Neural Kinematic Networks for Unsupervised Motion Retargetting

Ruben Villegas [1,*] Jimei Yang [2] Duygu Ceylan [2] Honglak Lee [1,3]
[1]University of Michigan, Ann Arbor
[2]Adobe Research
[3] Google Brain

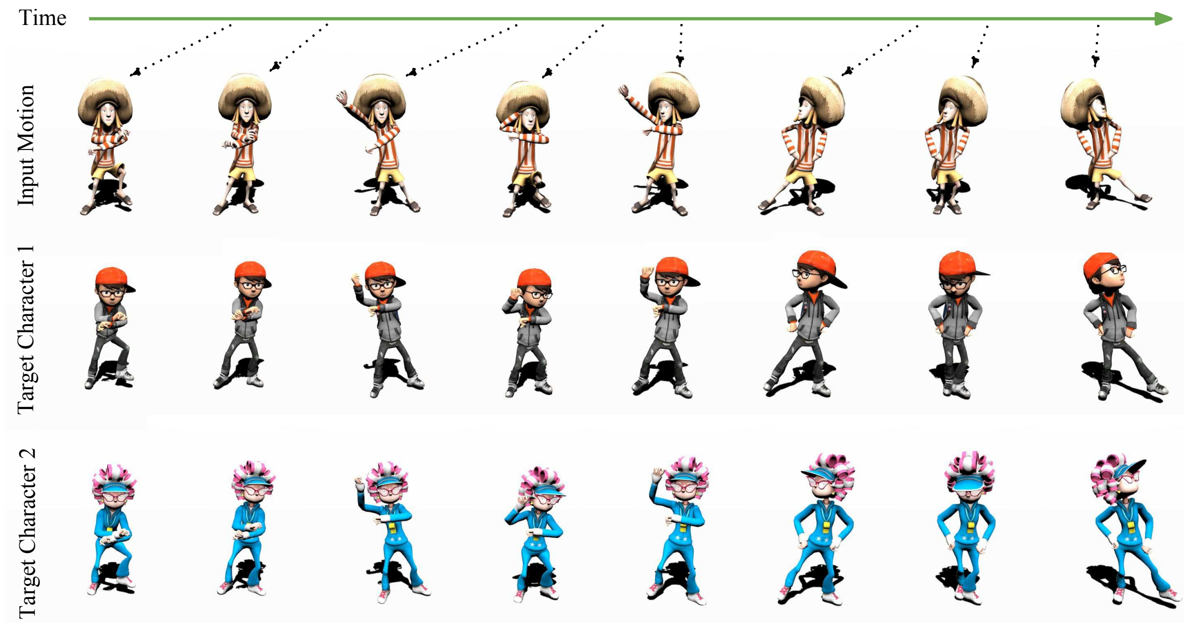

Figure 1: Our end-to-end method retargets a given input motion (top row), to new characters with different bone lengths and proportions, (middle and bottom row). The target characters are never seen performing the input motion during training.

Abstract

We propose a recurrent neural network architecture with a Forward Kinematics layer and cycle consistency based adversarial training objective for unsupervised motion retargetting. Our network captures the high-level properties of an input motion by the forward kinematics layer, and adapts them to a target character with different skeleton bone lengths (e.g., shorter, longer arms etc.). Collecting paired motion training sequences from different characters is expensive. Instead, our network utilizes cycle consistency to learn to solve the Inverse Kinematics problem in an unsupervised manner. Our method works online*, i.e., it adapts the motion sequence on-the-fly as new frames are received. In our experiments, we use the Mixamo animation data [1] to test our method for a variety of motions and characters and achieve state-of-the-art results. We also demonstrate motion retargetting from monocular human videos to 3D characters using an off-the-shelf 3D pose estimator.*

* Most of this work was done during Ruben's internship at Adobe.
[1]https://www.mixamo.com. See details in Section 5.

1. Introduction

Imitation is an important learning scheme for agents to acquire motor control skills [32]. It is often formulated as learning from expert demonstrations with access to sample trajectories of state-action pairs [3, 15]. However, this first-person imitation assumption may not always hold since 1) the teacher and the learner may have different physical structures, e.g., a human being vs a humanoid robot [4, 33] and 2) the learner may only observe the states of the teacher, e.g. joint positions, but not the actions that generate these states [28]. Adapting the motion of the teacher, e.g., a person, to the learner, e.g., a humanoid robot [2] or an avatar [34, 27], is often referred as motion retargetting in robotics and computer animation. This paper focuses on retargetting motions from a source to any target character with a known but different kinematic structure in terms of bone lengths and proportions. Skeletal differences between the source and target characters create the necessity of disentangling skeleton-independent features of the source motion and automatically adapting them to a target character in one shot, ideally without any post-processing optimization and hand-tuning steps.

Furthermore, a faithful solution needs to ensure the retargetted motion to be natural and realistic-looking which has been a long-standing challenge for animation.

Deep neural networks are known to have the ability to learn high-level features in sequential data that humans may not be able to easily identify, and have already achieved remarkable performance in machine translation [20] and speech recognition [13]. However, human motions are highly nonlinear and intrinsically constrained by kinematic structures of the skeletons. Thus classic sequence models such as recurrent neural networks (RNNs) may not be directly applicable to motion retargetting.

In this paper, we propose a novel neural network architecture to perform motion retargetting between characters with different skeleton structures (i.e., same topology but different bone length proportions). Our architecture relies on an analytic *Forward Kinematics* layer and two RNNs that work together to (i) encode the input motion data to motion features, and (ii) decode the joint rotations of the target skeleton from the identified features. The forward kinematics layer takes as input the joint rotations and the T-pose of a target skeleton, and renders the resulting motion. This fully differentiable layer forces the network to discover valid joint rotations by enabling to reason about the realism of the resulting motion. We use an adversarial training objective, rooted on the cycle consistency principle [44], to learn motion retargetting in an unsupervised way. In particular, the motion retargetted onto a target character should generate the original motion of the source character when retargetted back. Furthermore, the generated motion should be as natural as other known motions of the target character for an adversarially trained discriminator. The decoder RNN is conditioned on the target character, and together with the adverserial training, is able to generate natural motions for unseen characters as well. In our experiments, we show that the proposed method can perform *online* motion retargetting, i.e., adapting the input motion sequence on-the-fly as new frames are received. We also use 3D pose estimates from video sequences, e.g., in Human 3.6M dataset [18], as input to our network to animate Mixamo 3D characters.

The contributions of our work are summarized below:

- A novel *Neural Kinematic Network* consisting of two RNNs and a forward kinematics layer that automatically discovers the necessary joint rotations (i.e., solution to the *Inverse Kinematics (IK)* problem) for motion retargetting without requiring ground-truth rotations during training.

- A sequence-level adversarial cycle consistency objective function for unsupervised learning for motion retargetting which does not require input/output motion pairs of different skeletons during training.

2. Related work

Gleicher [11] first formulated motion retargetting as a spacetime optimization problem with kinematic constraints that is solved for the entire motion sequence. Lee and Shin [22] proposed a decomposition approach that first solves the IK problem for each frame to satisfy the constraints and then fits multilevel B-spline curves to achieve smooth results. Tak and Ko [35] further added dynamics constraints to perform sequential filtering to render physically plausible motions. Choi and Ko [9] proposed an online retargetting method by solving per-frame IK that computes the change in joint angles corresponding to the change in end-effector positions while imposing motion similarity as a secondary task. While the above-mentioned approaches require iterative optimization with hand-designed kinematic constraints for particular motions, our method learns to produce proper and smooth changes of joint angles (solving IK) in one-pass feed-forward inference of RNNs, and is able to generalize to unseen characters and novel motions. The idea of solving approximate IK can be traced back to the early blending-based methods [31, 21]. A target skeleton can be viewed as a new style. Our method can be applied to motion style transfer that has been a popular research area in computer animation [6, 17, 29, 40, 42].

Different machine learning algorithms have been used in modeling human motions. Early works used auto-regressive RBMs [36] or Gaussian process dynamic models [38, 14] to learn human motions in small scale. In particular, Grochow et al. [14] solves IK by constraining the generated poses to a learned Gaussian process prior. With the surge of deep learning, a variety of neural networks have been used to synthesize human motions [10, 16, 19, 7, 25, 23]. These networks are not applicable to motion retargetting as they directly generate the xyz-coordinates of joints and thus require a further post-processing to ensure bone length consistency. Instead, our method predicts quaternions that represent the rotation of each joint with respect to the T-pose without rotation supervision, which admits an end-to-end solution to motion retargetting and also has the potential of synthesizing kinematically plausible motions. Notably, Jain *et al*. [19] model human motions with a spatial-temporal graph that considers the skeletal structure but not in an analytic form.

Our work is also related to research efforts on "vision as inverse graphics". Differentiable rendering layers are incorporated into deep neural networks to disentangle imaging factors of rigid objects, such as 3D shape, camera, normal map, lighting and materials [41, 30, 37, 24]. Wu et al. [39] further incorporated a differentiable physics simulator [8] to disentangle physical properties of multiple rigid objects. Our network disentangles the hierarchical rotations of articulated skeletons through a differentiable forward kinematics layer.

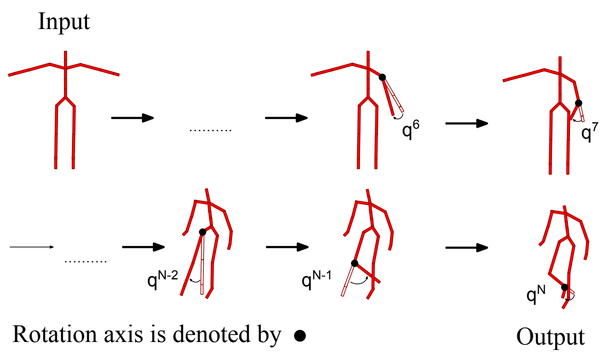

Figure 2: Forward kinematics from T-pose skeleton. Starting from the input skeleton, the forward kinematics layer rotates bones to achieve the desired output configuration.

3. Background

We first introduce some concepts in robotics and computer animation essential for building our model.

3.1. Forward kinematics

Forward kinematics (FK) refers to the process of computing the positions of skeleton joints, also known as *end-effectors*, in 3D space given the joint rotations and initial positions. FK is performed by recursively rotating the joints of an input skeleton tree starting from the root joint and ending in the leaf joints, and is defined by:

$$p^n = p^{parent(n)} + R^n \bar{s}^n,$$

where $p^n \in \mathbb{R}^3$ is the updated 3D position of the n-th joint and $p^{parent(n)} \in \mathbb{R}^3$ is the current position of its parent. $R^n \in \mathbb{SO}(3)$ is the rotation of the n-th joint with respect to its parent. $\bar{s}^n \in \mathbb{R}^3$ is the 3D offset of the n-th joint relative to its parent in the input skeleton, and is defined by:

$$\bar{s}^n = \bar{p}^n - \bar{p}^{parent(n)},$$

Note that $\bar{p}^n$ and $\bar{p}^{parent(n)}$ refer to joint positions in the input *T-pose skeleton* as depicted in Figure 2.

3.2. Inverse kinematics

While FK refers to computing the 3D joint locations by recursively applying joint rotations, inverse kinematics (IK) is the reverse process of computing joint rotations $R^{1:N}$ that ensure specific joints are placed at the desired target locations $p^{1:N}$ starting from initial positions $p_0^{1:N}$. Thus, IK is defined by:

$$R^{1:N} = \text{IK}(p^{1:N}, p_0^{1:N}).$$

IK is inherently an ill-posed problem. Target configuration of joint locations can be fulfilled by multiple joint rotations or no joint rotations. Classic IK solutions often resort to iterative optimization by calculating the inverse Jacobian of the highly nonlinear FK function numerically or analytically.

4. Method

In this section, we present our proposed method for unsupervised motion retargetting. There are two main components: (i) the neural kinematic network architecture for skeleton conditioned motion synthesis, and (ii) the adversarial cycle consistency training for unsupervised motion retargetting. We next describe these components in detail.

4.1. Neural kinematic networks

Our neural kinematic networks for motion synthesis component is built to strictly manipulate a target skeleton, which we refer as *condition skeleton*, into performing a given motion sequence performed by another source skeleton through a *Forward Kinematics* layer.

In our setup, the input motion data $x_{1:T}$ is decomposed into $p_{1:T}$ and $v_{1:T}$, where for each time t, $p_t \in \mathbb{R}^{3N}$ represents the local *xyz*-configuration of the skeleton's pose with respect to its root joint (i.e., hip joint), and $v_t \in \mathbb{R}^4$ represents the global motion of the skeleton's root joint (i.e., *x*,*y*,*z*-velocities and rotation with respect to the axis perpendicular to the ground). Given the condition skeleton, the motion synthesis module outputs the rotations, R_t^n, that are then applied to each joint n at time t, as well as the global motion parameters.

4.1.1 Forward kinematics layer

At the core of our neural kinematic networks for motion synthesis component lies the *Forward Kinematics layer* (Figure 2) which is designed to take in 3D rotations for each joint n at time t parameterized by *unit quaternions* $q_t^n \in \mathbb{R}^4$, and apply them to a skeleton bone configuration $\bar{s}^n$. A *quaternion* extends a *complex number* in the form $r + x\mathbb{i} + y\mathbb{j} + z\mathbb{k}$ and is used to rotate objects in 3 dimensional space, where r, x, y, and z are real numbers and $\mathbb{i}$, $\mathbb{j}$, $\mathbb{k}$ are quaternion units. The rotation matrix corresponding to an input quaternion is calculated as follows:

$$R_t^n = \begin{pmatrix} 1-2({q_{tj}^n}^2+{q_{tk}^n}^2) & 2(q_{ti}^n q_{tj}^n + q_{tk}^n q_{tr}^n) & 2(q_{ti}^n q_{tk}^n - q_{tj}^n q_{tr}^n) \\ 2(q_{ti}^n q_{tj}^n - q_{tk}^n q_{tr}^n) & 1-2({q_{ti}^n}^2+{q_{tk}^n}^2) & 2(q_{tj}^n q_{tk}^n + q_{ti}^n q_{tr}^n) \\ 2(q_{ti}^n q_{tk}^n + q_{tj}^n q_{tr}^n) & 2(q_{tj}^n q_{tk}^n - q_{ti}^n q_{tr}^n) & 1-2({q_{ti}^n}^2+{q_{tj}^n}^2) \end{pmatrix} \tag{1}$$

Given the rotation matrices $R_t^n \in \mathbb{SO}(3)$ for each joint, the FK layer updates the joint positions of the condition skeleton by applying these rotations in a recursive manner as described in Section 3.1 and shown in Figure 2,

$$p_t^{1:N} = \text{FK}(q_t^{1:N}, \bar{s}).$$

The FK layer serves as a tool for mapping the joint rotations to actual joint locations and thus helps our network to focus on learning skeleton independent motion features, i.e., joint rotations.

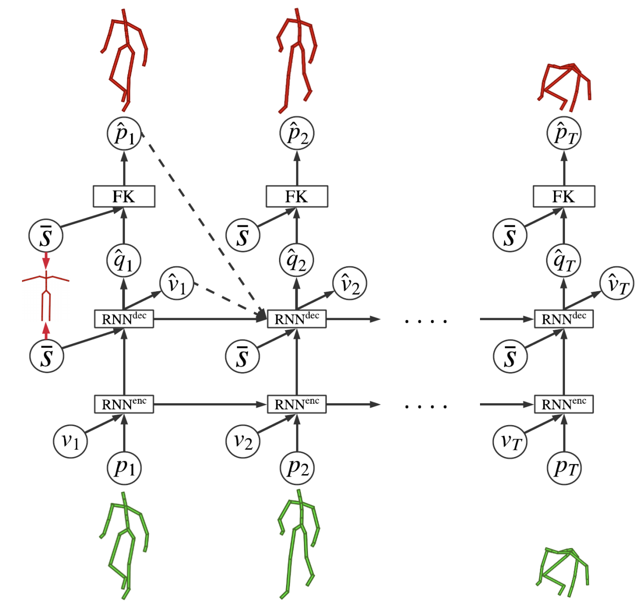

Figure 3: Neural kinematic networks for motion synthesis.

4.1.2 Online motion synthesis

Our proposed neural kinematic networks architecture for online motion synthesis is shown in Figure 3. Taking advantage of the temporal coherency in motion sequences, we synthesize the current motion step at time t by conditioning on previous steps through an RNN hidden representation.

The current step in the input motion is encoded by:

$$h_t^{\text{enc}} = \text{RNN}^{\text{enc}}(x_t, h_{t-1}^{\text{enc}}; W^{enc}), \tag{2}$$

where $\text{RNN}^{\text{enc}}(.,.)$ is an encoder RNN, h_t^{enc} is the encoding of the input motion up to time t, and $x_t = [p_t, v_t]$ is the current input. The encoded feature is then fed to a decoder RNN to perform skeleton conditioned motion synthesis by:

$$h_t^{\text{dec}} = \text{RNN}^{\text{dec}}(\hat{x}_{t-1}, h_t^{\text{enc}}, \bar{s}, h_{t-1}^{\text{dec}}; W^{dec}), \tag{3}$$

$$\hat{q}_t = \frac{W^{p^T} h_t^{\text{dec}}}{\|W^{p^T} h_t^{\text{dec}}\|}, \tag{4}$$

$$\hat{p}_t = \text{FK}(\hat{q}_t, \bar{s}), \tag{5}$$

$$\hat{v}_t = W^{v^T} h_t^{\text{dec}}, \tag{6}$$

$$\hat{x}_t = [\hat{p}_t, \hat{v}_t]\,. \tag{7}$$

where h_t^{dec} is the hidden representation of decoder RNN, $\hat{x}_t$ is the synthesized motion at time step t for the condition skeleton $\bar{s}$. The unit vector $\hat{q}_t \in \mathbb{R}^{4N}$ denotes the rotations — which can be interpreted as actions — to be applied to the condition skeleton through the FK layer. The outputs $\hat{p}_t$ and $\hat{v}_t$ are the estimated local and global motion of the condition skeleton. Finally, W^{enc}, W^{dec}, $W^{v} \in \mathbb{R}^{d\times 4}$ and $W^{p} \in \mathbb{R}^{d\times 4N}$ are learnable parameters.

When the condition skeleton is different from the skeleton where the input motion lives, the decoder is meant to generate the rotations of a new character to achieve motion retargetting. Please note that in the rest of the paper, we use superscripts A or B to refer to the identity of the skeleton we are retargetting motion from and into.

4.2. Adversarial cycle training for unsupervised motion retargetting

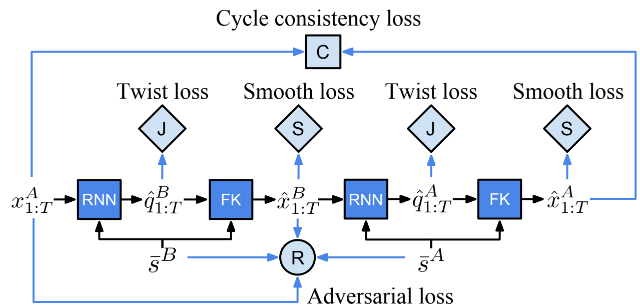

Figure 4: Adversarial cycle consistency framework.

In Section 4.1, we describe a method for skeleton conditioned motion synthesis based on a forward kinematics layer embedded within the network architecture. However, training such a network for motion retargetting is challenging as it is very expensive to collect paired motion data x_t^A and x_t^B where the same motion is performed by two different skeletons. Note that collecting such data requires using iterative optimization based IK methods in addition to human hand-tuning of the retargetted motion.

We propose a training paradigm based on the cycle consistency principle [43] and adversarial training [12] for unsupervised motion retargetting (Figure 4). Let f be our neural kinematic network, and let the superscripts define skeleton identity. Given an input motion sequence from skeleton A, we first retarget the input motion to skeleton B and back to A as follows:

$$\hat{x}_{1:T}^B = f(x_{1:T}^A, \bar{s}^B), \tag{8}$$

$$\hat{x}_{1:T}^A = f(\hat{x}_{1:T}^B, \bar{s}^A), \tag{9}$$

where $\hat{x}_{1:T}^B$ and $\hat{x}_{1:T}^A$ are synthesized motions for skeletons B and A, respectively. Therefore, we define four loss terms: adversarial loss on $\hat{x}_{1:T}^B$, cycle consistency loss on $\hat{x}_{1:T}^A$, twist loss on rotations $\hat{q}_{1:T}^A$ and $\hat{q}_{1:T}^B$, and smoothing loss on $\hat{v}_t^A$ and $\hat{v}_t^B$, so our full training objective is defined by:

$$\min_f \max_d \quad C(\hat{x}_{1:T}^A, x_{1:T}^A) + R(\hat{x}_{1:T}^B, x_{1:T}^A) + \lambda\, J(\hat{q}_{1:T}^B, \hat{q}_{1:T}^A) + \omega\, S(\hat{v}_{1:T}^B, \hat{v}_{1:T}^A), \tag{10}$$

where C is the cycle consistency loss, R the adversarial loss, J the joint twist loss, and S the velocity smoothing loss.

Adversarial loss. The input motion $x_{1:T}^A = [p_{1:T}^A, v_{1:T}^A]$, the synthesized motion $\hat{x}_{1:T}^B = [\hat{p}_{1:T}^B, \hat{v}_{1:T}^B]$, and their respective skeleton are fed to a discriminator network g that computes a *realism score* for *real* and *fake* motion sequences:

$$r^A = g(p^A_{2:T} - p^A_{1:T-1}, v^A_{1:T-1}, \bar{s}^A), \tag{11}$$

$$r^B = g(\hat{p}^B_{2:T} - \hat{p}^B_{1:T-1}, \hat{v}^B_{1:T-1}, \bar{s}^B), \tag{12}$$

where r^A is the output of the discriminator given real data, and r^B is the output of the discriminator given the fake data (i.e., the motion retargetted by our network into skeleton B). The inputs to the discriminator $p^A_{2:T} - p^A_{1:T-1}$ and $p^B_{2:T} - p^B_{1:T-1}$ are the local motion difference between two adjacent time steps, and $\bar{s}^A$ and $\bar{s}^B$ denote the input and target skeletons A and B, respectively. During training, we randomly sample $\bar{s}^B$ from all the available skeletons, thus, it is possible for skeleton B to be the same as skeleton A. In case skeleton B is the same as skeleton A, $\hat{x}^B_{1:T} = \hat{x}^A_{1:T}$, we switch between adversarial and square loss as follows:

$$R(\hat{x}^B_{1:T}, x^A_{1:T}) = \begin{cases} \|\hat{x}^B_{1:T} - x^A_{1:T}\|^2_2, & \text{if } B = A \\ \log r^A + \beta \log(1 - r^B), & \text{otherwise.} \end{cases}, \tag{13}$$

When B and A are not the same, we rely on the motion distributions learned by g as a training signal. By observing other motion sequences performed by skeleton B, the discriminator network learns to identify motion behaviors of skeleton B. The generator (encoder and decoder RNNs) uses this as indirect guidance to learn how the motion should be retargetted to B and thus fool the discriminator. When applying the adversarial loss, we use a balancing term β to regulate the strength of the discriminator signal when optimizing f to fool g. We use $\beta = 0.001$ in our experiments.

Cycle consistency loss. The cycle consistency loss C optimizes the following objective:

$$C(\hat{x}^A_{1:T}, x^A_{1:T}) = \|x^A_{1:T} - \hat{x}^A_{1:T}\|^2_2. \tag{14}$$

Equation 14 encourages f to be able to take its own retargetted motion and map it back to the original motion source effectively achieving cycle consistency.

Twist loss. By optimizing the first two terms in Equation 10, our network discovers the necessary rotations to move the input skeleton end-effectors to the required positions for motion retargetting. However, this does not prevent potential excessive bone twisting since *xyz*-coordinates can be perfectly predicted regardless of how many times we rotate a bone around its own axis. Thus, the third term in our objective constrains the bone rotations around its own axis.

$$J(\hat{q}^B_{1:T}, \hat{q}^A_{1:T}) = \|\max(\mathbf{0}, |euler_y(\hat{q}^B_{1:T})| - \alpha)\|^2_2 + \|\max(\mathbf{0}, |euler_y(\hat{q}^A_{1:T})| - \alpha)\|^2_2, \tag{15}$$

where $euler_y(.)$ converts the quaternion outputs of our network into rotation angles around the standard *xyz*-axes and the subscript y means to select the rotation angle around the plane parallel to the bone (i.e. y-axis). Therefore, any bone rotation exceeding α degrees in either negative or positive direction is penalized in our objective function. We use $\alpha = 100°$, and $\lambda = 10$ in our experiments.

Smoothing loss. Finally, the first two terms in our objective function treat global motion at each time step independently. However, global motion in consecutive timesteps are highly dependent on each other, that is, global motion in the next timestep should change only slightly with respect to the previous global motion. We constraint the global motion by:

$$S(\hat{v}^B_{1:T}, \hat{v}^A_{1:T}) = \|\hat{v}^B_{2:T} - \hat{v}^B_{1:T-1}\|^2_2 + \|\hat{v}^A_{2:T} - \hat{v}^A_{1:T-1}\|^2_2, \tag{16}$$

We use $\omega = 0.01$ in our experiments.

5. Experiments

Dataset. We evaluate our method on the Mixamo dataset [1] which contains approximately 2400 unique motion sequences for 71 characters (i.e., skeletons). For training, we collected non-overlapping motion sequences for 7 characters (AJ, Big Vegas, Goblin Shareyko, Kaya, Malcolm, Peasant Man, and Warrok Kurniawan) which in total results in 1646 training sequences at 30 frames per second.

For testing, we collected motion sequences for 6 characters (Malcolm, Mutant, Warrok Kurniawan, Sporty Granny, Claire, and Liam) and perform retargetting in four scenarios:

- Input motion is seen during training, and the target character is also seen during training but the target motion sequence is not.
- Input motion is seen during training but the target character is never seen during training.
- Input motion is not seen during training but the target character is seen during training.
- Neither the input motion nor the target character are seen during training.

Note that we also collected the ground truth retargetted motions of testing sequences for quantitative evaluation purposes only. While we discuss our main findings below, detailed results and analysis of each scenario and character can be found in the supplementary material as well as details of how to acquire the exact training and testing data.

Data preprocessing. Each motion sequence is preprocessed by separating into local and global motion, similar to [16]. For local motion, we remove the global displacement (i.e., the motion of the root joint), and rotation around the axis vertical to the ground. Global motion consists of the velocity of the root in the *x*, *y*, and *z* directions, and an additional value representing the rotation around the axis perpendicular to the ground. For training, and testing we use the following 22 joints: Root, Spine, Spine1, Spine2, Neck, Head, LeftUpLeg, LeftLeg, LeftFoot, LeftToeBase, RightUpLeg, RightLeg, RightFoot, RightToeBase, LeftShoulder, LeftArm, LeftForeArm, LeftHand, RightShoulder, RightArm, RightForeArm, and RightHand.

Baseline methods. While there have been several optimization based approaches for the IK problem, most of these expect the user to provide motion specific constraints or goals. Since this is not feasible to do at a large scale, we instead show comparisons to learning based baseline methods that aim to identify such constraints automatically. The first baseline is an RNN architecture without the FK layer that directly outputs *xyz*-coordinates for the local motion, and the global motion output is the same as ours. Second, we use an MLP architecture that lacks recurrent connections, and directly outputs the *xyz*-coordinates for the local motion, and the same global motion output as our method. We also train both baselines with our adversarial cycle consistency objective. Finally, we include another baseline that directly copies the per-joint rotation and the global motion of the input motion into the target skeleton.

Training and evaluation. We train our method and baselines by randomly sampling 2-second motion clips (60 frames) from the training sequences, and testing on motion clips of 4 seconds (120 frames) from the test sequences. We initialized the quaternion outputs of the decoder RNN to be close to the identity rotation (i.e., close to zero rotation). For training the discriminator network, we sample random motion sequences being performed by the same skeleton into which the motion synthesis network is retargetting motion. Details of the network architecture and hyperparameters can be found in the supplementary material. We perform two types of evaluations: 1) We evaluate the overall quality of the motion retargetting using a target character normalized Mean Square Error (MSE) on the estimated joint locations through time (i.e., *xyz*-coordinates after combining local and global motion together). 2) We compare end-effector locations through time against the ground-truth. 3) We show qualitative results by rendering the animated 3D characters using the outputs of our network.

5.1. Online Motion Retargetting From Character

In this section we evaluate our method on the task of online motion retargetting, i.e., retargetting motion from one character to a target character as new motion frames are received. We present an ablation study to demonstrate the benefits of the different components of our method, and also compare against the previously described baselines. In Table 1, we report the average MSE of the retargetted motion when our network is trained with different objectives: 1) Our skeleton conditioned motion synthesis network (Section 4.1) trained with the autoencoder objective (i.e., input reconstruction) and the bone twisting constraint only. 2) Our network trained with the cycle consistency objective without adversarial training. Specifically, the "otherwise" branch in Equation 13, returns 0. 3) Our network trained with our full adversarial cycle consistency objective function which requires examples of motions performed by skeleton B but not paired with any motions used as inputs during training.

Method	MSE
Ours: Autoencoder Objective	10.25
Ours: Cycle Consistency Objective	8.51
Ours: Adversarial Cycle Consistency Objective	**7.10**
Baseline: Conditional RNN	13.65
Baseline: Conditional RNN + Adv. Cycle Consistency	26.93
Baseline: Conditional MLP	17.02
Baseline: Conditional MLP + Adv. Cycle Consistency	16.96
Baseline: Copy input quaternions and velocities	9.00

Table 1: Quantitative evaluation of online motion retargetting using mean square error (MSE).

As it can be seen in Table 1, simply using the proposed FK layer within RNNs and training with an autoencoder objective (Ours: Autoencoder Objective), outperforms all standard neural network based baselines. One explanation is that it is highly probable for the baselines to ignore the bone lengths of the target skeleton, and learn a motion representation that is dependent on the input skeleton. The inability to disentangle motion properties from the input skeleton is more evident after training with our adversarial cycle consistency objective which still results in poor performance. The inputs to the discriminator network are velocities, that is, local motion difference between adjacent time steps and global motion. While this input contains information about the shift in joint locations through time, it does not capture any information about the spatial structure. As a result, optimizing the baselines to fool the discriminator network, does not impose bone length constraints. Furthermore, encouraging velocities to be similar to the real data causes further bone length degradation (i.e., excessive stretching or shrinking) in absence of such constraints. On the other hand, our architecture is designed to learn a skeleton invariant motion representation that can be directly transferred to the target skeleton through the FK layer.

The performance of our method improves when training our motion synthesis network with the proposed objectives for cycle consistency and adversarial cycle consistency. While training with the autoencoder objective results in reasonable performance, often the network tries to match end-effector locations but does not fully capture the properties of the input motion. For example, when an input motion of a small character raising its hands is retargetted to a very tall character, the tall character is likely not able to raise its hands but only point in the same direction as the input motion. Our network improves when trained with the cycle consistency objective alone. In the example of motion retargetting from a small to a tall character, cycle consistency loss prevents the

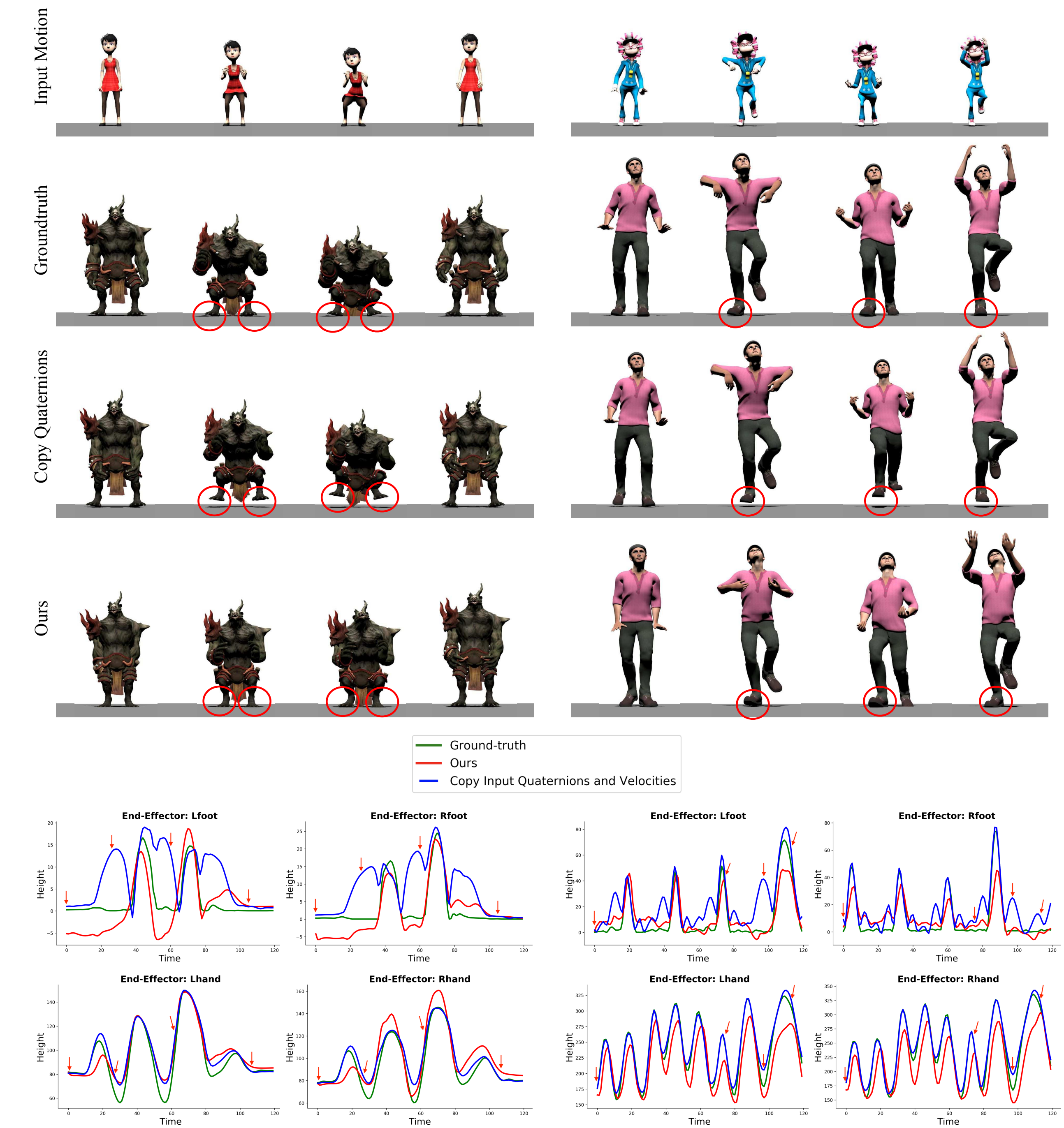

Figure 5: Qualitative evaluation. We present a motion retargetting example of our method against the best baseline. Motion is retargetted from character Claire into Warrok Kurniawan (left) and Sporty Granny to Malcolm (right). Plots illustrating the left/right feet and hand end-effectors' height comparing against the groundtruth are shown at the bottom. Arrows in the plots determine the time steps of the shown animation frames. Please visit `goo.gl/mDTvem` for animated videos.

tall character from directly matching end-effector positions of the small character as retargetting back to the small character would have resulted in stretching the limbs in the small character. The cycle consistency encourages the network to better learn the high level features of the input motion.

Finally, our method performs the best when our objective imposes both cycle consistency and realism via the full adversarial cycle consistency objective. The adversarial training helps the network to produce motions that cannot be distinguished from realistic motions of the target character.

The baseline "Copy input quaternions and velocities" works better than the neural network baselines due to the fol-

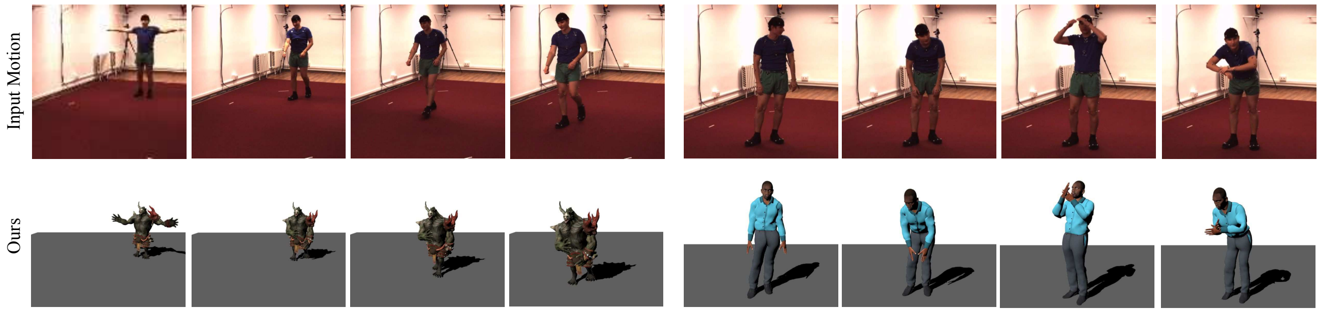

Figure 6: Qualitative evaluation on human videos. Motion is retargetted from estimated 3D pose from the Human 3.6M dataset into Mixamo 3D characters using the estimated 3D pose from [26]. Please visit goo.gl/mDTvem for animated videos.

lowing reasons: 1) Copying per-joint rotations of the input and performing forward kinematics already respects the target skeleton bone lengths, and 2) copying the velocities (i.e., global motion) avoids drifting that prediction models may suffer from. However, when retargetting motions between characters with significant skeleton difference, this baseline is prone to artifacts such as foot floating (see Figure 5). This baseline is also not scalable to cases where different skeleton limits or topological structures are considered.

In Figure 5, we show qualitative results where we animate target characters using the output of our network using Blender [5], a character animation software. For all the joints that are not modeled by our network (e.g., the fingers), we simply directly copy the joint rotations from the input motion if the corresponding joint names match in the input and the target skeleton, otherwise we leave them fixed.

5.2. Online Motion Retargetting from Human Video

In this section we present motion retargetting from human video input into characters using the model trained from the Mixamo data only. We use the Human 3.6M videos as input, the algorithm from [26] to estimate the 3D pose of each frame, and the ground truth 3D skeleton root displacement (3D pose estimation algorithms usually assume the person is centered). The videos are subsampled to 25 FPS, and the estimated 3D poses are processed similar to our previous experiment. The algorithm in [26] only outputs 17 joints compared to the 22 joints needed by our network. Therefore, we manually map the 17 joints to 22 by duplicating the following joint positions in Human 3.6M to corresponding Mixamo joints: Spine into Spine and Spine1, LeftShoulder into LeftShoulder and LeftArm, RightShoulder into RightShoulder and RightArm , LeftFoot into LeftFoot and LeftToeBase, RightFoot into RightFoot and RightToeBase. Note that this mapping will create bones of zero length during test time. Thus, our network essentially only sees 17 joints but uses 22 joints as input. During visualization, we do not rotate joints that are not predicted by our network (i.e., fingers). As shown in Figure 6, our network is able to generalize to never-seen skeletons and motions estimated from monocular human videos. More video results and analyses are included in supplementary materials.

6. Conclusion and Future Work

We have presented a neural kinematic network with an adversarial cycle consistency training objective for motion retargetting. Our network only observes a sequence of xyz-coordinates of joints from existing animations, motion capture or 3D pose estimates of monocular human videos, and transfers the motion to a target humanoid character without risking skeleton deformations that occur in the baselines. The success of our method attributes to the following factors: 1) The proposed *Forward Kinematics layer* helps to discover joint rotations of target skeleton that are independent of the input skeleton. 2) The cycle consistency of the retargetting objective prevents regressing to the end-effector positions of the input motion. 3) The adversarial objective helps the network to produce realistic motions. 4) The bone twist loss constrains the solution space of *Inverse Kinematics* and prevents bone twisting in the retargetted motion.

Our current method has limitations. First, we perform retargetting on a fixed number of joints. Handling a variable number of joints is challenging as the retargetting algorithm is expected to automatically select end-effectors of interest when transferring motions. Second, we assume the environment in which the target character is being animated lacks physical constraints such as gravity. Future work will include equipping the network with physics simulators to generate more natural and physically plausible movements of the target characters with different muscle/bone mass distributions. Third, the input to our method still requires 3D information (xyz-coordinates of joints). Future work will also include training our network end-to-end by using monocular videos as input. That may require the algorithm to learn view-invariant features.

Acknowledgments. This work was supported in part by Gift from Adobe Research, NSF CAREER IIS-1453651, and Rackham Merit Fellowship. We would like to thank Aaron Hertzmann for helpful discussions.

References

[1] Adobe's Mixamo. https://www.mixamo.com. Accessed: 2017-09-28. 5

[2] K. Ayusawa and E. Yoshida. Motion retargeting for humanoid robots based on simultaneous morphing parameter identification and motion optimization. *IEEE Trans. on Robotics*, 33(6), 2017. to appear. 1

[3] J. A. D. Bagnell. An invitation to imitation. Technical Report CMU-RI-TR-15-08, Pittsburgh, PA, March 2015. 1

[4] G. Bin Hammam, P. M. Wensing, B. Dariush, and D. E. Orin. Kinodynamically consistent motion retargeting for humanoids. *International Journal of Humanoid Robotics*, 12(04):1550017, 2015. 1

[5] Blender Online Community. *Blender - a 3D modelling and rendering package*. Blender Foundation, Blender Institute, Amsterdam, 2017. 8

[6] M. Brand and A. Hertzmann. Style machines. In *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, pages 183–192. ACM Press/Addison-Wesley Publishing Co., 2000. 2

[7] J. Bütepage, M. J. Black, D. Kragic, and H. Kjellström. Deep representation learning for human motion prediction and classification. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, page 2017, 2017. 2

[8] M. B. Chang, T. Ullman, A. Torralba, and J. B. Tenenbaum. A compositional object-based approach to learning physical dynamics. In *ICLR*, 2017. 2

[9] K.-J. Choi and H.-S. Ko. On-line motion retargetting. In *Computer Graphics and Applications, 1999. Proceedings. Seventh Pacific Conference on*, pages 32–42. IEEE, 1999. 2

[10] K. Fragkiadaki, S. Levine, P. Felsen, and J. Malik. Recurrent network models for human dynamics. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 4346–4354, 2015. 2

[11] M. Gleicher. Retargetting motion to new characters. In *Proceedings of the 25th annual conference on Computer graphics and interactive techniques*, pages 33–42. ACM, 1998. 2

[12] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014. 4

[13] A. Graves, A.-r. Mohamed, and G. Hinton. Speech recognition with deep recurrent neural networks. In *Acoustics, speech and signal processing (icassp), 2013 ieee international conference on*, pages 6645–6649. IEEE, 2013. 2

[14] K. Grochow, S. L. Martin, A. Hertzmann, and Z. Popović. Style-based inverse kinematics. In *ACM transactions on graphics (TOG)*, volume 23, pages 522–531. ACM, 2004. 2

[15] J. Ho and S. Ermon. Generative adversarial imitation learning. In *Advances in Neural Information Processing Systems*, pages 4565–4573, 2016. 1

[16] D. Holden, J. Saito, and T. Komura. A deep learning framework for character motion synthesis and editing. *ACM Transactions on Graphics (TOG)*, 35(4):138, 2016. 2, 5

[17] E. Hsu, K. Pulli, and J. Popović. Style translation for human motion. In *ACM Transactions on Graphics (TOG)*, volume 24, pages 1082–1089. ACM, 2005. 2

[18] C. Ionescu, D. Papava, V. Olaru, and C. Sminchisescu. Human3.6M: Large scale datasets and predictive methods for 3D human sensing in natural environments. *PAMI*, 36(7):1325–1339, 2014. 2

[19] A. Jain, A. R. Zamir, S. Savarese, and A. Saxena. Structural-rnn: Deep learning on spatio-temporal graphs. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5308–5317, 2016. 2

[20] M. Johnson, M. Schuster, Q. V. Le, M. Krikun, Y. Wu, Z. Chen, N. Thorat, F. Viégas, M. Wattenberg, G. Corrado, M. Hughes, and J. Dean. Google's multilingual neural machine translation system: Enabling zero-shot translation. *Transactions of the Association of Computational Linguistics*, 5:339–351, 2017. 2

[21] L. Kovar and M. Gleicher. Automated extraction and parameterization of motions in large data sets. In *ACM Transactions on Graphics (ToG)*, volume 23, pages 559–568. ACM, 2004. 2

[22] J. Lee and S. Y. Shin. A hierarchical approach to interactive motion editing for human-like figures. In *Proceedings of the 26th annual conference on Computer graphics and interactive techniques*, pages 39–48. ACM Press/Addison-Wesley Publishing Co., 1999. 2

[23] Z. Li, Y. Zhou, S. Xiao, C. He, and H. Li. Auto-conditioned lstm network for extended complex human motion synthesis. In *ICLR*, 2018. 2

[24] G. Liu, D. Ceylan, E. Yumer, J. Yang, and J.-M. Lien. Material editing using a physically based rendering network. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 2280–2288. IEEE, 2017. 2

[25] J. Martinez, M. J. Black, and J. Romero. On human motion prediction using recurrent neural networks. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4674–4683. IEEE, 2017. 2

[26] J. Martinez, R. Hossain, J. Romero, and J. J. Little. A simple yet effective baseline for 3d human pose. *ICCV*, 2017. 8

[27] D. Mehta, S. Sridhar, O. Sotnychenko, H. Rhodin, M. Shafiei, H.-P. Seidel, W. Xu, D. Casas, and C. Theobalt. Vnect: Real-time 3d human pose estimation with a single rgb camera. *ACM Transactions on Graphics (TOG)*, 36(4):44, 2017. 1

[28] J. Merel, Y. Tassa, S. Srinivasan, J. Lemmon, Z. Wang, G. Wayne, and N. Heess. Learning human behaviors from motion capture by adversarial imitation. *arXiv preprint arXiv:1707.02201*, 2017. 1

[29] J. Min, H. Liu, and J. Chai. Synthesis and editing of personalized stylistic human motion. In *Proceedings of the 2010 ACM SIGGRAPH symposium on Interactive 3D Graphics and Games*, pages 39–46. ACM, 2010. 2

[30] D. J. Rezende, S. A. Eslami, S. Mohamed, P. Battaglia, M. Jaderberg, and N. Heess. Unsupervised learning of 3d structure from images. In *Advances In Neural Information Processing Systems*, pages 4996–5004, 2016. 2

[31] C. F. Rose III, P.-P. J. Sloan, and M. F. Cohen. Artist-directed inverse-kinematics using radial basis function interpolation. In *Computer Graphics Forum*, volume 20, pages 239–250. Wiley Online Library, 2001. 2

[32] S. Schaal. Is imitation learning the route to humanoid robots? *Trends in cognitive sciences*, 3(6):233–242, 1999. 1

[33] P. Sermanet, C. Lynch, J. Hsu, and S. Levine. Time-contrastive networks: Self-supervised learning from multi-view observation. *arXiv preprint arXiv:1704.06888*, 2017. 1

[34] A. Shon, K. Grochow, A. Hertzmann, and R. P. Rao. Learning shared latent structure for image synthesis and robotic imitation. In Y. Weiss, P. B. Schölkopf, and J. C. Platt, editors, *Advances in Neural Information Processing Systems 18*, pages 1233–1240. MIT Press, 2006. 1

[35] S. Tak and H.-S. Ko. A physically-based motion retargeting filter. *ACM Transactions on Graphics (TOG)*, 24(1):98–117, 2005. 2

[36] G. W. Taylor, G. E. Hinton, and S. T. Roweis. Modeling human motion using binary latent variables. In *Advances in neural information processing systems*, pages 1345–1352, 2007. 2

[37] S. Tulsiani, T. Zhou, A. A. Efros, and J. Malik. Multi-view supervision for single-view reconstruction via differentiable ray consistency. In *CVPR*, volume 1, page 3, 2017. 2

[38] J. M. Wang, D. J. Fleet, and A. Hertzmann. Gaussian process dynamical models for human motion. *IEEE transactions on pattern analysis and machine intelligence*, 30(2):283–298, 2008. 2

[39] J. Wu, E. Lu, P. Kohli, W. T. Freeman, and J. B. Tenenbaum. Learning to see physics via visual de-animation. In *Advances in Neural Information Processing Systems*, 2017. 2

[40] S. Xia, C. Wang, J. Chai, and J. Hodgins. Realtime style transfer for unlabeled heterogeneous human motion. *ACM Transactions on Graphics (TOG)*, 34(4):119, 2015. 2

[41] X. Yan, J. Yang, E. Yumer, Y. Guo, and H. Lee. Perspective transformer nets: Learning single-view 3d object reconstruction without 3d supervision. In *Advances in Neural Information Processing Systems*, pages 1696–1704, 2016. 2

[42] M. E. Yumer and N. J. Mitra. Spectral style transfer for human motion between independent actions. *ACM Transactions on Graphics (TOG)*, 35(4):137, 2016. 2

[43] T. Zhou, P. Krahenbuhl, M. Aubry, Q. Huang, and A. A. Efros. Learning dense correspondence via 3d-guided cycle consistency. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 117–126, 2016. 4

[44] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *ICCV*, 2017. 2