Appendix

A. Quantitative Evaluation per Motion Retargetting Scenario, and Analysis

In this section, we present quantitative evaluation for the different motion retargetting scenarios mentioned in the main text. We then present findings showing how our method significantly outperforms the best performing baseline (copy quaternions and velocities).

In Table 2, we show results of retargetting motion previously seen during training into two target scenarios: 1) Character has been seen during training, but not performing the motion used as input (left). 2) Character has never been seen during training (right). In Table 3, we show results of retargetting motion never seen during training into two target scenarios: 1) Character that has been seen during training (left). 2) Character has never been seen during training (right).

Known Motion / Known Character	
Method	MSE
Ours: Autoencoder Objective	8.61
Ours: Cycle Consistency Objective	5.68
Ours: Adversarial Cycle Consistency Objective	5.35
Ground-truth joint location variance through time: 4.8	

Known Motion / New Character	
Method	MSE
Ours: Autoencoder Objective	2.16
Ours: Cycle Consistency Objective	1.55
Ours: Adversarial Cycle Consistency Objective	1.35
Ground-truth joint location variance through time: 1.5	

Table 2: Quantitative evaluation of online motion retargetting using mean square error (MSE). Case study: Known motion / known character (left), and known motion / new character (right).

New Motion / Known Character	
Method	MSE
Ours: Autoencoder Objective	6.55
Ours: Cycle Consistency Objective	4.38
Ours: Adversarial Cycle Consistency Objective	4.39
Ground-truth joint location variance through time: 3.6	

New Motion / New Character	
Method	MSE
Ours: Autoencoder Objective	24.16
Ours: Cycle Consistency Objective	23.49
Ours: Adversarial Cycle Consistency Objective	18.02
Ground-truth joint location variance through time: 11.6	

Table 3: Quantitative evaluation of online motion retargetting using mean square error (MSE). Case study: New motion / known character (left), and new motion / new character (right).

From these results, we can observe the benefits of our full adversarial cycle training vs only using cycle training. In both input motion scenarios — seen during training and never seen during training — retargetting into a never before seen target skeleton results in overall performance improvement. For known input motions, retargetting into a new character results in a performance improvement of $\mathbf{12.9}\%$, while retargetting into a known character results in a performance improvement of $\mathbf{5.8}\%$. Additionally, for new input motions, retargetting into a new character results in a performance improvement of $\mathbf{23.3}\%$, while retargetting into a known character results in a similar performance. In the previous analysis, we can clearly see that learning character behaviors in the training data results in an overall performance boost when the target character has been seen before. Most importantly, learning skeleton conditioned behaviors results in much better generalization to new characters compared to training with cycle alone. However, we can also see that some scenarios reflect larger errors than others. To explain this phenomenon, we measure the movement in the ground-truth motion sequences by computing the average character height normalized joint location variance through time presented at the bottom of each table. This result shows that the more movement there is in the ground-truth sequence, the larger the MSE becomes, thus the larger errors seen on some of the test scenarios previously presented.

Next, we quantitative evaluate our method and the best performing baseline (copy input quaternions and velocities) by separating testing examples into bins based on average movement through time observed in the ground-truth target motion. This evaluation gives us a clearer insight on how much input movement in space each method can handle during retargetting, and how our method is outperforming the best baseline.

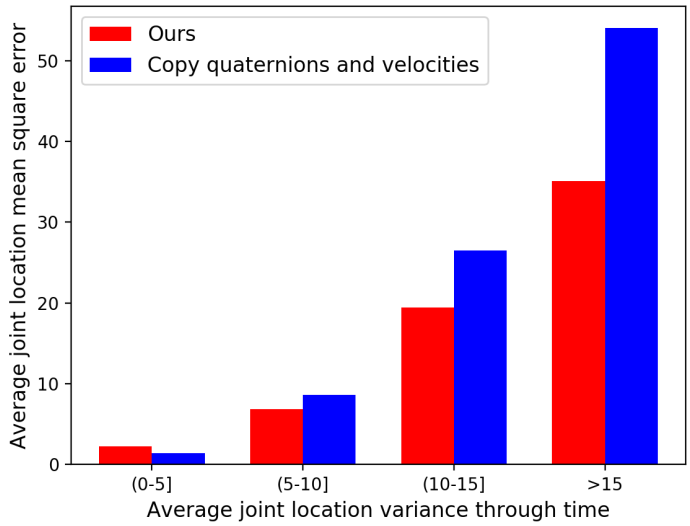

Figure 7: Quantitative evaluation based on movement through time. The vertical axis denotes mean square error, and the horizontal axis denotes the *xyz*-coordinate average variance through time observed in the ground-truth. The average joint location variance is normalized by character height.

In Figure 7, we can observe that our method outperforms the baseline as the movement in the evaluation videos increase. Our method substantially outperforms the baseline when the average joint location variance is larger than 5, however, the baseline marginally outperforms our method when the average joint location variance is less than or equals to 5. This result shows that by simply copying the input motion into the target character we cannot guarantee motion retargetting that follows the correct motion in the target character. Therefore, we have to rely on a model that has understanding of the target character and input motion relationships for synthesizing skeleton conditioned motion.

B. Denoising 3D Pose Estimation by Motion Retargetting

In this section, we show the denoising power of our model on estimated 3D poses. Most 3D pose estimation algorithms do it in a per-frame manner completely ignoring temporal correlations among the estimated poses in videos at every time step. We use our method trained on the Mixamo dataset to retarget the 3D pose estimated by [16] back into the input motion skeleton (Human 3.6M skeleton) to demonstrate denoising effects on the input pose sequence. We compute the Human 3.6M skeleton from the first frame pose in the sequence `WalkTogether 1.60457274` performed by `Subject 9`. We evaluate for denoising by plotting the end-effector height trajectories (hands and feet) of the local motion output of our method since the algorithm we use to estimate 3D pose assumes centered human input. Below we plot end-effector trajectories of our retargetted poses and the originally estimated poses (input to our method) for selected examples (None: Please check our project website for better appreciation of the denoising happening goo.gl/mDTvem).

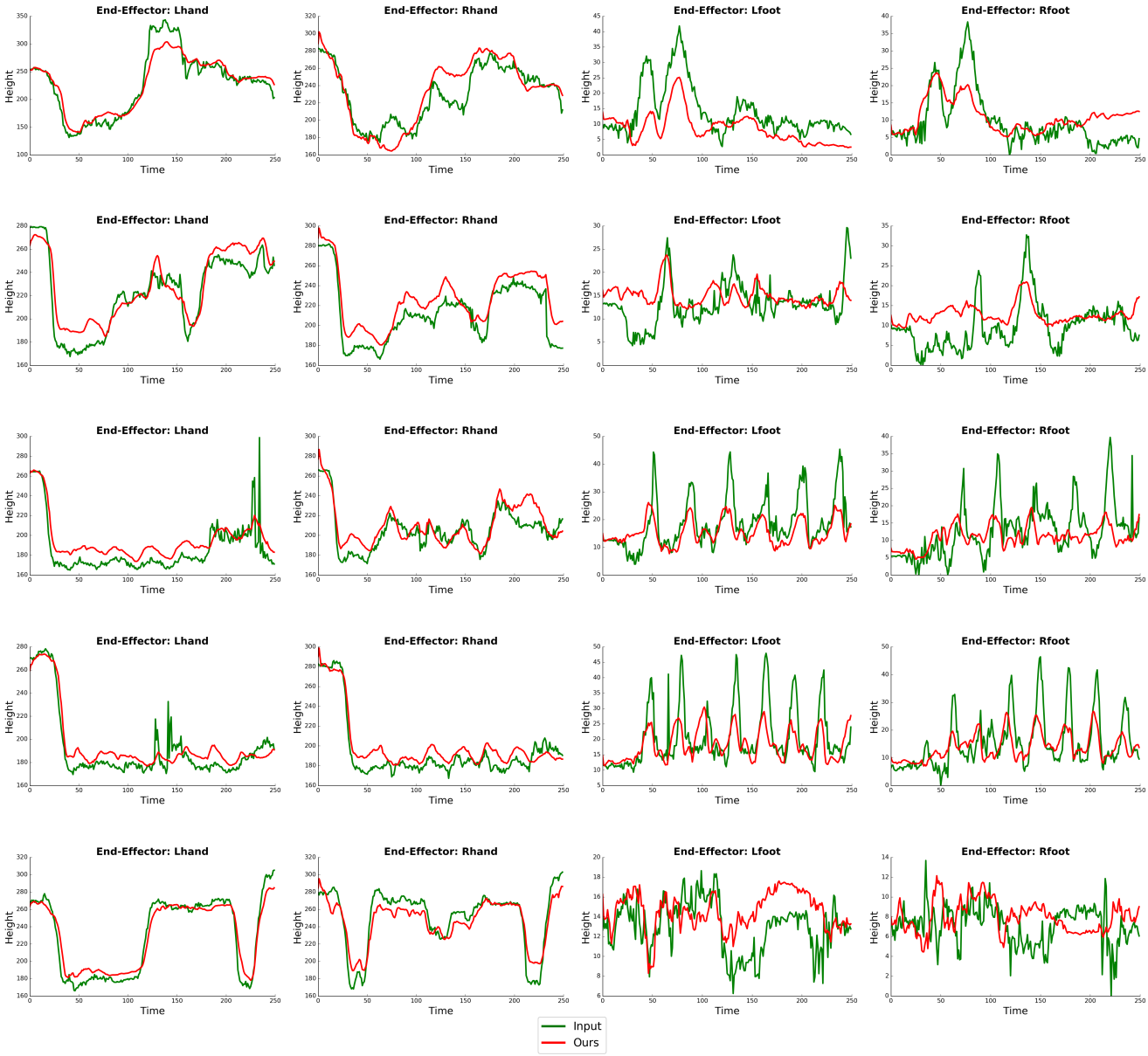

Figure 8: 3D pose estimation denoising. We present end-effector trajectoriess for 5 examples. Each row belongs a single example in the Human3.6M test set used in [16]. Please refer to our website for visual illustrations of the denoising results. goo.gl/mDTvem.

In Figure 8, we can see that our method denoises the hand end-effectors well, even without having trained with such data before. The feet end-effector denoising is good as well, however in some cases it misses the overall feet height the original estimation had. However, if we take a look at the provided videos, we can clearly see that our method's understanding of the input motion allows it to fix a lot of the shaking seen in the initially estimated 3D pose after motion retargetting.

C. Demo Video and Qualitative Motion Retargetting Evaluation

For the results demo video, please refer to the youtube video link https://youtu.be/v13eGcl08AY (Note: The demo video contains audio. Please wear headphones if you believe you may disturb people around you). For more videos, please go to goo.gl/mDTvem.

D. Data collection process

In this section, we describe the exact steps for collecting the training and testing data from the Mixamo website [1]. As training data, we collected 1656 unique motion sequences distributed over 7 different characters. As testing data, we use 68 unique sequences of at least 4 seconds each (74 total) from which we extract 173 unique non-overlapping 4-second clips (185 total). Please note that the last clip in each sequence which may overlap if there are less that 4 seconds left over after all non-overlapping clips have been extracted. The Mixamo website contains motion separated by pages, the specific pages we downloaded for each character are specified in Table 4 below:

Training data	
Character	Motion sequence page
Malcolm	[1-5]
Warrok W Kurniawan	[6-10]
Goblin D Shareyko	[11-15]
Kaya	[16-20]
Peasant Man	[21-25]
Big Vegas	[26-30]
AJ	[31-35]

Test data	
Input -> Target	Motion sequence page
Malcolm	28, 51
Warrok W Kurniawan	18, 52
Liam	23, 45
Mutant	33, 45, 52
Claire	52
Sporty Granny	51

Table 4: Data collection for each character and animation page in the Mixamo website.

At test time, we perform motion retargetting for each testing scenario as shown in Tables 5 and 6 below:

Known Motion / Known Character	
Input → Target	Motion sequence page
Kaya → Warrok W Kurniawan	18
Big Vegas → Malcolm	28

Known Motion / New Character	
Input → Target	Motion sequence page
Peasant Man → Liam	23
AJ → Mutant	33

Table 5: Quantitative evaluation of online motion retargetting using mean square error (MSE). Case study: Known motion / known character (left), and known motion / new character (right).

New Motion / Known Character	
Input → Target	Motion sequence page
Sporty Granny → Malcolm	51
Claire → Warrok W Kurniawan	52

New Motion / New Character	
Input → Target	Motion sequence page
Mutant → Liam	45
Claire → Mutant	52

Table 6: Quantitative evaluation of online motion retargetting using mean square error (MSE). Case study: New motion / known character (left), and new motion / new character (right).

E. Architecture and training details

In this section, we provide the network architectures detailes used throughout this paper.The RNN architectures are implemented by a 2-layer Gated Recurrent Unit (GRU) with 512-dimensional hidden state. As the discriminator network, we use a 5 layer 1D fully-convolutional neural network with size 4 kernel, and convolutions across. Layers 1-4 have leakyReLU activations with leak of 0.2, dropout of 0.7 keep probability, "same" convolution output with stride 2. Layers 2-4 each have a instance normalization layer with default parameters in the tensorflow implementation. The last layer implements a "valid" convolution with linear activation. For training the networks, we use the Adam optimizer with a learning rate of 1e-4 for both the retargetting RNN and Discriminator, and clip the RNN gradients by global norm of 25. We also implemented a balancing technique between the retargetting network and the discriminator, where the discriminator is not updated if the probability of the generator output being a real falls below 0.3.