for the kinematically retargeted motion produces an overall modification of the style. For example, it is possible to give Dino a more horizontal feeling, with the tail straight behind the back and not touching the ground, by tuning the spine parameter to be more bent over. An illustration of this tail is provided in Figure 4 and in the supplementary video, and noticeable difference can be observed compared to Figure 8.

5.3 Importance of asymmetric observations

During training we provide a richer observation to the value function compared to the one we provide to the policy. Specifically, while at inference the controller receives only real time sparse information (i.e. no future and no full-body pose), there is no need to constrain the value function since at training time this signal is available. In our experiments, we notice that the outcome of training a policy with a value function that receives no future and no full body pose, is an overall less robust policy. It is able to retarget easy walking examples coming from the training data, but it fails at harder motions like running and is incapable of generalizing to real data coming from an unseen user.

5.4 Quality of open-source datasets

We test our method with two different datasets, a 4 hour in-house dataset and a 74 minutes open-source dataset. While we notice that a larger and more diverse dataset improves the quality of the final motions, models trained with either of these datasets are robust and capable. Both are able to generalize to unseen users, and perform in real-time, even with headset-only sensory input.

6 CONCLUSIONS

We have presented a method to retarget a user's motion to simulated characters, in a challenging setting: the target characters can differ significantly in size and body morphology; we require a real-time remapping; and the mapping needs to be driven by the sparse motion data coming from an AR/VR device. We show that physics-based simulations, driven by asymmetric actor-critic RL policies, allow for effective retargeting in this difficult setting. The motions generated by the policies track those of the user while also being appropriate to the physics of the target character. We introduce a general reward description which allows for tuning of the degree of supervision and adapts to a range of character morphologies. Numerous ablations allow us to understand the impact of various parameters and design choices, including varying degrees of available tracking information, the impact of contact rewards, choices related to the secondary motion of tails and ears, and more.

Our work still has a number of limitations. Our controller fails to track challenging motion sequences, where the user performs fast and dynamic movements or uncorrelated upper/lower body motions. In these scenarios, a kinematic-based controller acting directly in the pose space will still be able to produce a motion, albeit not of high quality, and it will be able to catch up as parts of the motion become easier by "teleport" between poses without correlation. Instead, our controller has to produce a correct sequence of joint torques to control the character and may suffer from compounding tracking errors until it fails. An approach that divides the pipeline in two stages, similar to Ye et al. [2022] where first a network predicts the full-body pose and then a high-frequency controller outputs torques, could allow to regain the advantages of kinematic-based systems when needed. While our framework allows richer forms of self expressions for users,

empowering them to control different kind of characters, we are only scratching the surface of the complexities that arise due to different target skeletons. Our characters are still bipeds. Increased character complexity might be achieved by supplying skeleton information to the policy [Won and Lee 2019], using graph neural networks to learn a flexible policy similarly to Wang et al. [2018], or training an auxiliary network to find a mapping between source and target skeletons.

REFERENCES

Kfir Aberman, Peizhuo Li, Dani Lischinski, Olga Sorkine-Hornung, Daniel Cohen-Or, and Baoquan Chen. 2020. Skeleton-aware networks for deep motion retargeting. *ACM Transactions on Graphics (TOG)* 39, 4 (2020), 62–1.

Mazen Al Borno, Ludovic Righetti, Michael J Black, Scott L Delp, Eugene Fiume, and Javier Romero. 2018. Robust Physics-based Motion Retargeting with Realistic Body Shapes. In *Computer Graphics Forum*, Vol. 37. Wiley Online Library, 81–92.

Sadegh Aliakbarian, Pashmina Cameron, Federica Bogo, Andrew Fitzgibbon, and Tom Cashman. 2022. FLAG: Flow-based 3D Avatar Generation from Sparse Observations. In *2022 Computer Vision and Pattern Recognition*. https://www.microsoft.com/en-us/research/publication/flag-flow-based-3d-avatar-generation-from-sparse-observations/

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

Z. Cao, G. Hidalgo Martinez, T. Simon, S. Wei, and Y. A. Sheikh. 2019. OpenPose: Realtime 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 *Proceedings of the 11th annual international conference on motion, interaction, and games*. 1–10.

Stelian Coros, Philippe Beaudoin, and Michiel Van de Panne. 2010. Generalized biped walking control. *ACM Transactions On Graphics (TOG)* 29, 4 (2010), 1–9.

Andrea Dittadi, Sebastian Dziadzio, 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*.

Levi Fussell, Kevin Bergamin, and Daniel Holden. 2021. SuperTrack: motion tracking for physically simulated characters using supervised learning. *ACM Transactions on Graphics (TOG)* 40, 6 (2021), 1–13.

Thomas Geijtenbeek, Nicolas Pronost, and Frank van der Stappen. 2012. Simple data-driven control for simulated bipeds. In *Eurographics/ACM SIGGRAPH Symposium on Computer Animation (SCA)*.

Thomas Geijtenbeek, Michiel van de Panne, and A Frank Van Der Stappen. 2013. Flexible muscle-based locomotion for bipedal creatures. *ACM Transactions on Graphics (TOG)* 32, 6 (2013), 1–11.

Rıza 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.

Félix G. Harvey, Mike Yurick, Derek Nowrouzezahrai, and Christopher Pal. 2020. Robust Motion In-Betweening. 39, 4 (2020).

Yinghao Huang, Manuel Kaufmann, Emre Aksan, Michael J. Black, Otmar Hilliges, and Gerard Pons-Moll. 2018. Deep Inertial Poser: Learning to Reconstruct Human Pose from Sparse Inertial Measurements in Real Time. *ACM TOG* 37, 6 (12 2018).

Yifeng Jiang, Yuting Ye, Deepak Gopinath, Jungdam Won, Alexander W Winkler, and C Karen Liu. 2022. Transformer Inertial Poser: Real-time Human Motion Reconstruction from Sparse IMUs with Simultaneous Terrain Generation. *journal = ACM Trans. Graph.* (2022).

Angjoo Kanazawa, Jason Y. Zhang, Panna Felsen, and Jitendra Malik. 2019. Learning 3D Human Dynamics from Video. In *Computer Vision and Pattern Recognition (CVPR)*.

Jongmin Kim, Yeongho Seol, and Taesoo Kwon. 2021. Interactive multi-character motion retargeting. *Computer Animation and Virtual Worlds* 32, 3-4 (2021), e2015.

Sunwoo Kim, Maks Sorokin, Jehee Lee, and Sehoon Ha. 2022. Human Motion Control of Quadrupedal Robots using Deep Reinforcement Learning. In *Proceedings of Robotics: Science and Systems*. New York, USA.

Ariel Kwiatkowski, Eduardo Alvarado, Vicky Kalogeiton, C Karen Liu, Julien Pettré, Michiel van de Panne, and Marie-Paule Cani. 2022. A survey on reinforcement learning methods in character animation. In *Computer Graphics Forum*, Vol. 41. Wiley Online Library, 613–639.

Yoonsang Lee, Sungeun Kim, and Jehee Lee. 2010. Data-driven biped control. In *ACM SIGGRAPH 2010 papers*. 1–8.

Libin Liu, KangKang Yin, Michiel van de Panne, Tianjia Shao, and Weiwei Xu. 2010. Sampling-based contact-rich motion control. In *ACM SIGGRAPH 2010 papers*. 1–10.

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.

Viktor Makoviychuk, Lukasz Wawrzyniak, Yunrong Guo, Michelle Lu, Kier Storey, Miles Macklin, David Hoeller, Nikita Rudin, Arthur Allshire, Ankur Handa, and Gavriel State. 2021. Isaac Gym: High Performance GPU-Based Physics Simulation For Robot Learning. https://doi.org/10.48550/ARXIV.2108.10470

Meta. 2023. The World Beyond. https://github.com/oculus-samples/Unity-TheWorldBeyond.

Jean-Sébastien Monzani, Paolo Baerlocher, Ronan Boulic, and Daniel Thalmann. 2000. Using an intermediate skeleton and inverse kinematics for motion retargeting. In *Computer Graphics Forum*, Vol. 19. Wiley Online Library, 11–19.

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

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. 2019. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems* 32 (2019), 8026–8037.

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

Xue Bin Peng, Pieter Abbeel, Sergey Levine, and Michiel van de Panne. 2018b. DeepMimic: Example-guided Deep Reinforcement Learning of Physics-based Character Skills. *ACM Trans. Graph.* 37, 4, Article 143 (July 2018), 143:1–143:14 pages.

Xue Bin Peng, Glen Berseth, KangKang Yin, and Michiel van de Panne. 2017. Deeploco: Dynamic locomotion skills using hierarchical deep reinforcement learning. *ACM Transactions on Graphics (TOG)* 36, 4 (2017), 1–13.

Lerrel Pinto, Marcin Andrychowicz, Peter Welinder, Wojciech Zaremba, and Pieter Abbeel. 2018. Asymmetric Actor Critic for Image-Based Robot Learning. In *Robotics (Robotics: Science and Systems)*, Hadas Kress-Gazit, Siddhartha S. Srinivasa, Tom Howard, and Nikolay Atanasov (Eds.). MIT Press Journals. https://doi.org/10.15607/RSS.2018.XIV.008 Publisher Copyright: © 2018, MIT Press Journals. All rights reserved.; 14th Robotics: Science and Systems, RSS 2018 ; Conference date: 26-06-2018 Through 30-06-2018.

Daniele Reda, Tianxin Tao, and Michiel van de Panne. 2020. Learning to Locomote: Understanding How Environment Design Matters for Deep Reinforcement Learning. In *Proc. ACM SIGGRAPH Conference on Motion, Interaction and Games*.

Davis Rempe, Tolga Birdal, Aaron Hertzmann, Jimei Yang, Srinath Sridhar, and Leonidas J Guibas. 2021. Humor: 3d human motion model for robust pose estimation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 11488–11499.

Yu Rong, Takaaki Shiratori, and Hanbyul Joo. 2021. FrankMocap: A Monocular 3D Whole-Body Pose Estimation System via Regression and Integration. In *IEEE International Conference on Computer Vision Workshops*.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal Policy Optimization Algorithms. https://doi.org/10.48550/ARXIV.1707.06347

Yeongho Seol, Carol O'Sullivan, and Jehee Lee. 2013. Creature features: online motion puppetry for non-human characters. In *Proceedings of the 12th ACM SIGGRAPH/Eurographics Symposium on Computer Animation*. 213–221.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is All you Need. In *Advances in Neural Information Processing Systems*, Vol. 30.

Systems Vicon. 2022. Vicon Motion Systems https://www.vicon.com/.

Ruben Villegas, Duygu Ceylan, Aaron Hertzmann, Jimei Yang, and Jun Saito. 2021. Contact-Aware Retargeting of Skinned Motion. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 9720–9729.

Timo von Marcard, Bodo Rosenhahn, Michael Black, and Gerard Pons-Moll. 2017. Sparse Inertial Poser: Automatic 3D Human Pose Estimation from Sparse IMUs. *Computer Graphics Forum 36(2), Proceedings of the 38th Annual Conference of the European Association for Computer Graphics (Eurographics)* (2017), 349–360.

Kevin Wampler, Zoran Popović, and Jovan Popović. 2014. Generalizing locomotion style to new animals with inverse optimal regression. *ACM Transactions on Graphics (TOG)* 33, 4 (2014), 1–11.

Tingwu Wang, Renjie Liao, Jimmy Ba, and Sanja Fidler. 2018. Nervenet: Learning structured policy with graph neural networks. In *International conference on learning representations*.

Alexander Winkler, Jungdam Won, and Yuting Ye. 2022. QuestSim: Human Motion Tracking from Sparse Sensors with Simulated Avatars. In *SIGGRAPH Asia 2022 Conference Papers*. 1–8.

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

Jungdam Won and Jehee Lee. 2019. Learning body shape variation in physics-based characters. *ACM Transactions on Graphics (TOG)* 38, 6 (2019), 1–12.

Jungdam Won, Jongho Park, Kwanyu Kim, and Jehee Lee. 2017. How to train your dragon: example-guided control of flapping flight. *ACM Transactions on Graphics (TOG)* 36, 6 (2017), 1–13.

Yuanlu Xu, Song-Chun Zhu, and Tony Tung. 2019. Denserac: Joint 3d pose and shape estimation by dense render-and-compare. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 7760–7770.

Katsu Yamane, Yuka Ariki, and Jessica Hodgins. 2010. Animating Non-Humanoid Characters with Human Motion Data. In *Eurographics/ ACM SIGGRAPH Symposium on Computer Animation*, MZoran Popovic and Miguel Otaduy (Eds.). The Eurographics Association. https://doi.org/10.2312/SCA/SCA10/169-178

Yuting Ye and C Karen Liu. 2010. Optimal feedback control for character animation using an abstract model. In *ACM SIGGRAPH 2010 papers*. 1–9.

Yongjing Ye, Libin Liu, Lei Hu, and Shihong Xia. 2022. Neural3Points: Learning to Generate Physically Realistic Full-body Motion for Virtual Reality Users. *Computer Graphics Forum* (2022). https://doi.org/10.1111/cgf.14634

KangKang Yin, Kevin Loken, and Michiel Van de Panne. 2007. Simbicon: Simple biped locomotion control. *ACM Transactions on Graphics (TOG)* 26, 3 (2007), 105–es.

A REWARD DETAILS

Parameter values for each term of Equation 5 and Equation 8 are provided in Appendix A.

Table 2. Reward parameters for each character.

Parameter	Oppy	Dino	Jesse
w_q	1	1	4
k_q	20	20	25
$w_{\dot{q}}$	0	0	0.5
$k_{\dot{q}}$	-	-	1
w_x	1	1	2.5
k_x	6	6	6
$w_{\dot{x}}$	0	0	0.7
$k_{\dot{x}}$	-	-	2
w_{contact}	1.5	1.5	0
k_{contact}	1	1	-
$w_{\text{orientation}}$	1	1	2
$k_{\text{orientation}}$	3	3	3
$w_{\text{action diff}}$	2	2	1.5
$k_{\text{action diff}}$	150	150	10
$w_{\text{action min}}$	0.2	0.2	0.5
$k_{\text{action min}}$	25	25	25
fail reward	-5	-5	-5

Given the state of the simulated character and the ground truth pose coming from the motion capture dataset, the distance metric for the different imitation reward components is formulated as a weighted sum of the Euclidean distance between the two values:

$$d(x_{sim}, x_{gt}) = \sum_i w_i \|q_{x,sim} - q_{x,gt}\|_2^2 \tag{11}$$

where i represent the joint angles or the link positions and weights vary according to the character. As described in subsection 3.5, the imitation reward defines the degree of supervision. As the two characters are closer alike, we can rely more on this reward. For Jesse, in fact, all joint weights are equal to 1. For Oppy and Dino, which have a different lower body size compared to a human, we rely more on the style reward for a good motion and decrease the weight of all lower body joints to 0.3. For link weights, for Oppy and Dino we set all weights to zero other than for the root, which is set to 1, for Jesse we track also end effectors.

Contact distance metric is also computed through the Euclidean distance between ground truth human motion data and simulated character data. We define that a human foot is in contact if its height is less than 20cm above the ground and the norm of its velocity is less than 0.4 m/s. For the simulated character, a force threshold of 1 N is set on the feet link.

The orientation distance metric, given the two orientations in quaternions, first computes the composition of the ground truth quaternion with the inverse of the simulated quaternion. Then, takes the distance norm of its axis angle representation.

B PROXIMAL POLICY OPTIMIZATION

Let an experience tuple be $e_t = (o_t, a_t, o_{t+1}, r_t)$ and a trajectory be $\tau = \{e_0, e_1, \ldots, e_T\}$. We episodically collect trajectories for a fixed number of environment transitions and we use this data to

train the controller and the value function networks. The value function network approximates the expected future returns of each state, and is defined for a policy π as

$$V^{\pi}(o) = E_{o_0=o, a_t \sim \pi(\cdot|o_t)} \left[\sum_{t=0}^{\infty} \gamma^t r(o_t, a_t) \right].$$

This function can be optimized using supervised learning due to its recursive nature:

$$V^{\pi_\theta}(o_t) = \gamma\, V^{\pi_\theta}(o_{t+1}) + r_t,$$

where

$$V^{\pi_\theta}(o_T) = r_T + \gamma V^{\pi_{\theta_{old}}}(o_{T+1}).$$

In PPO, the value function is used for computing the advantage

$$A_t = V^{\pi_\theta} - V^{\pi_{\theta_{old}}}$$

which is then used for training the policy by maximizing:

$$L_\pi(\theta) = \frac{1}{T} \sum_{t=1}^{T} \min(\rho_t \hat{A}_t,\ \text{clip}(\rho_t, 1-\epsilon, 1+\epsilon)\hat{A}_t),$$

where $\rho_t = \pi_\theta(a_t|o_t) \,/\, \pi_{\theta_{old}}(a_t|o_t)$ is an importance sampling term used for calculating the expectation under the old policy $\pi_{\theta_{old}}$.

C TRAINING PARAMETERS

Table 3. Training details. We use the same parameters for every character.

Hyperparameter	Value
Learning Rate	1.2×10^{-4}
Optimizer	Adam
Batch Size	8192
Num Environments	4096
Episode Steps	16
Num PPO Epochs	5
Discount Factor γ	0.97
GAE-Lambda	0.95
Value Loss Coefficient	0.2
Clip parameter	0.2
Max Grad Norm	1.0
Exploration Noise	0.02
Control time	36 fps
Activation function	Tanh
Policy network	[300, 200, 100]
Value network	[400, 400, 300, 200]

D TORQUE LIMITS

Table 4. Torque limit scale value for each character's joint. If not written, then value is not scaled. Minimum value is negative of maximum value. Moreover, Oppy's tail and ears do not have torque values as they are passively actuated, similarly to final six joints of Dino's tail.

Parameter	Oppy	Dino	Jesse
Shoulder	0.2	0.2	0.2
Elbow	0.1	0.1	0.1
Head	0.1	0.1	0.1
Neck	0.1	0.1	0.1
Spine0	1	1	0.25
Spine1	1	1	0.25
Spine2	1	1	0.25
Spine3	1	1	0.25
Tail0	-	0.5	-
Tail1	-	0.4	-