MoCaNet: Motion Retargeting in-the-wild via Canonicalization Networks

Wentao Zhu,†1,2 Zhuoqian Yang,*3 Ziang Di,4 Wayne Wu,2,3† Yizhou Wang,1 Chen Change Loy5

1 School of Computer Science, Peking University 2 Shanghai AI Laboratory 3 SenseTime Research 4 Southeast University 5 S-Lab, Nanyang Technological University

wtzhu@pku.edu.cn, yangzhuoqian@sensetime.com, 213180794@seu.edu.cn, wuwenyan0503@gmail.com, yizhou.wang@pku.edu.cn, ccloy@ntu.edu.sg

Abstract

We present a novel framework that brings the 3D motion retargeting task from controlled environments to in-the-wild scenarios. In particular, our method is capable of retargeting body motion from a character in a 2D monocular video to a 3D character without using any motion capture system or 3D reconstruction procedure. It is designed to leverage massive online videos for unsupervised training, requiring neither 3D annotations nor motion-body pairing information. The proposed method is built upon two novel canonicalization operations, structure canonicalization and view canonicalization. Trained with the canonicalization operations and the derived regularizations, our method learns to factorize a skeleton sequence into three independent semantic subspaces, i.e., motion, structure, and view angle. The disentangled representation enables motion retargeting from 2D to 3D with high precision. Our method achieves superior performance on motion transfer benchmarks with large body variations and challenging actions. Notably, the canonicalized skeleton sequence could serve as a disentangled and interpretable representation of human motion that benefits action analysis and motion retrieval.

Introduction

3D motion retargeting aims at transferring one character’s motion to a virtual 3D avatar. It is an important and challenging task in computer vision and computer graphics with a wide spectrum of applications in human-computer interaction (Hoshyari et al. 2019; Kim and Lee 2020) and augmented reality (Kang et al. 2019; Kim et al. 2016). Traditional approaches rely on motion capture (MoCap) systems (Ott, Lee, and Nakamura 2008; Koenemann, Burget, and Bennewitz 2014), which are capable of capturing precise 3D motion but are limited to sophisticated environments only available to film and game studios. This kind of setup is inaccessible for everyday users with in-the-wild usage scenarios, e.g., on-device augmented reality.

The challenge of in-the-wild motion retargeting lies in that videos recorded on mobile devices or downloaded from the Internet only provide 2D visual information (usually noisy), from which 3D motion needs to be inferred. The common practice is to estimate 3D pose first, then perform 3D-to-3D retargeting through Inverse Kinematics (IK). Nevertheless, the results of current 3D pose estimation (Ci et al. 2019; Pavllo et al. 2019; Moon, Chang, and Lee 2019; Yang et al. 2018) and model-based mesh reconstruction (Liu et al. 2019; Kocabas, Athanasiou, and Black 2020; Choi, Moon, and Lee 2020) methods deteriorate severely under occlusions, large body variations and uncommon actions from unconstrained real-world videos, as discussed in (Dong et al. 2020; Chen et al. 2019b; Duan et al. 2021). The errors originated from 3D pose estimation can easily propagate to the IK stage and degrade the quality of retargeted motion.

In this work, we argue that the conventional way of performing IK after 3D estimation is not sufficiently robust and reliable for in-the-wild motion retargeting. To this end, we propose an end-to-end learnable model, Canonicalization Networks, which allows us to bypass the error-prone direct 3D pose estimation and use an off-the-shelf 2D pose estimator to extract geometrical information that is more robust and reliable. Our method then takes the 2D skeleton sequences and generates 3D retargeted skeleton sequences, as shown in Fig. 1. However, retargeting motion from 2D skeleton sequences is not a trivial task. Arbitrarily captured real-world videos contain large structural variations and diverse camera views. The same action, e.g., standing up, may appear dramatically different in 2D under different views given different body structures. It is therefore meaningful and challenging to extract the unmixed motion in a high-level semantic modality.

We tackle the challenge through the novel notion of canonicalization and retargeting. Canonicalization, by def-
in our approach enables applying regularization on 3D poses assuming static view angle. Meanwhile, our canonicalization-based approach to retarget motion in 2D. It requires the exact same structures. Aberman et al. proposed a supervised learning method to handle different body confidence map as 2D pose representations. Chan et al. designed a global pose normalization to handle different body kinematic constraints. In recent years, deep-learning-based approaches show promising results on obtaining human pose and action, which not only increases the availability of motion data, but also inspires motion retargeting research with deep neural networks. Villegas et al. proposed to capture high-level motion with Forward Kinematics (FK) layers in recurrent neural networks. Lim, Chang, and Choi further optimized the retargeting precision by disentangling pose and movement. Skeleton-Aware Networks (Aberman et al. 2020) could automatically adapt to different skeleton topologies. Nevertheless, the above methods all require high-precision movement. Skeleton-Aware Networks (Aberman et al. 2020) could automatically adapt to different skeleton topologies. Nevertheless, the above methods all require high-precision motion (e.g., quaternions), which takes expensive motion capture systems or complex optimization in practice. Otherwise, estimating 3D motion from monocular RGB video could be error-prone (Dong et al. 2020; Chen et al. 2019b; Duan et al. 2021).

Several works have also explored to retarget motion from 2D inputs that are more accessible to everyday use cases. Aberman et al. proposed a two-branch framework with part confidence map as 2D pose representations. Chan et al. designed a global pose normalization to handle different body structures. Aberman et al. proposed a supervised learning approach to retarget motion in 2D. It requires the exact same motion performed by different characters at different view angles as training data. TransMoMo (Yang et al. 2020) explored unsupervised motion retargeting via invariance properties. These methods all generate 2D label maps for further rendering via image-to-image translation methods. Therefore, they can not be readily applied to driving 3D characters. They only consider accuracy for 2D joint positions and assume static view angle. Meanwhile, our canonicalization-based approach enables applying regularization on 3D poses directly and training with time-varying explicit 3D view.

In our framework, 2D poses are used as input and our
method produces 3D skeletons as output. Therefore, our approach can exploit user-friendly monocular RGB videos with the help of robust 2D pose estimation algorithms (Cao et al. 2018; Bulat et al. 2020; Artacho and Savakos 2020). In addition, the 3D skeleton output allows combination with state-of-the-art neural rendering (Wang et al. 2018, 2019) and avatar creation (Saito et al. 2020; Deng et al. 2020; Alldieck et al. 2018; Weng, Curless, and Kemelmacher-Shlizerman 2019) techniques. Our flexible approach thus provides a viable alternative to the traditional 3D reconstruction+IK pipeline.

**Representation Disentanglement.** Disentangling latent variables from observations has been a fundamental topic in machine learning. Previous approaches (Kingma et al. 2014; Mathieu et al. 2016) use labeled data to separate class-dependent and class-independent features. A vast literature focuses on unsupervised learning of disentangled representations with generative models (Chen et al. 2014; Denton and Brodak 2017; Villegas et al. 2017; Tulyakov et al. 2018; Higgins et al. 2017; Lee et al. 2018; Huang et al. 2018). A recent line of work explored view-angle-agnostic motion representations aimed at improving motion recognition performance (Liu et al. 2021; Nie, Liu, and Liu 2020; Zhao et al. 2021). In this work, we achieve disentanglement of motion, body-structure and view-angle via unsupervised canonicalization.

**3D Canonicalization.** Several different methods have been proposed for automatically computing the canonical form from a 3D pose or mesh. (Pickup et al. 2015b,a, 2018, 2016) propose to learn pose-neutral shape of any non-rigid meshes for 3D shape retrieval. (Shamai, Zibulevsky, and Kimmel 2015) further improves the method with multidimensional scaling. Lian, Godil, and Xiao present a feature-preserved canonical form that directly deforms the 3D models. Body reconstruction methods (Saito et al. 2020; Wang, Geiger, and Tang 2021) sometimes involve the canonical pose space for handling pose-dependent deformations. In those papers, canonicalization refers to pose canonicalization, that is, “transforming to a standard pose”. In this work, we explore the canonicalization of different factors, namely structure and view angle, while keeping the motion unchanged. C3DPO (Novotny et al. 2019) designed a rotation-invariant canonical shape for structure reconstruction. CanonPose (Wandt et al. 2021) learns a canonical camera view and respective camera rotations, but it relies on multiple synchronized cameras for training. We design canonicalization operations for both structure and view angle, which are applied to the entire action sequence rather than a single pose. Based on the designed operations, our model can be trained without data pairing.

**Methodology**

**Overview**

Our solution to in-the-wild 3D motion retargeting uses 2D skeleton sequences for training and inference. Specifically, we extract pose sequences from unannotated web videos with an off-the-shelf human pose estimator (Alp Güler, Neverova, and Kokkinos 2018; Girshick et al. 2018). We devise a 2D-to-3D autoencoder that factorizes the input 2D human pose sequence into three independent spaces, namely character-agnostic 3D motion, body structure of the character and view angle. Notably, we formulate the view angle to be explicit and dynamic, which allows for direct manipulation. 3D motion retargeting is then achieved through decoding a recombination of motion and structure encoded from different sources.

We design a training scheme that simultaneously achieves (i) 3D geometry inference and (ii) disentangled motion representation learning. The two goals are tightly coupled and mutually beneficial. The first goal is addressed via a 2D-to-3D AutoEncoder trained with adversarial learning (Chen et al. 2019a). The second challenge is addressed via the proposed self-supervised canonicalization.

**2D-to-3D AutoEncoder**

As shown in Figure 4, the proposed 2D-to-3D AutoEncoder is a modular encoder-decoder network, where the encoding part decomposes an input 2D skeleton sequence $x \in \mathbb{R}^{2N \times T}$ into three spaces: motion, structure and view angle, while the decoding part takes any combination of motion and structure and decodes a 3D skeleton sequence. Here, $N$ denotes the number of body joints and $T$ denotes the sequence length. The decoded 3D skeleton sequence should reproduce the input when projected back to 2D while remain plausible when it is viewed at different angles. This serves as the foundation of the representation learning framework and allows us to manipulate the data for canonicalization training.

**Encoders.** The encoded representation of the three spaces are defined as follows: Motion is represented as a temporal sequence of latent vectors $E_m(x) = m \in \mathbb{R}^{M \times C_m}$, where $M$ is the encoded length and $C_m$ is the number of latent channels. The motion encoder uses several 1-D temporal convolutional layers to extract this information, yielding the encoded length $M = T/8$. The structure encoder has a similar network structure $E_s(x) = s \in \mathbb{R}^{M \times C_s} = [s^{(1)}, s^{(2)}, \ldots, s^{(M)}]$. This can be interpreted as performing multiple structure estimations in sliding windows. Since the structure does not change with time, we use a temporal max pooling to aggregate information from these estimations to
get the final structure: \( E_v(x) = \tilde{s} = \text{maxpool}(s), \tilde{s} \in \mathbb{R}^C \).

We assume view angle to be dynamic, therefore it is encoded as a temporal sequence of explicit 3D rotations. Instead of using the commonly used Euler angles or quaternions, we choose to use a continuous 6D representation, which is better suited for deep learning regression (Zhou \textit{et al.} 2019). Specifically, \( E_v(x) = \nu \in \mathbb{R}^{6 \times N}, \) i.e., \( \nu \) is a sequence of 6D rotation vectors, each rotation vector \( \nu_i \) is responsible for rotating one frame. \( \nu \) can be uniquely translated to a sequence of rotation matrices \( \Omega = [\omega(1), \omega(2), \ldots, \omega(T)], \omega(i) \in SO3 \) using a function: \( g(\nu) = \Omega \).

Decoder. As shown in Fig. 4, the decoder takes the concatenation of the motion and structure as input to decode a 3D skeleton sequence \( G(m, s) = X_{vc} \in \mathbb{R}^{3N \times T} \). The decoded sequence is assumed to be in the canonical view angle. To reproduce the input 2D sequence, the output 3D skeleton sequence \( X_{vc} \) is rotated using the encoded view angles \( \nu \) and then projected to 2D, i.e., \( X_{rec} = r(X_{vc}, g(\nu)) \). The function \( r(X, \Omega) \) is one which rotates a 3D skeleton sequence \( X \) with a sequence of SO3 rotation matrices \( \Omega \). \( o(X) \) stands for orthographic projection.

**3D Self-Supervision.** Since 3D ground-truth is not available, the ill-posed 2D-to-3D lifting is achieved via supervision in 2D space. Specifically, we use (i) an L1 reconstruction loss to ensure that the reprojected 2D skeleton sequence reproduces the input and (ii) an adversarial loss to ensure the reconstructed 3D skeleton, when viewed at random angles, still lies within the distribution of real 2D skeleton sequences. Specifically, the L1 reconstruction loss is defined as

\[
\mathcal{L}_{\text{rec}} = |x - x_{rec}|,
\]

where \( x \) is the input 2D sequence and \( x_{rec} \) is the sequence reconstructed by the autoencoder. To calculate the adversarial loss, we rotate the lifted 3D sequence with some random view angles. We generate \( K \) versions of rotated sequences \( X_{rec}^{(k)} = r(X_{rec}, \Omega_{k}), X_{vc}^{(k)} = o(X_{rec}^{(k)}) \). Note that the rotation matrices are time-varying, and in practice we keep them temporally smooth by interpolation. A discriminator \( D \) is used for the adversarial training. The discriminator tries to distinguish between real and reprojected 2D sequences while the AutoEncoder tries to make its 3D output indistinguishable regardless of viewing angle. This process eliminates the ambiguity in the 2D-to-3D mapping by penalizing unusual structures in the reprojected views (Chen \textit{et al.} 2019a). The adversarial loss is defined as

\[
\mathcal{L}_{\text{adv}} = \sum_{k=1}^{K} \mathbb{E}_{x \sim p_d} \left[ \log D(x) + \log(1 - D(x_{rec}^{(k)})) \right]
\]

**Canonicalization**

Canonicalization aims at restoring canonicality in one of the spaces while leaving the other two unchanged, a process that enforces the independence of one space against the other two. We propose view canonicalization (restore canonical view for any 3D skeleton sequence) and structure canonicalization (restore canonical body structure for any 3D skeleton sequence). We randomly manipulate the 3D sequence in only one of the three spaces and apply canonicalization in that space for both the original sequence and the manipulated sequence. This process gives us several variants that can be used to derive the supervision signals.

**View Canonicalization.** View canonicalization \( \phi_{vc}(x) \) restores canonical view for any skeleton sequence, i.e., performing view canonicalization on figures with arbitrary orientation should yield figures facing the same direction, while leaving the motion and body structure unchanged. This is achieved by reconstructing the 3D sequence with the decoder and then rotating it to canonical view, i.e., \( \phi_{vc}(x) = r(G(E_m(x), E_s(x)), g(v_{cano})) \). The canonical view \( v_{cano} \) is defined as corresponding to the identity matrix, i.e., no rotation. Thus, the output of the decoder is assumed to be in a canonical view, see Fig. 5 (a) for an example.

Recall in the previous section that we generated 2D reprojections \( x_{rec}^{(k)} \) of the input sequence \( x \). For training, we per-
form view canonicalization on both the input and the re-projected versions. Since \( \mathbf{x} \) and \( \mathbf{x}_r^{(k)} \) contain the same motion and structure and the only variable is the view angle, the results of view canonicalization should be identical. From this property we derive the view canonicalization loss in skeleton space:

\[
    L_{sc}^X = \sum_{k=1}^{K} |X_{sc} - \hat{X}_{sc}^{(k)}|,
\]

(3)

where \( X_{sc} = \phi_{sc}(\mathbf{x}) \) and \( \hat{X}_{sc}^{(k)} = \phi_{sc}(\mathbf{x}_r^{(k)}) \). Also, since rotation and view canonicalization does not modify information in the motion and structure space, the re-encoded motion and structure should stay the same. These properties give us the view canonicalization loss in feature space.

\[
    L_{sc}^m = |\mathbf{m} - m_{sc}| + \sum_{k=1}^{K} |\mathbf{m} - \mathbf{m}_r^{(k)}|,
\]

(4)

\[
    L_{sc}^s = |\mathbf{s} - s_{sc}| + \sum_{k=1}^{K} |\mathbf{s} - \mathbf{s}_r^{(k)}|,
\]

(5)

where \( \mathbf{m} \) and \( \mathbf{s} \) are the motion and structure encoded from the input sequence \( \mathbf{x} \). Together, the view canonicalization loss is defined as

\[
    L_{vc} = \lambda_{vc}^X L_{sc}^X + \lambda_{m}^m L_{sc}^m + \lambda_{s}^s L_{sc}^s.
\]

(6)

**Structure Canonicalization.** Structure canonicalization restores a standard body structure for any skeleton sequence while leaving the motion and view angle unchanged. During training, we estimate a centroid of the structure space and define it as the canonical structure \( s_{cano} \). Structure canonicalization is defined as \( \phi_{sc} = r(G(E_m(\mathbf{x}), s_{cano}), E_v(\mathbf{x})) \). The pipeline for structure canonicalization training is shown in Fig. 5 (b).

To achieve self-supervision, a technique called limb-scaling (Yang et al. 2020), i.e., randomly shortening or extending the length of the limbs in the input 2D sequence \( \mathbf{x} \), is used to obtain a sequence \( \mathbf{x}_s \), which has the same motion and view angle but a different body structure. We then perform structure canonicalization on both sequences, i.e., \( \mathbf{X}_{sc} = \phi_{sc}(\mathbf{x}) \), \( \hat{\mathbf{X}}_{sc} = \phi_{sc}(\mathbf{x}_s) \). By definition, this pair of canonicalization results should be the same, from which we derive the structure canonicalization loss in the skeleton space:

\[
    L_{sc}^X = |X_{sc} - \hat{X}_{sc}|.
\]

(7)

Since limb-scaling and structure canonicalization does not modify information in the other two spaces, the re-encoded motion and view angle should stay the same. These properties give us the structure canonicalization loss in feature space.

\[
    L_{sc}^m = |\mathbf{m} - m_{sc}| + |\mathbf{m} - \mathbf{m}_r|,
\]

(8)

\[
    L_{sc}^s = |\mathbf{s} - s_{sc}| + |\mathbf{s} - \mathbf{s}_r|.
\]

(9)

Together, the structure canonicalization loss is defined as

\[
    L_{sc} = \lambda_{sc}^X L_{sc}^X + \lambda_{m}^m L_{sc}^m + \lambda_{s}^s L_{sc}^s.
\]

(10)

**Total Loss**

The total training loss is given as

\[
    \mathcal{L} = \lambda_{rec} \mathcal{L}_{rec} + \lambda_{adv} \mathcal{L}_{adv} + \mathcal{L}_{vc} + \mathcal{L}_{sc}.
\]

We back-propagate \( \mathcal{L} \) and train the whole network in an end-to-end fashion.

**Experiments**

In this section, we first present the implementation details. We then evaluate and compare the motion retargeting performance on both in-the-wild and synthetic data. Next, we investigate the effects of canonicalization on the learned representations. We also conduct an ablation study to examine the effectiveness of each module. We include the details of the neural network and the metrics in the appendix.

**Implementation Details**

**Dataset.** For in-the-wild training, we use the SoloDancer(Yang et al. 2020) dataset which is a collection of YouTube videos featuring a single dancer in the scene. The dataset contains 8 categories of 337 videos. The total length of the videos add up to 11.5 hours. We then used an off-the-shelf 2D keypoints detector (Alp Güler, Neverova, and Kokkinos 2018) to extract keypoints frame-by-frame to be used as our training data. We also perform the proposed unsupervised training pipeline on the synthetic Mixamo dataset (Adobe 2021) in order to quantitatively measure the transfer results with ground truth and baseline methods. The training set comprises of 32 characters, each character has 800 sequences and a total of 1.2 hours for each character. During evaluation, we test the models on a held-out partition with 4 new characters and 64 new motions.

**Training.** Our canonicalization networks are trained for 200000 steps with batch size 64 and learning rate 0.0001 using an Adam (Kingma and Ba 2014) optimizer. The full model is end-to-end trained with all the proposed loss terms. The weights of the loss terms are given as follows: \( \lambda_{rec} = 15 \), \( \lambda_{adv} = 2 \), \( \lambda_{trip} = 10 \), \( \lambda_{vc}^X = \lambda_{vc}^m = 5 \), \( \lambda_{m}^m = \lambda_{sc}^m = \lambda_{sc}^s = 2 \). For view canonicalization, we generate \( K = 3 \) randomly rotated versions of the reconstructed 3D sequence. For limb-scaling in structure canonicalization, we use global and local scaling factors randomly sampled from a uniform distribution in the range [0.5, 2]. These parameters are empirically determined. All the experiments are implemented on a Linux Ubuntu machine with NVIDIA 1080Ti GPU using PyTorch 1.4 (Paszke et al. 2019). The model takes 3GB of GPU memory and 8 hours to train on a single GPU. We set all the random seeds to be 123.

**Motion Retargeting**

Figure 6 shows the motion retargeting results of our method. Each output skeleton is conditioned on the motion and view code from the source character and the structure code from the target character. The results demonstrate that our method could transfer the motion naturally and realistically to the target body. In addition, we compare our method with the state-of-the-art methods. To implement the conventional 3D+IK pipeline, we use several SOTA 3D pose estimation and mesh reconstruction methods as detailed below.
Table 1: 2D-to-3D (Synthetic) Motion Retargeting Comparison. We calculate the 3D Mean Square Error (MSE $\times 10^{-2}$) and 3D Mean Per Joint Position Error (MPJPE $\times 10^{-1}$). Both MSE and MPJPE are root-relative and normalized by the target character’s body height, following the practice of (Aberman et al. 2019b). The test characters are shown in Fig. 8.

| Methods          | Metrics           | 3DGT+IK MSE | 3DGT+IK MPJPE | Ours MSE | Ours MPJPE | Ours (wild) MSE | Ours (wild) MPJPE | TransMoMo MSE | TransMoMo MPJPE | LCN+IK MSE | LCN+IK MPJPE | Pose2Mesh+IK MSE | Pose2Mesh+IK MPJPE | VideoPose3D+IK MSE | VideoPose3D+IK MPJPE |
|------------------|-------------------|-------------|---------------|----------|------------|----------------|------------------|---------------|----------------|-------------|---------------|----------------|-------------------|-------------------|-------------------|
| Andro. $\leftrightarrow$ P. Hulk |                  | 0.049       | 0.325         | 0.799    | 1.221      | 0.792          | 1.215            | 1.246         | 1.563         | 1.986       | 1.925         | 2.822          | 2.361             | 3.198             | 2.469             |
| Andro. $\leftrightarrow$ S. Granny |               | 0.042       | 0.294         | 0.948    | 1.320      | 0.945          | 1.320            | 1.668         | 1.830         | 1.973       | 1.881         | 2.504          | 2.159             | 5.309             | 3.173             |
| Andro. $\leftrightarrow$ TY |               | 0.016       | 0.168         | 0.853    | 1.206      | 0.832          | 1.191            | 1.497         | 1.725         | 1.769       | 1.730         | 1.910          | 1.878             | 4.044             | 2.678             |
| P. Hulk $\leftrightarrow$ S. Granny |           | 0.131       | 0.507         | 0.955    | 1.334      | 0.951          | 1.333            | 2.057         | 2.072         | 2.235       | 2.013         | 3.182          | 2.491             | 6.390             | 3.760             |
| P. Hulk $\leftrightarrow$ TY |               | 0.037       | 0.371         | 0.874    | 1.230      | 0.853          | 1.220            | 1.837         | 1.965         | 2.135       | 1.942         | 2.544          | 2.235             | 5.471             | 3.429             |
| S. Granny $\leftrightarrow$ TY |              | 0.024       | 0.209         | 0.927    | 1.266      | 0.912          | 1.256            | 2.166         | 2.147         | 2.141       | 1.886         | 2.313          | 2.073             | 7.542             | 4.077             |
| OVERALL          |                  | 0.056       | 0.316         | 0.891    | 1.261      | 0.878          | 1.253            | 1.750         | 1.888         | 2.046       | 1.899         | 2.552          | 2.204             | 5.329             | 3.272             |

Figure 6: 3D Motion Retargeting in-the-Wild. The uppermost row gives the source videos, and the leftmost column gives the target videos. For each source-target combination, the network transfers the motion from source to target and produces 3D results.

**Video-to-3D (In-the-Wild) Comparison.** We first test under the in-the-wild setting where the source and target are given by dancing videos downloaded from the Internet, and the results are shown in Fig. 7. VideoPose3D (Pavllo et al. 2019) takes a 2D skeleton sequence and lifts it to 3D with temporal convolutions. 3DMPPE (Moon, Chang, and Lee 2019) regresses 3D pose from the 2D image directly. In the traditional two-stage pipeline, errors in the 3D pose estimation stage are accumulated and enlarged in the IK stage. Therefore, these methods are highly reliant on the quality of 3D pose estimation, which tends to fail due to occlusions or short 2D projections. TransMoMo (Yang et al. 2020) produces an intermediate 3D result before projection. However, it is designed to generate 2D label maps for further rendering, so they only consider 2D supervision with an oversimplified static view assumption. Although its 2D projection results are plausible, multiple artifacts can be found in 3D, e.g., wrong direction, temporally twisted moves.

**2D-to-3D (Synthetic) Comparison.** In order to quantitatively compare the motion retargeting performance with the previous methods, we compute the 2D-to-3D motion retargeting error on a synthetic animation dataset called Mixamo (Adobe 2021) because it has ground-truths available.

Figure 7: Video-to-3D (In-the-Wild) Motion Retargeting Comparison. The compared methods retarget the motion from source videos to target videos in the form of 3D skeleton sequences. Please zoom in for details, especially those circled with red that shows the erroneous results produced by existing methods.

Figure 8: Mixamo Testset Characters.

The training set comprises 32 characters, and each character has 500 actions. During evaluation, we test the models on a held-out partition with 4 new characters (Fig. 8) and 64 new actions. Since 3DMPPE (Moon, Chang, and Lee 2019) is not applicable to the 2D pose input, we further add two recent SOTA methods LCN (Ci et al. 2020) and Pose2Mesh (Choi, Moon, and Lee 2020) for the conventional pipeline. We further compare a strong benchmark where ground-truth 3D motion is retargeted with IK. It resembles the practice of using high-precision MoCap systems and serves as a lower bound of the error. The results are shown in Table 1. Our approach outperforms previous methods by a considerable margin on all the test character pairs. The performance of 3DGT+IK (the upper-bound baseline with 3D pose ground-truths) proves that the IK step itself is almost error-free. Hence, errors are mainly produced at the 3D stage and amplified in the IK stage as for the conventional two-stage pipeline. It is also notable that Solo-Dancer is only 30% of Mixamo training set in terms of video length. The model trained on Solo-Dancer performs better, which demonstrates the strength of training on in-the-wild data with higher mo-
Canonicalization and Disentanglement

By canonicalization training, the model learns to disentangle the three latent factors without external supervision. We first visualize the canonicalization results, which shows that the model learns to transform the skeleton sequences as intended. In addition, we show that the learned canonical form could serve as an effective representation of human motion.

Canonicalization Effects. As shown in Fig. 9, input skeleton sequences with different orientations are aligned to the same view angle after view canonicalization (Row 2), while the motion and structure information are not changed. The view canonicalization requires the network to infer the underlying 3D structure of the input motion sequence and automatically cast it to a specific view. Despite the large input body structure discrepancies, structure canonicalization (Row 3) generates a “standard character” performing the same motion at the exact view angle. The structure canonicalization requires the network to capture the average structure and learn a character-agnostic motion representation. As shown in the last row of Fig. 9, the 3D skeleton sequence after both structure and view canonicalization contains un-mixed motion information, independent of body structure and view angle. Therefore, it can be used as a direct, disentangled, and interpretable motion representation. We conduct further experiments to show the practicability of using the dual-canonicalized skeleton sequence as a distilled motion representation.

Motion Clustering. We apply a simple K-Means algorithm to cluster the Mixamo test set by motions. For K-Means, we set $K = 64$ and iterate 300 times. We visualize the motion clustering result in Fig. 10. The test skeleton sequences are canonicalized and mapped to 2D using t-SNE. The network learns to disentangle and separate pure motion through the proposed canonicalization operations despite the variance of character structure and camera view. The clustering results are further evaluated by ARI (Adjusted Rand index) (Hubert and Arabie 1985), AMI (Adjusted Mutual Information) (Gleicher 1998), Homogeneity, Completeness, and V-Measure (Rosenberg and Hirschberg 2007). We compare two motion representations: 1) Latent motion embedding learned by TransMoMo (Yang et al. 2020), and 2) Skeleton sequence after our canonicalization operations. Table 2 shows that our canonical motion representation consistently outperforms the previous work. It suggests that the proposed canonicalization operations help to learn better motion representation.

Motion Retrieval. The dual-canonicalized motion representation also enables the motion retrieval task, i.e., retrieve videos that exhibit similar motions with the query video. We first extract pose from the Solo-Dancer dataset, then slice the pose sequences into clips of 64 frames, producing a motion library of 33678 clips. Then, for a query motion, we retrieve its nearest neighbors in the dual-canonicalized motion representation space. The results in Fig. 11 show that our method is able to retrieve videos with similar motion semantics even if they appear drastically different due to body structure and view angle variations. It is because canonicalization operations make the proposed representation independent of both structure and view angle.

Consistency Evaluation. Ideally, the motion retargeting results should be robust to the variation of target character’s motion and view-angle. To examine our method’s capability of generating consistent motion-retargeted sequences, we test the same source sequence and target structure with varying target motion and view-angles, as shown in Fig. 12.
Figure 11: Motion Retrieval Results. The leftmost column shows a frame sampled from a query video depicting a short motion, and the three other columns show the top three retrieved videos.

Figure 12: Consistency Results. For each group (left, right), the target character is the same but with different actions and viewpoints. Given the same source video, the retargeting results are consistent.

Ablation Study

We first ablate several loss terms to show the importance of the designed modules. Then, we provide a stepwise comparison with TransMoMo (Yang et al. 2020). In Table 3, the removal of any part of our full model results in non-trivial performance degradation. More specifically, we observe that the network trained without adversarial loss tends to generate poses with a wrong depth, and the network trained without canonicalization loss fails to keep the desired motion. Although directly applying dynamic view and 3D loss terms to TransMoMo (Yang et al. 2020) slightly helps, the models still exhibit a considerable performance gap compared to the proposed canonicalization-based approach. The models also fail to learn time-varying 3D view angle as expected. We argue that the proposed canonicalization operations not only help to learn disentangled motion better, but also ease the learning of dynamic view representations, which are more realistic yet more complex.

Table 3: Ablation Study. w/o adv refers to the model with adversarial loss ablated, w/o VC refers to the model with view canonicalization loss ablated, w/o SC refers to the model with structure canonicalization loss ablated. T(3D) refers to TransMoMo re-implemented with 3D loss terms. T(3D+DV) refers to TransMoMo trained with 3D loss terms and dynamic view. The numbers are measured on the Mixamo dataset.

|          | w/o adv | w/o VC | w/o SC | T(3D) | T(3D+DV) | Ours (Full) |
|----------|---------|--------|--------|-------|----------|-------------|
| MSE × 10^{-2} | 1.579   | 1.240  | 1.599  | 1.652 | 1.688    | 0.891       |
| MPJPE × 10^{-1} | 1.785   | 1.602  | 1.693  | 1.776 | 1.743    | 1.261       |

Conclusion

In this work, we present Canonicalization Networks for tackling in-the-wild 2D-to-3D motion retargeting. The proposed method robustly infers and transfers motion from 2D monocular videos to drive 3D humanoids, enabling motion retargeting in everyday use cases. We derive self-supervision from the canonicalization operations in factorized semantic spaces, from which the presented framework can be trained easily with web-crawled videos only given no annotations. To sum up, our 3D motion retargeting approach enables both training and inference in-the-wild. The experiment results show improvement of the proposed method over the existing methods, especially in terms of robustness and generalization.

Acknowledgement. This study is partially supported under the RIE2020 Industry Alignment Fund Industry Collaboration Projects (IAF-ICP) Funding Initiative, as well as cash and in-kind contribution from the industry partner(s). We thank Xiaoxuan Ma, Shikai Li, Jiajun Su and Peizhuo Li for insightful discussion and kind support.
Novotny, D.; Ravi, N.; Graham, B.; Neverova, N.; and Vedaldi, A. 2019. C3DPO: Canonical 3D Pose Networks for Non-Rigid Structure From Motion. In ICCV. 3

Ott, C.; Lee, D.; and Nakamura, Y. 2008. Motion capture based human motion recognition and imitation by direct marker control. In IEEE-RAS. 1

Paszke, A.; Gross, S.; Massa, F.; Lerer, A.; Bradbury, J.; Chanan, G.; Killeen, T.; Lin, Z.; Gimelshein, N.; Antiga, L.; Desmaison, A.; Köpf, A.; Yang, E.; DeVito, Z.; Raison, M.; Tejani, A.; Chilamkurthy, S.; Steiner, B.; Fang, L.; Bai, J.; and Chintalal, S. 2019. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In Wallach, H.; Larochelle, H.; Beygelzimer, A.; d’Alché-Buc, F.; Fox, E.; and Garnett, R., eds., Advances in Neural Information Processing Systems 32, 8024–8035. Curran Associates, Inc. 5

Pavllo, D.; Feichtenhofer, C.; Grangier, D.; and Auli, M. 2019. 3D human pose estimation in video with temporal convolutions and semi-supervised training. In CVPR. 1, 6

Pickup, D.; Liu, J.; Sun, X.; Rosin, P. L.; Martin, R. R.; Cheng, Z.; Lian, Z.; Nie, S.; Jin, L.; Shamai, G.; Sahillioglu, Y.; and Kavan, L. 2018. An evaluation of canonical forms for non-rigid 3D shape retrieval. Graphical Models, 97: 17 – 29. 3

Pickup, D.; Sun, X.; Rosin, P. L.; and Martin, R. R. 2015a. Euclidean-distance-based canonical forms for non-rigid 3D shape retrieval. Pattern Recognition, 48(8): 2500 –2512. 3

Pickup, D.; Sun, X.; Rosin, P. L.; and Martin, R. R. 2016. Skeleton-based canonical forms for non-rigid 3D shape retrieval. Comput. Vis. Media, 2(3): 231–243. 3

Pickup, D.; Sun, X.; Rosin, P. L.; Martin, R. R.; Cheng, Z.; Nie, S.; and Jin, L. 2015b. Canonical Forms for Non-Rigid 3D Shape Retrieval. In Eurographics Workshop. 3

Rosenberg, A.; and Hirschberg, J. 2007. V-Measure: A Conditional Entropy-Based External Cluster Evaluation Measure. In EMNLP-CoNLL. 7

Saito, S.; Simon, T.; Saragih, J.; and Joo, H. 2020. PIFuHD: Multi-Level Pixel-Aligned Implicit Function for High-Resolution 3D Human Digitization. In CVPR. 3

Shamai, G.; Zibulevsky, M.; and Kimmel, R. 2015. Accelerating the Computation of Canonical Forms for 3D Nonrigid Objects using Multidimensional Scaling. In Pratikakis, I.; Spagnuolo, M.; Theoharis, T.; Gool, L. V.; and Veltkamp, R., eds., Eurographics Workshop. 3

Tak, S.; and Ko, H. 2005. A physically-based motion retargeting filter. ACM Trans. Graph., 24: 98–117. 2

Tulyakov, S.; Liu, M.; Yang, X.; and Kautz, J. 2018. MoCoGAN: Decomposing Motion and Content for Video Generation. In CVPR. 3

Villegas, R.; Yang, J.; Ceylan, D.; and Lee, H. 2018. Neural Kinematic Networks for Unsupervised Motion Retargetting. In CVPR. 2

Villegas, R.; Yang, J.; Hong, S.; Lin, X.; and Lee, H. 2017. Decomposing Motion and Content for Natural Video Sequence Prediction. In ICLR. 3

Wandt, B.; Rudolph, M.; Zell, P.; Rhodin, H.; and Rosenhahn, B. 2021. CanonPose: Self-Supervised Monocular 3D Human Pose Estimation in the Wild. In Computer Vision and Pattern Recognition (CVPR). 3

Wang, T.-C.; Liu, M.-Y.; Tao, A.; Liu, G.; Kautz, J.; and Catanzaro, B. 2019. Few-shot Video-to-Video Synthesis. In NeurIPS. 3

Wang, T.-C.; Liu, M.-Y.; Zhu, J.-Y.; Liu, G.; Tao, A.; Kautz, J.; and Catanzaro, B. 2018. Video-to-Video Synthesis. In NeurIPS. 3

Yang, W.; Ouyang, W.; Wang, X.; Ren, J.; Li, H.; and Wang, X. 2018. 3D human pose estimation in the wild by adversarial learning. In CVPR. 1

Yang, Z.; Zhu, W.; Wu, W.; Qian, C.; Zhou, Q.; Zhou, B.; and Loy, C. C. 2020. TransMoMo: Invariance-Driven Unsupervised Video Motion Retargeting. In CVPR. 2, 5, 6, 7, 8

Zhao, L.; Wang, Y.; Zhao, J.; Yuan, L.; Sun, J. J.; Schroff, F.; Adam, H.; Peng, X.; Metaxas, D.; and Liu, T. 2021. Learning View-Disentangled Human Pose Representation by Contrastive Cross-View Mutual Information Maximization. In CVPR, 12793–12802. 3

Zhao, Y.; Barnes, C.; Lu, J.; Yang, J.; and Li, H. 2019. On the continuity of rotation representations in neural networks. In CVPR. 4, 11
Table 4: **Encoder Structure.** We use three layers of 1-D temporal convolution for our encoders. The parameter of the convolution on abbreviated: $c$ denotes number of output channels, $k$ denotes kernel size, $p$ denotes padding, $s$ denotes stride and $pool$ denotes the type of pooling. The pooling is also performed on the temporal dimension.

### Appendix

**Network Architecture**

**Encoders.** The sizes of the latent representations are: motion $C_m = 128$, structure $C_s = 256$. We use an explicit 6D representation (Zhou et al. 2019) for view-angle, therefore $C_v = 6$. The temporal dimension is down-sampled by the motion encoders to an eighth of the original length, therefore $M = \frac{8}{7}$. The detailed network structure is shown in Table 4.

**Decoder.** The decoder is also a three-layer temporal convolutional network. The input channel is $C_m + C_s = 384$ and the number of output channels of each layer are 256, 128 and 45. The kernel size is 7.

**Evaluation Metrics**

For an inferred sequence $\hat{X}$ and a groundtruth sequence $X$,

\[
\text{MSE} = \frac{1}{3NT} \sum_{i,t} \frac{\|X_{i,t} - \hat{X}_{i,t}\|^2}{h^2(X)}
\]

\[
\text{MPJPE} = \frac{1}{NT} \sum_{i,t} \frac{\|X_{i,t} - \hat{X}_{i,t}\|}{h(X)}
\]

where $i$ is the subscript of body joints and $t$ is the subscript of time; $h(\cdot)$ calculates the height of a character by summing the lengths of the head, body and legs. This normalization is introduced to offset the effect that errors tend to be larger on larger characters. The errors are computed after hip-alignment so that the measurement is focused on the movements of different body parts local to the hip joint.