CAS 737 - Final Project Report
Motion Transfer and Retargeting

Amir Eskandari

April 2023

1 Introduction

Character motion plays a crucial role in many industries such as including but not limited to Game Development, Cinematic Film Production, Augmented and Virtual Reality (AR/VR), Robotics, and Computer Vision. The study of character motion, therefore, assumes critical importance owing to its far-reaching implications and impact on these sectors. There are different ways for generating motion in 3D space from keyframe animation generation by artists using various software to using Motion Capture (MoCap). Motion Capture (MoCap) is widely used in the gaming and AR/VR industries to capture human motion. The MoCap technique involves recording the motion data (e.g., position, orientation, etc.) of a specific actor through the use of sensors. This data is collected during various actions, such as walking, jogging, and jumping, and is subsequently used to animate a 3D character in different software applications. The popularity of MoCap has increased significantly in recent years, largely due to its speed and efficiency when compared to keyframe animation, which is created manually by artists. Despite its advantages, MoCap remains a challenging and expensive task, as there are endless possibilities for character motion that cannot be captured feasibly. Thus, the importance of reusing available motion data becomes critical.
There exist multiple techniques for reusing motion data in the generation and editing of new motion. Previously, animation researchers utilized hand-engineering methods to attain their desired motions. However, in recent years, data-driven approaches have gained momentum with the considerable increase in the quantity of available motion data. As a result, the focus has shifted toward the utilization of data-driven techniques in the field of animation.
The rapid evolution of these data-driven methods that utilize machine learning made it possible to automatically generate motions or edit the available character animation. Furthermore Deep learning-based approaches help generating complex human motion more cost effectively. This made them an important subject of research. In the following report we will review some Retargeting methods based on the aforementioned methods and try to compare them for our better understanding of these trends.

2 Methods

Retargeting [1] is the process of taking a skeletal animation sequence that was created for one character and transferring it to another character with different body proportions. The goal of retargeting is to replicate the movements of the original sequence while adapting them to the new

character's body structure. This can be a complex task since the new character may have different bone lengths and a different skeletal topology. Despite the lack of a "precise definition" for the task as pointed out in [2], retargeting aims to abstract the dynamics of the original sequence and apply them to the new character's skeleton, while ensuring that the resulting motion looks natural and visually believable. Essentially, retargeting involves creating a new animation sequence for the target character that mimics the original source sequence while taking into account the unique features of the target character's body.
This section provides an overview of previous research on motion transfer and retargeting. Early approaches for motion retargeting utilized skeleton parametrizations, which describe human pose using a sparse set of joint locations. Gleicher [1] pioneered this area by presenting a technique for adapting the motion of one character to another character with different body proportions but the same skeletal structure. This method identifies features of the source motions as kinematic constraints and formulates retargeting as a constrained optimization problem with space-time constraints. Many optimization-based motion retargeting methods followed, introducing specific constraints such as joint angle constraints [3], trajectory constraints [4], or dynamics constraints [5]. Furthermore, the authors in [6] proposed a hierarchical framework that adaptively adds motion details to satisfy certain constraints by adopting a multilevel B-spline representation scheme, which provides a significant speedup by avoiding the time-consuming space-time optimization process. However, these methods require a long iterative optimization process with hand-designed kinematic constraints for particular motions, making them computationally expensive. Some of them also incorporate inverse kinematics (IK) in the algorithm, which is an arithmetically inefficient process for characters with many constraints and complex joint structure, despite many numerical IK solvers based on Jacobian matrices being proposed to find optimal joint angle displacements for characters to move the end-effectors slightly toward the goal in an iterative manner.
In recent times, the availability of large amounts of captured motion data and the success of deep learning techniques in various fields have led to the development of new data-driven approaches for retargeting. These approaches utilize deep feedforward networks for motion retargeting, which has become an attractive option due to their ability to produce outstanding results without the need for the aforementioned hand-engineered processes. Compared to traditional optimization-based methods, data-driven approaches using deep feedforward networks offer several advantages. These methods offer faster computation times and can handle more complex motion data, making them an attractive option for many applications. In addition, these methods require less manual intervention, which makes them easier to use and more efficient.
Even though these methods require less manual intervention, some of them utilize paired training data, whose design needs human effort. Paired training data involves having both a source and a target sequence with exactly the same motion, which can be challenging to capture in practice. As a result, researchers have also explored unsupervised retargeting strategies that rely on unpaired training data without motion correspondences across characters.

One of the most influential works in unsupervised models is introduced by Villegas et al. in [7]. Their proposed model's design is based on two RNNs and a Forward Kinematic Layer. One of the RNNs is used to encode the source's motion context to features/hidden state, which is then fed to the other decoder RNN for it to decode the joint rotations of the target skeleton from it. Moreover, the forward kinematic layer utilizes a reference T-pose of the target skeleton and the joint rotations to render the output motion, making it a skeleton-conditioned network.
To learn the model in an unsupervised manner, the authors have used an adversarial training objective, based on the cycle consistency principle [8], alongside regularization loss. Specifically, the cycle

consistency loss ensures that when a motion that is retargetted to a target skeleton is retargetted back the source one, it stays as close as possible to the original motion of the source. The adversarial loss also attempts to minimize differences between the generated motion with other known motions of the target, making the results as natural as possible. A simplified overview of the this model can be seen in Figure. 1.

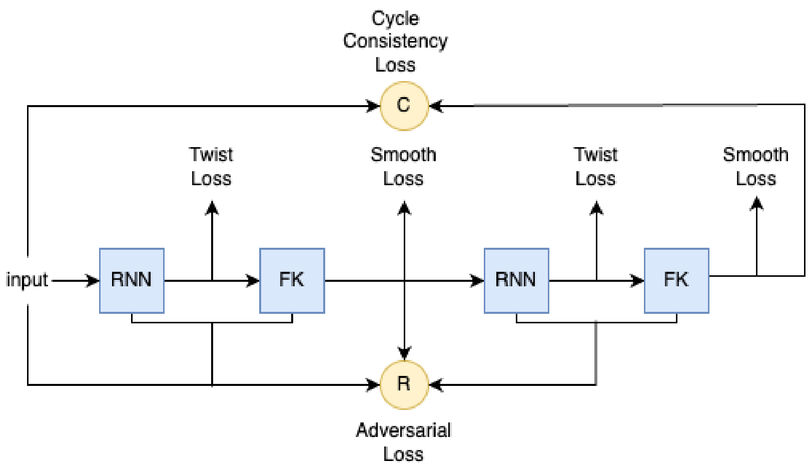
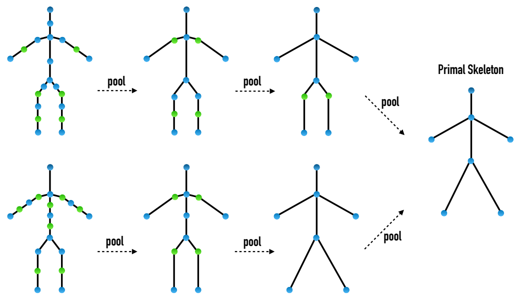
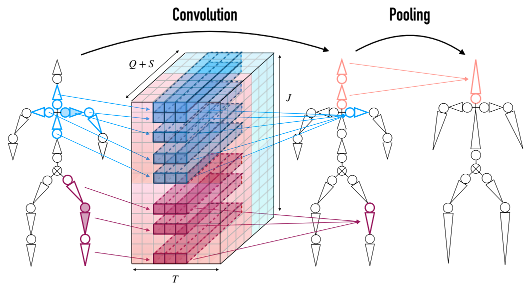


Figure 1: [7] Simplified Overview. Twist loss prevents excessive bone twisting. Smoothing loss is used to prevent extreme changes in global motion in consecutive timesteps, making the result more realistic and smooth.

Nevertheless, the said method has its limitations, like performing retargeting on only a fixed number of joints, or requiring 3D coordinates of the joints, making it impractical for video data. [9] mentions another considerable limitation of this method, which is ignoring any physical constraints, like gravity, in animating the target character, resulting in “bodies floating above the ground.” To address these issues, [9] proposes a method called PMnet. PMnet, contrary to previous models that learned the motion in a recurrent fashion, disentangles the character motion and learns frame-by-frame poses and overall movement separately to obtain specialized and complementary representations from them. In particular, the frame-by-frame pose and the overall movement are referred to as the relative coordinates from the root joint position, and the the velocity of the root joint respectively. To achieve this, two parts are involved in the model: a) a pose encoder with two mapping networks, b) a movement regressor network. The pose encoder is focused on ensuring that the target character has a pose that is similar to the input character. At each frame, the pose encoder encodes the pose representation that is invariant to the skeleton, and the mapping networks convert the pose representation into unit quaternions that can adapt to the target character’s different kinematic configuration. The next part, is for generating the overall movement of the target character, making the generated motion appear realistic. This is achieved through a movement regressor network and a normalization process that makes the movement independent of the character’s size. Furthermore, a novel training loss is used to train PMnet, where the cycle consistency loss of [7] is replaced by a reconstruction loss associated to self-retargeting to the same character. Other parts of the training loss are associated with perceptual pose, motion discrimination, and rotation constraints.
Current approaches cannot automatically perform retargeting between skeletons that differ in their



(a) Primal Skeleton Obtained With Pooling



(b) Skeletal Convolution and Pooling

Figure 2: Figures From [2]

structure or the number of joints. However, this is not always the case, and it is substantial to be able to retarget motion between skeletons that may have different structure. An approach in this situation, is to manually specify the correspondences between the different skeletons, which often cause unavoidable retargeting errors. Dealing with this matter, [2] extended the scope of retargeting to skeletons with different topologies, subject to the condition that all considered topologies are homeomorphic. Their key idea is that these homeomorphic skeleton graphs can all be reduced into a common minimal graph by eliminating nodes of degree 2 along linear branches, resulting in a *primal skeleton* (Figure. 2(a)). This way, the unchanging (static) bone offset vectors can be separated from the changing (dynamic) bone rotations for each joint, resulting in the model having two parallel branches: static, and dynamic. Moreover, several *skeleton-aware* differentiable operators (Figure. 2(b)) are introduced in this model, meaning that they explicitly consider the skeleton structure. The retargeting system consists of three kinds of modules: space-time graph-convolutional operators that operate on joint neighborhoods in space and time, along with graph pooling and unpooling operators. Graph pooling combines two neighboring edges (bone segments) into one feature, while graph unpooling separates one edge into two adjacent edges, copying the original features. An autoencoder is created for each skeletal structure in the training dataset, consisting of an encoder and a decoder. The encoder takes motion data for character A and generates embeddings for the static bone offset and dynamic joint rotation components expressed in a common skeleton topology. The decoder for character B uses these embeddings to produce the retargeted motion for this character. Retargeting is then accomplished by combining the encoder for the source character, with the decoder for the target character.

3 Comparison & Supplementary Material

In this section, we will elaborate on the supplementary material attached with this review paper, which are the results of trying various retargeting methods. For performing a fair comparison, we have chosen the characters ”Granny” and ”Aj” from the Mixamo [10] dataset, and motion of ”Open Door Outwards”.

Firstly, we have tried retargetting using the industry-standard software **Houdini** and the free and open source **Blender**. In Houdini we used the node-base network to do the retargeting. Firstly, we imported the characters ”AJ” and ”Granny” by FBX character import node. Then we imported

Method	**URL**
Deep-Motion-Editing [2, 11]	`https://github.com/DeepMotionEditing/deep-motion-editing`
Neural Kinematic Networks for Unsupervised Motion Retargetting [7]	`https://github.com/rubenvillegas/cvpr2018nkn`
TransMoMo: Invariance-Driven Unsupervised Video Motion Retargeting [12]	`https://github.com/yzhq97/transmomo.pytorch`
MoCaNet: Motion Retargeting in-the-wild via Canonicalization Networks [13]	`https://github.com/Walter0807/mocanet.pytorch`
Learning Character-Agnostic Motion for Motion Retargeting in 2D [14]	`https://github.com/ChrisWu1997/2D-Motion-Retargeting`
Rokoko Live Studio Retargeting Algorithm	`https://github.com/Rokoko/retargeting.py`

Table 1: A set of publicly available source codes for retargeting methods

the animation from Mixamo "Open Door Outwards". In next step we used "Rig Match Pose" to scale the source to target skeleton and changed the initial orientation and position of source skeleton to match the target skeleton. Lastly the "Full Body IK" will solve the IK for the source skeleton and the retargeting will be done. you can see the process in the `Granny_houdini.mp4` and `AJ_houdini.mp4` .
As Blender is an open-source software, many plug-ins are available for it for different tasks. Retargeting in Blender is done with a plug-in called Rokoko, which is used for working with motion capture data. The process of using this software to retarget the motion of opening the door outwards from the character Aj, to the character Sporty Granny can be viewed in the file `blender.mov`.
Furthermore, we explored the publicly available source codes of the recently proposed retargetting methods, which can be seen in the table 1.

As much as we wanted to be able to run these models and test them, we couldn't do it due to lack of computational resources and pre-trained models. However, we were able to test the code available for [2], which is part of a bigger library built by the authors, called "deep motion editing." The result and the process of running this model on the aforementioned characters can be seen in file `skeleton.mov`. The code is also attached alongside this paper, and can be tested with any character and motion in its dataset, using the script `demo.py`. Also, the subset of Mixamo used in the method can be accessed using this link.

Finally, in order to comprehend these methods better, we decided to implement one of the deep-learning based retargeting models from scratch. For this matter, we chose to implement [2]. Most of the skeleton-aware operators introduced in the paper is implemented, alongside the encoder and the decoder part of the model. Even though they are bug-free, they do not have visually understandable results yet. Therefore, just the source code is attached with this paper (in the folder `my-skeleton-network`) for interested readers.

References

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

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

[3] Kwang-Jin Choi and Hyeong-Seok Ko. Online motion retargetting. *The Journal of Visualization and Computer Animation*, 11(5):223–235, 2000.

[4] Andrew Feng, Yazhou Huang, Yuyu Xu, and Ari Shapiro. Automating the transfer of a generic set of behaviors onto a virtual character. In *Motion in Games: 5th International Conference, MIG 2012, Rennes, France, November 15-17, 2012. Proceedings 5*, pages 134–145. Springer, 2012.

[5] Seyoon Tak and Hyeong-Seok Ko. A physically-based motion retargeting filter. *ACM Transactions on Graphics (TOG)*, 24(1):98–117, 2005.

[6] Jehee Lee and Sung Yong 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, 1999.

[7] Ruben Villegas, Jimei Yang, Duygu Ceylan, and Honglak Lee. Neural kinematic networks for unsupervised motion retargetting. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 8639–8648, 2018.

[8] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *Proceedings of the IEEE international conference on computer vision*, pages 2223–2232, 2017.

[9] Jongin Lim, Hyung Jin Chang, and Jin Young Choi. Pmnet: Learning of disentangled pose and movement for unsupervised motion retargeting. In *BMVC*, volume 2, page 7, 2019.

[10] Adobe. Mixamo.

[11] Kfir Aberman, Yijia Weng, Dani Lischinski, Daniel Cohen-Or, and Baoquan Chen. Unpaired motion style transfer from video to animation. *ACM Transactions on Graphics (TOG)*, 39(4):64, 2020.

[12] Zhuoqian Yang, Wentao Zhu, Wayne Wu, Chen Qian, Qiang Zhou, Bolei Zhou, and Chen Change Loy. Transmomo: Invariance-driven unsupervised video motion retargeting. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[13] Wentao Zhu, Zhuoqian Yang, Ziang Di, Wayne Wu, Yizhou Wang, and Chen Change Loy. Mocanet: Motion retargeting in-the-wild via canonicalization networks. In *AAAI*, 2022.

[14] Kfir Aberman, Rundi Wu, Dani Lischinski, Baoquan Chen, and Daniel Cohen-Or. Learning character-agnostic motion for motion retargeting in 2d. *ACM Transactions on Graphics (TOG)*, 38(4):75, 2019.