PMnet

Lim, Jongin; Chang, Hyung Jin; Choi, Jin Young

PMnet: Learning of Disentangled Pose and Movement for Unsupervised Motion Retargeting

Jongin Lim[1]
ljin0429@snu.ac.kr
Hyung Jin Chang[2]
h.j.chang@bham.ac.uk
Jin Young Choi[1]
jychoi@snu.ac.kr

[1] Department of ECE, ASRI, Seoul National University, Seoul, Korea
[2] School of Computer Science, University of Birmingham, Birmingham, UK

Abstract

In this paper, we propose a deep learning framework for unsupervised motion retargeting. In contrast to the existing method, we decouple the motion retargeting process into two parts that explicitly learn poses and movements of a character. Here, the first part retargets the pose of the character at each frame, while the second part retargets the character's overall movement. To realize these two processes, we develop a novel architecture referred to as the pose-movement network (PMnet), which separately learns frame-by-frame poses and overall movement. At each frame, to follow the pose of the input character, PMnet learns how to make the input pose first and then adjusts it to fit the target character's kinematic configuration. To handle the overall movement, a normalizing process is introduced to make the overall movement invariant to the size of the character. Along with the normalizing process, PMnet regresses the overall movement to fit the target character. We then introduce a novel loss function that allows PMnet to properly retarget the poses and overall movement. The proposed method is verified via several self-comparisons and outperforms the state-of-the-art (sota) method by reducing the motion retargeting error (average joint position error) from 7.68 (sota) to 1.95 (ours).

1 Introduction

Motion retargeting is the process of applying a source motion to a new target character with different kinematic configurations. More formally, given a motion sequence of character A and kinematic information of another character, B, the goal is to make B imitate the motion of A so that another motion sequence of B can be generated. For this paper, we assumed that the skeletons have the same topology but different bone lengths and proportions. Due to its reusability for motion data, motion retargeting has been studied extensively in computer graphics [9, 11, 23, 34] and robotics [4, 6, 29, 33]. Most existing methods have formulated motion retargeting as a constrained optimization problem. However, since they usually rely on motion-specific constraints or manually designed kinematic constraints, these optimization-based methods cannot generalize to a wide range of motions and characters.

Recent deep learning based methods [7, 10, 15, 16, 18, 25, 26, 31] have succeeded in modeling human motion based on training using large-scale motion capture datasets [1, 17]. However, applying the deep learning framework to motion retargeting is challenging because there is a lack of paired motion data from different characters. Thus, a supervised learning scheme with the ground truth for the retargeted motion is impractical. Recently, Villegas *et al.* [36] proposed a recurrent neural architecture for motion retargeting. By exploiting cycle consistency [39], the possibility of unsupervised motion retargeting was presented. However, the resulting motions showed a large number of errors in preserving the pose of the input and also showed unrealistic motions, such as bodies floating above the ground.

In our study, to address the aforementioned limitations, we propose a novel architecture referred to as the pose-movement network (PMnet) for unsupervised motion retargeting, which learns frame-by-frame poses and overall movement separately. Here, the frame-by-frame pose is referred to as the relative coordinates from the root joint position, while the overall movement is referred to as the velocity of the root joint [15, 16]. In contrast to previous work [36], which learns the character motion in a recurrent architecture, we disentangle the character motion into frame-by-frame poses and overall movement to learn specialized and complementary representations from them.

The proposed method consists of two parts. The first part learns to ensure that the target character has a pose similar to the input character. To this end, we propose a novel architecture consisting of a pose encoder and two mapping networks. At each frame, the pose encoder encodes the skeleton invariant pose representation, and two mapping networks map the pose representation to unit quaternions for the target character to adapt to a different kinematic configuration than the input character. The second part learns to generate the overall movement of the target character, which makes the resulting motion seem realistic. To accomplish this, we propose a movement regressor network and a normalizing process that make the movement invariant to the size of the character. Then, to ensure that the aforementioned parts work properly, we design a novel training loss. Rather than using cycle consistency, our training loss consists of the following four loss terms: *i) reconstruction*, *ii) perceptual pose*, *iii) motion discrimination*, and *iv) rotation constraint*. By means of the new loss terms, PMnet implicitly learns to retarget motion while preserving the detailed pose of the input and making realistic movements. We validate the proposed method via several self-comparisons and show that it significantly outperforms the previous work, reducing the motion retargeting error from 7.68 to 1.95 in the same experimental setup.

2 Related Works

2.1 Motion Retargeting

To adapt the motion from source to target, Gleicher [11] formulated a spacetime constraint problem, then solved it numerically. Lee *et al.* [23] decoupled the motion retargeting problem, which solved the inverse kinematics problem for each frame and then adjusted multilevel B-spline curves for smooth results. Choi *et al.* [9] presented online motion retargeting technique by solving inverse kinematics problem under constraints and computing the changes in joint angles at each frame. Tak *et al.* [34] proposed per-frame motion retargeting framework which incorporates both kinematic constraints and dynamic constraints into Kalman filter formulation. Motion retargeting techniques can be applied to adapting motions from humans to humanoid robots [4, 6, 8, 28, 29, 30, 32]. While retargeting human motion onto humanoids additionally requires considerations, they are basically based on optimizations to meet specified constraints [4]. The aforementioned methods have limitations in applying to large scale motion data since they relied on optimizations with motion-specific constraints or manually designed kinematic constraints. Shon *et al.* [33] proposed Gaussian Process model to learn common latent structure shared between sets of motion capture data and corresponding poses from a humanoid robot, presenting robotic imitation of human poses. But, this method requires a set of paired training data from two domains.

Recently, instead of numerical approaches, data-driven approaches based on deep learning framework have been proposed. Jang *et al.* [19] proposed a deep learning framework that can produce 27 variations of the source motion with a set of different levels of arms, legs, and torso lengths. However, this method requires fully-supervised training from the corresponding data pairs based on [22]. Furthermore, it requires post-optimization process for the latent variable to meet the valid kinematic consistency. Villegas *et al.* [36] proposed *Neural Kinematic Network* with forward kinematics layer and cycle consistency [39] based objectives, which suggested the possibility of unsupervised motion retargeting. However, the quality of resulting motion has still room for improvement.

2.2 Human Motion Modeling

Modeling human motion is a long standing problem in computer vision and machine learning. Early works used Restricted Boltzmann Machines [35], Markov Models [24] and Gaussian Process [13, 38], but limited to small scale of motions. The most common strategy in recent years is a data-driven approach where postures are reconstructed based on large-scale motion capture datasets [1, 17]. Specifically, deep learning based methods have succeeded in synthesizing or predicting plausible human motions [7, 15, 16, 25]. However, these methods can not be applied to motion retargeting because they require re-projection onto kinematic constraint to avoid invalid bone length configuration as they regress on joint positions. Several works have used human poses described through 3D joint rotations [10, 18, 26], which used the angle loss so that they were vulnerable to small angle errors at the root joint. Pavllo *et al.* [31] represented rotations with unit quaternions and presented loss function which performs forward kinematics on a skeleton to penalize position errors instead of angle errors. However, these methods are not applicable when joint angles are not given and only joint positions are observed.

3 Methodology

Let $x^A_{1:T}$ be a source motion performed by the character A, given as T frames of 3D joint positions. Motion retargeting is the process of generating the motion $x^B_{1:T}$ which is the same action sequence performed by another character B. Assuming that the kinematic configuration of the target character is given as a T-pose skeleton, namely, ref_B, Villegas *et al.* [36] formulated motion retargeting problem as outputting 3D rotations for each joint parameterized by unit quaternions and applying them to ref_B. To this end, forward kinematics layer was presented to compute the resulting joint positions from the joint rotations and ref_B. Resulting *Neural Kinematic Network* (NKN) with forward kinematics layer was trained based on cycle consistency objectives and suggested the possibility of unsupervised motion retargeting. However, NKN shows unsatisfactory results in preserving the details of input motions, and it even shows unrealistic motions such as body floating or sinking when input motions have

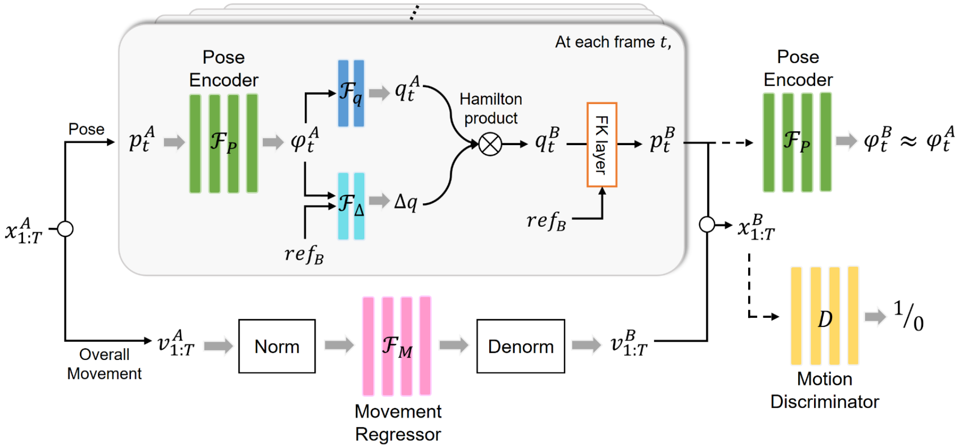


Figure 1: Overview of the proposed method.

dynamic movements. This is because cycle consistency is insufficient to capture the proper pose representation. Also, there is a lack of consideration for the overall movements of the character. Our research focuses on solving the aforementioned problems.

The motion capture data is often decomposed into two parts, the relative coordinates from the root joint position and the the root joint's velocities and direction (rotation with respect to the axis perpendicular to the ground) [15, 16]. In this paper, we viewed the former as frame-by-frame poses and the latter as a overall movement.

As shown in Figure 1, the input motion data $x_{1:T}^A$ is decomposed into frame-by-frame poses $p_{1:T}^A$ and the overall movement $v_{1:T}^A$ (see [16] for more details). Then, we decouple the process to handle the pose and the movement separately for the purpose of extracting different and complementary meanings. At each frame t, the proposed method outputs the unit quaternions $q_t^B \in \mathbb{R}^{4N}$, which represent the 3D rotations for each joint of the character B, from the input pose $p_t^A \in \mathbb{R}^{3N}$ of A and the target T-pose skeleton of B, $ref_B \in \mathbb{R}^{3N}$, where N is the number of joints. To make the target character B follow the pose of the input character A in good shape, we present a novel architecture that successfully encodes the skeleton invariant pose representation from the pose (see Section 3.1). The retargeted pose of the character B, p_t^B, is then computed by applying q_t^B to ref_B using forward kinematics (FK) layer. When processing the overall movement, on the other hand, the temporal context must be grasped. Therefore, $v_{1:T}^B$, the movement of the target character, is regressed from the entire input movement $v_{1:T}^A$ to capture the character's overall movement. (see Section 3.2). Then, the retargeted motion $x_{1:T}^B$ can be computed by combining the obtained frame-by-frame pose and the overall movement of the character B. Note that learning proceeds in an unsupervised manner, *i.e.*, there is no ground truth for the retargeted motion $x_{1:T}^B$. To this end, we further introduce a novel unsupervised learning scheme which allows the model to retarget motions while maintaining good pose and overall movement (see Section 3.3).

3.1 Architecture for Pose Retargeting

In this section, we present an architecture that produces the unit quaternions q_t^B to make the target character B follow the pose of the input p_t^A at each frame. This is challenging because there is no ground truth for p_t^B, so we can not give a guide to what pose B should take. The key insight to tackle this problem is to learn how to reconstruct p_t^A first, then make minor modifications to fit with B.

Auto-encoders have shown reliable results for unsupervised feature learning [14, 37]. Similarly, we present the auto-encoding phase to reconstruct the input pose p_t^A. To encode the pose representation from p_t^A, we present a pose encoder $\mathcal{F}_P$. That is,

$$\varphi_t^A = \mathcal{F}_P(p_t^A; W_P), \tag{1}$$

where W_P denotes the connection weights of the pose encoder. The pose representation φ_t^A from the pose encoder conveys shape information about the pose at t frame. Then, a mapping function $\mathcal{F}_q$ is introduced to map the pose representation φ_t^A to the quaternion space. After the mapping, the output is normalized because it must be defined as unit quaternions to represent the 3D rotations as follows,

$$q_t^A = \frac{\mathcal{F}_q(\varphi_t^A; W_q)}{\left\|\mathcal{F}_q(\varphi_t^A; W_q)\right\|_2}. \tag{2}$$

Here, q_t^A represents the rotation to make the pose of A. Then, to make modifications to fit with B, another mapping function $\mathcal{F}_\Delta$ is presented. $\mathcal{F}_\Delta$ outputs the modifications on the quaternion space from the pose representation φ_t^A and the T-pose skeleton ref_B as follows,

$$\Delta q = \frac{\mathcal{F}_\Delta(\varphi_t^A, ref_B; W_\Delta)}{\left\|\mathcal{F}_\Delta(\varphi_t^A, ref_B; W_\Delta)\right\|_2}. \tag{3}$$

In the quaternion space, the product of two rotation quaternions q_1 and q_2, called the Hamilton product, represents the rotation equivalent to the rotation of q_1 followed by the rotation of q_2. That is, q_t^B, modified version of q_t^A by Δq, can be expressed as a hamilton product of two quaternions. Then, the retargeted pose p_t^B can be obtained as follows,

$$q_t^B = q_t^A \otimes \Delta q, \tag{4}$$

$$p_t^B = \mathrm{FK}(q_t^B, ref_B). \tag{5}$$

By learning to reconstruct p_t^A by applying q_t^A to ref_A using FK layer, what rotations q_t^A makes the desired pose can be learned. Thus, learning the subtle changes in q_t^A is much easier than learning from a zero base (see Section 3.3).

3.2 Architecture for Movement Retargeting

In this section, we present an architecture which handles the overall movement. Unlike the pose, the temporal context must be grasped. To this end, we present a movement regressor $\mathcal{F}_M$ which regresses the entire input movement $v_{1:T}^A \in \mathbb{R}^{4T}$. Recent studies suggest that convolutional models empirically outperform recurrent models [5, 20, 27]. In our work, $\mathcal{F}_M$ is realized by 1D convolutions on the time sequence.

Since the overall movement contains the x, y, z velocities of the root joint, it is affected by the size of the character, *e.g.*, for the same walking motion, the adult step is faster than the child step. To capture the scale invariant information about the overall movement, the overall movement is normalized with respect to the scale of the character before fed into the movement regressor $\mathcal{F}_M$. The scale factor S of the character is defined as follows,

$$S = \frac{1}{N_s} \sum_{i \in H} d(i, parents(i)), \tag{6}$$

where N_s be a scaling coefficient that adjusts S not to be too large, H be a set of joints whose elements are *Root, Spine 0,1,2, Neck, LeftUpLeg, LeftLeg and LeftFoot*, and $d(i,j)$ denotes the $L2$ distance between i and j. Then, normalizing is done by dividing the x,y,z velocities (except the rotation) by the corresponding scale factor. As shown in Figure 1, $v^A_{1:T}$ is normalized to $nv^A_{1:T}$ with respect to its scale factor S_A then the movement regressor $\mathcal{F}_M$ regresses the scale invariant overall movement $nv^B_{1:T}$ from $nv^A_{1:T}$. That is,

$$nv^B_{1:T} = \mathcal{F}_M(nv^A_{1:T}; W_M). \tag{7}$$

Similarly, $nv^B_{1:T}$ is then denormalized to $v^B_{1:T}$ by multiplying the scale factor S_B to the x, y, z velocities in $nv^B_{1:T}$.

3.3 Unsupervised Training for Motion Retargeting

In this section, we present the training loss which allows the aforementioned parts to work properly. Our loss consists of four parts:

i) Reconstruction Loss. First, the model learns to reconstruct p^A_t by applying q^A_t to ref_B using FK layer. By minimizing the reconstruction error, $\mathcal{F}_P$ learns to encode the meaningful pose representation φ^A_t while $\mathcal{F}_q$ learns what rotations q^A_t makes the desired pose given φ^A_t. The reconstruction loss for the entire T frames is defined as follows,

$$\mathcal{L}_{recon} = \sum_t \left\| p^A_t - \mathrm{FK}(q^A_t, ref_A) \right\|^2_2. \tag{8}$$

Additionally, meeting the end-effector positions may be critical to make the pose similar. In our work, joints that affect the end-effector positions have double the weight to the loss.

ii) Perceptual Pose Loss. By Eq. (8), the rotation q^A_t makes the desired pose of A. B should be the same pose of A but minor modifications are needed because of the different kinematic configuration. To this end, we present the perceptual pose loss exploiting the pose encoder $\mathcal{F}_P$. We feed the retargeted pose p^B_t into $\mathcal{F}_P$ then penalize the pose representation φ^B_t to be close to the pose representation of A, φ^B_t. As learning proceeds to minimize the difference between the pose representations, the pose encoder $\mathcal{F}_P$ learns to encode the skeleton invariant pose representation. Then, the modifications coming from the different kinematic configuration can be encoded in Δq. In our experiments, we minimize the L_2 distance between the pose representations and use $\lambda_p = 5$. That is,

$$\mathcal{L}_{pose} = \lambda_p \left\| \varphi^A_{1:T} - \varphi^B_{1:T} \right\|^2_2. \tag{9}$$

iii) Motion Discrimination Loss. Now, we present the loss term that penalize to make the retargeted motion $x^B_{1:T}$ look realistic. We define the mean point trajectory $\bar{m}_{1:T}$ where $\bar{m}_t = mean(x_t) \in \mathbb{R}^3$, *i.e.*, the mean position of all joints. Then, $\bar{m}^A_{1:T}$ and $\bar{m}^B_{1:T}$ are normalized with respect to its corresponding scale factors computed by Eq. (6). To avoid the unrealistic motion such as body floating or sinking, we make these normalized mean point trajectories be similar to each other. Exploiting the adversarial training [12], we present the motion discriminator $D(.;W_D)$ which discriminates the normalized mean point trajectories. Then, PMnet is trained to fool the discriminator as follows,

$$\mathcal{L}_{motion} = \lambda_m \mathbb{E}_{x \sim p_B(x)} \left[\log \left(1 - D \left(\frac{\bar{m}^B_{1:T}}{S_B} \right) \right) \right], \tag{10}$$

where $x \sim p_B(x)$ means the sampling from the distribution of retargeted motion generated by the proposed method and $\lambda_m = 2$ is used as a balancing parameter.

iv) Rotation Constraint Loss. The last term in our loss aims to limit the rotation angles to be within a certain range, which avoids the excessive bone twisting.

$$\mathcal{L}_{rot} = \lambda_r \left(\left\| \max(0, |euler_y(q^A_{1:T})| - \alpha) \right\|_2^2 + \left\| \max(0, |euler_y(q^B_{1:T})| - \alpha) \right\|_2^2 \right). \tag{11}$$

As in [36], $euler_y(.)$ denotes the rotation angle around the plane parallel to the bone (*i.e.*, y-axis). In our experiments, $\lambda_r = 10$ and $\alpha = 100°$ are used.

Finally, the total loss is defined by the summation of all four loss terms as follows,

$$\mathcal{L}_{total} = \mathcal{L}_{recon} + \mathcal{L}_{pose} + \mathcal{L}_{motion} + \mathcal{L}_{rot}. \tag{12}$$

The training proceeds end-to-end, which updates the whole network parameters W_P, W_q, W_Δ, and W_M and the discriminator parameter W_D iteratively. As in [36], when constructing a mini batch, we chose the same character to the inputs with probability of $p = 0.5$ for training stability. We used the Adam optimizer [21] with a batch size of 16 and a learning rate of 0.0001 then trained the model for 15000 iteration.

3.4 Implementation

We used TensorFlow [1] for implementation. The pose encoder $\mathcal{F}_P$ and two mapping networks $\mathcal{F}_q$ and $\mathcal{F}_\Delta$ are realized by fully-connected (fc) layers while the movement regressor $\mathcal{F}_M$ and the discriminator D are realized by 1D convolution (1DConv) layers. The detailed architectures are reported in Table 1. The source code is available at `https://github.com/ljin0429/PMnet`.

4 Experiments

We evaluated the motion retargeting capability of the proposed PMnet on the Adobe Mixamo dataset [2] and compared the results with the state-of-the-art method (NKN [36]). The NKN results were obtained using the code released by the authors. We also compared the results with the most trivial approach for motion retargeting which directly copies the input quaternions and velocities (Copy). For fair comparison, we followed the same experimental setup as described in NKN. The training set consisted of 1650 non-overlapping motion sequences for 7 characters. For training, we used randomly sampled 2-second clips (60 frames). For testing, we evaluated the methods on 185 scenarios. Each scenario involved retargeting the 4-second clip (120 frames) of the motion sequence. The ground truths of the testing sequences were also collected for quantitative evaluation. For quantitative evaluation, we evaluated a target character-normalized mean square error (MSE) on the joint positions throughout the entire sequences, which is the same metric presented in NKN. For qualitative evaluation, we visualized the results by rendering the animated 3D characters. Further, we validated the proposed method through several self-comparisons.

Table 1: Detailed architecture for $\mathcal{F}_P$, $\mathcal{F}_q$, $\mathcal{F}_\Delta$, $\mathcal{F}_M$ and D. We used the leak rate of 0.2 for leaky ReLU and applied dropout for the whole network with the keep probability of 0.8. Please visit `https://github.com/ljin0429/PMnet` for more details.

Name	Layer	Number of Neurons	Activation
$\mathcal{F}_P$	input	66	-
	fc	512	ReLU
	fc	512	ReLU
	fc	512	ReLU
	fc	512	ReLU
$\mathcal{F}_q$	input	512	-
	fc	88	-
$\mathcal{F}_\Delta$	input	512	-
	fc	88	-

Name	Layer	Kernel	Stride	Channel Depth	Activation	Length
$\mathcal{F}_M$	input	-	-	4	-	60
	1DConv	3	1	128	leaky ReLU	60
	1DConv	3	1	4	-	60
D	Input	-	-	3	-	60
	1DConv	4	2	16	leaky ReLU	30
	1DConv	4	2	32	leaky ReLU	15
	1DConv	4	2	64	leaky ReLU	8
	1DConv	4	2	128	leaky ReLU	4
	1DConv	4	2	1	-	1
	output	-	-	1	sigmoid	1

4.1 Self-comparisons

In order to validate the effectiveness of the proposed method, we compared the PMnet with the following four variants: *wPP*-PMnet, *wM*-PMnet, *wR*-PMnet, and *eqR*-PMnet. In *wPP*-PMnet, $\mathcal{L}_{pose}$ was omitted while, in *wM*-PMnet, $\mathcal{L}_{motion}$ was omitted. Then, we investigated the effectiveness of reconstruction. We omitted $\mathcal{L}_{recon}$ in *wR*-PMnet and used the equal penalty for all joints when computing $\mathcal{L}_{recon}$ in *eqR*-PMnet.

As shown in Table 2, the four variants of PMnet also achieved higher performances than NKN and the Copy baseline. This demonstrates that the proposed PMnet architecture is effective for learning about the pose and movement for motion retargeting. The results of *wPP*-PMnet showed increased error compared with PMnet, reflecting that the perceptual pose loss $\mathcal{L}_{pose}$ contributes to making the proper modifications to fit with the target character. Without $\mathcal{L}_{motion}$ in *wM*-PMnet, there is no guide for retargeting the overall movement properly, which results in increased motion retargeting error. Without $\mathcal{L}_{recon}$, as expected, *wR*-PMnet shows significant performance degradation. This means that the proposed scheme, which learns to reconstruct A first and then modify it rather than directly learning B from a zero-base, is effective. In addition, we confirmed that more weights to the joints that affect the end-effectors when computing $\mathcal{L}_{recon}$ results in a slight improvement to the performance as compared with *eqR*-PMnet.

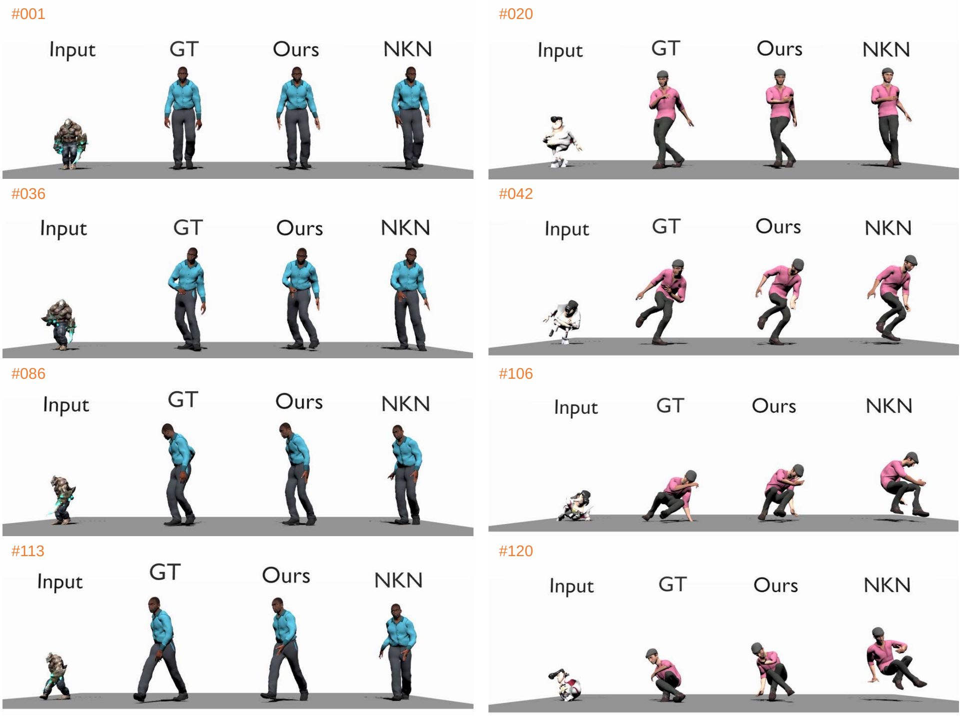


Figure 2: Qualitative comparisons. We present two motion retargeting results, from Mutant to Liam (left) and from Big Vegas to Malcolm (right). The number in orange on the upper left indicates the frame number.

4.2 Qualitative and Quantitative Comparisons

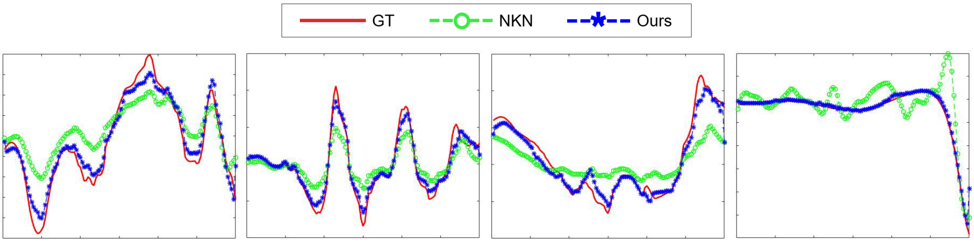
We confirmed that the proposed PMnet is capable of retargeting motion and significantly outperforms NKN, as shown in Table 2. For the NKN results, we listed both the performance we achieved using the provided code and the performance reported in their paper. PMnet showed superior performance (1.95) to all the others, reducing the error by 74.6% compared with NKN (7.68) and 78.3% compared wiht the Copy baseline (9.00).

Figure 2 shows the qualitative results of PMnet and NKN. We present the results from two testing sequences. The number in orange on the upper left indicates the frame number. As shown in the left sequence, unlike NKN, PMnet preserved the details of the input motion well, including the motion of the legs and tilt of the body. This demonstrates that our model is capable of encoding good pose representation and making proper modifications to adapt it to different kinematic configurations. Further, NKN showed unrealistic movements (floating body) as can be seen in the right sequence. Even in this case, PMnet also showed reliable results, which demonstrates the effectiveness of the proposed movement regressor with normalizing and denormalizing processes.

Figure 3 shows the overall movement from PMnet, NKN, and the ground truth where (a)-(c) depict the x, y, z velocities of the root joint and (d) depicts the rotation of the root joint. The overall movement of PMnet is much more similar to the ground truth, suggesting that the resulting motion from PMnet looks more realistic, as shown in Figure 2.

Table 2: Quantitative results on Mixamo dataset [2]. For the NKN results, the numbers on the left are the results of our experiments using the code provided from the authors, and the numbers in brackets indicate the performance reported in their paper.

Method	MSE	Notes
NKN (CVPR'18) [36]	10.50 (10.25)	Auto-Encoder
	7.68 (8.51)	Cycle loss
	8.87 (**7.10**)	Cycle + Adv loss
Copy	9.00	Copy input quaternions and velocities
wPP-PMnet	2.54	without $\mathcal{L}_{pose}$
wM-PMnet	2.79	without $\mathcal{L}_{motion}$
wR-PMnet	2.99	without $\mathcal{L}_{recon}$
eqR-PMnet	2.15	equal weights for $\mathcal{L}_{recon}$
PMnet	**1.95**	Ours



(a) x-velocity (b) y-velocity (c) z-velocity (d) rotation

Figure 3: Plots for the overall movement where the horizontal axis represents the frame sequence and the vertical axis represents the corresponding velocity and rotation values. We present the larger plots at `https://github.com/ljin0429/PMnet`.

5 Conclusion

In this paper, we proposed a new unsupervised motion retargeting framework based on learning disentangled meanings of pose and movement. Our contributions can be summarized as follows: 1) We decoupled motion retargeting process into two parts, where the first part generates the pose of the target character in each frame and the second part regresses the overall movement. 2) To generate the target pose, a novel architecture for obtaining quaternions of the input and modifying it using the Hamilton product was proposed. 3) The movement regressor along with normalizing and denormalizing processes was proposed to generate realistic movements. 4) A new training loss was designed to afford our deep network the capability to generate a target motion that mimics the input pose well and shows realistic movements. The experiments show compelling results, illustrating that the proposed method significantly outperforms the state-of-the-art method qualitatively and quantitatively.

Acknowledgement

This work was supported by Next-Generation ICD program through NRF funded by Ministry of S&ICT [2017M3C4A7077582], ICT R&D program of MSIP/IITP [No.B0101-15-0552, Predictive Visual Intelligence Technology] and Brain Korea 21 Plus Project.

References

[1] Cmu graphics lab motion capture database. URL http://mocap.cs.cmu.edu/.

[2] Adobe's mixamo. URL https://www.mixamo.com/.

[3] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S Corrado, Andy Davis, Jeffrey Dean, et al. Tensorflow: Large-scale machine learning on heterogeneous distributed systems. *arXiv preprint arXiv:1603.04467*, 2016.

[4] Ko Ayusawa and Eiichi Yoshida. Motion retargeting for humanoid robots based on simultaneous morphing parameter identification and motion optimization. *IEEE Transactions on Robotics*, 33(6):1343–1357, 2017.

[5] Shaojie Bai, J Zico Kolter, and Vladlen Koltun. An empirical evaluation of generic convolutional and recurrent networks for sequence modeling. *arXiv preprint arXiv:1803.01271*, 2018.

[6] Ghassan Bin Hammam, Patrick M Wensing, Behzad Dariush, and David E Orin. Kinodynamically consistent motion retargeting for humanoids. *International Journal of Humanoid Robotics*, 12(04):1550017, 2015.

[7] Judith Butepage, Michael J Black, Danica Kragic, and Hedvig Kjellstrom. Deep representation learning for human motion prediction and classification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6158–6166, 2017.

[8] Hyung Jin Chang, Tobias Fischer, Maxime Petit, Martina Zambelli, and Yiannis Demiris. Kinematic Structure Correspondences via Hypergraph Matching. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4216–4225, June 2016. doi: 10.1109/CVPR.2016.457.

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

[10] Katerina Fragkiadaki, Sergey Levine, Panna Felsen, and Jitendra Malik. Recurrent network models for human dynamics. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 4346–4354, 2015.

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

[12] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014.

[13] Keith Grochow, Steven L Martin, Aaron Hertzmann, and Zoran Popović. Style-based inverse kinematics. In *ACM transactions on graphics (TOG)*, volume 23, pages 522–531. ACM, 2004.

[14] Geoffrey E Hinton and Ruslan R Salakhutdinov. Reducing the dimensionality of data with neural networks. *science*, 313(5786):504–507, 2006.

[15] Daniel Holden, Jun Saito, Taku Komura, and Thomas Joyce. Learning motion manifolds with convolutional autoencoders. In *SIGGRAPH Asia 2015 Technical Briefs*, page 18. ACM, 2015.

[16] Daniel Holden, Jun Saito, and Taku Komura. A deep learning framework for character motion synthesis and editing. *ACM Trans. Graph.*, 35(4):138:1–138:11, July 2016. ISSN 0730-0301. doi: 10.1145/2897824.2925975. URL http://doi.acm.org/10.1145/2897824.2925975.

[17] Catalin Ionescu, Dragos Papava, Vlad Olaru, and Cristian Sminchisescu. Human3.6m: Large scale datasets and predictive methods for 3d human sensing in natural environments. *IEEE transactions on pattern analysis and machine intelligence*, 36(7): 1325–1339, 2014.

[18] Ashesh Jain, Amir R Zamir, Silvio Savarese, and Ashutosh 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.

[19] Hanyoung Jang, Byungjun Kwon, Moonwon Yu, Seong Uk Kim, and Jongmin Kim. A variational u-net for motion retargeting. In *SIGGRAPH Asia 2018 Posters*, page 1. ACM, 2018.

[20] Angjoo Kanazawa, Jason Zhang, Panna Felsen, and Jitendra Malik. Learning 3d human dynamics from video. *arXiv preprint arXiv:1812.01601*, 2018.

[21] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[22] Tejas D Kulkarni, William F Whitney, Pushmeet Kohli, and Josh Tenenbaum. Deep convolutional inverse graphics network. In *Advances in Neural Information Processing Systems*, pages 2539–2547, 2015.

[23] Jehee Lee and Sung Yong Shin. A hierarchical approach to interactive motion editing for human-like figures. In *Siggraph*, volume 99, pages 39–48, 1999.

[24] Andreas M Lehrmann, Peter V Gehler, and Sebastian Nowozin. Efficient nonlinear markov models for human motion. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1314–1321, 2014.

[25] Zimo Li, Yi Zhou, Shuangjiu Xiao, Chong He, Zeng Huang, and Hao Li. Auto-conditioned recurrent networks for extended complex human motion synthesis. *arXiv preprint arXiv:1707.05363*, 2017.

[26] Julieta Martinez, Michael J Black, and Javier Romero. On human motion prediction using recurrent neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2891–2900, 2017.

[27] John Miller and Moritz Hardt. When recurrent models don't need to be recurrent. *arXiv preprint arXiv:1805.10369*, 2018.

[28] Kanako Miura, Mitsuharu Morisawa, Shin'ichiro Nakaoka, Fumio Kanehiro, Kensuke Harada, Kenji Kaneko, and Shuuji Kajita. Robot motion remix based on motion capture data towards human-like locomotion of humanoid robots. In *Humanoid Robots, 2009. Humanoids 2009. 9th IEEE-RAS International Conference on*, pages 596–603. IEEE, 2009.

[29] Thomas Moulard, Eiichi Yoshida, and Shin'ichiro Nakaoka. Optimization-based motion retargeting integrating spatial and dynamic constraints for humanoid. In *IEEE ISR 2013*, pages 1–6. IEEE, 2013.

[30] Christian Ott, Dongheui Lee, and Yoshihiko Nakamura. Motion capture based human motion recognition and imitation by direct marker control. In *Humanoids 2008-8th IEEE-RAS International Conference on Humanoid Robots*, pages 399–405. IEEE, 2008.

[31] Dario Pavllo, David Grangier, and Michael Auli. Quaternet: A quaternion-based recurrent model for human motion. *arXiv preprint arXiv:1805.06485*, 2018.

[32] Oscar E Ramos, Layale Saab, Sovannara Hak, and Nicolas Mansard. Dynamic motion capture and edition using a stack of tasks. In *2011 11th IEEE-RAS International Conference on Humanoid Robots*, pages 224–230. IEEE, 2011.

[33] Aaron Shon, Keith Grochow, Aaron Hertzmann, and Rajesh P Rao. Learning shared latent structure for image synthesis and robotic imitation. In *Advances in neural information processing systems*, pages 1233–1240, 2006.

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

[35] Graham W Taylor, Geoffrey E Hinton, and Sam T Roweis. Modeling human motion using binary latent variables. In *Advances in neural information processing systems*, pages 1345–1352, 2007.

[36] 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.

[37] Pascal Vincent, Hugo Larochelle, Yoshua Bengio, and Pierre-Antoine Manzagol. Extracting and composing robust features with denoising autoencoders. In *Proceedings of the 25th international conference on Machine learning*, pages 1096–1103. ACM, 2008.

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

[39] 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.