Unpaired Motion Style Transfer from Video to Animation

KFIR ABERMAN*, AICFVE, Bejing Film Academy & Tel-Aviv University
YIJIA WENG*, CFCS, Peking University & AICFVE, Beijing Film Academy
DANI LISCHINSKI, The Hebrew University of Jerusalem & AICFVE, Beijing Film Academy
DANIEL COHEN-OR, Tel-Aviv University & AICFVE, Beijing Film Academy
BAOQUAN CHEN†, CFCS, Peking University & AICFVE, Beijing Film Academy

Transferring the motion style from one animation clip to another, while preserving the motion content of the latter, has been a long-standing problem in character animation. Most existing data-driven approaches are supervised and rely on paired data, where motions with the same content are performed in different styles. In addition, these approaches are limited to transfer of styles that were seen during training.

In this paper, we present a novel data-driven framework for motion style transfer, which learns from an unpaired collection of motions with style labels, and enables transferring motion styles not observed during training. Furthermore, our framework is able to extract motion styles directly from videos, bypassing 3D reconstruction, and apply them to the 3D input motion.

Our style transfer network encodes motions into two latent codes, for content and for style, each of which plays a different role in the decoding (synthesis) process. While the content code is decoded into the output motion by several temporal convolutional layers, the style code modifies deep features via temporally invariant adaptive instance normalization (AdaIN).

Moreover, while the content code is encoded from 3D joint rotations, we learn a common embedding for style from either 3D or 2D joint positions, enabling style extraction from videos.

Our results are comparable to the state-of-the-art, despite not requiring paired training data, and outperform other methods when transferring previously unseen styles. To our knowledge, we are the first to demonstrate style transfer directly from videos to 3D animations - an ability which enables one to extend the set of style examples far beyond motions captured by MoCap systems.

CCS Concepts: • **Computing methodologies → Motion processing**; **Neural networks**.

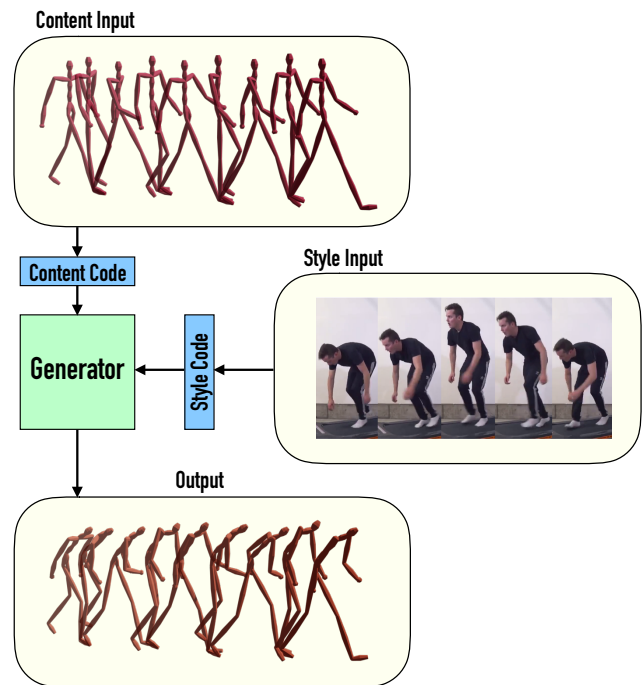Additional Key Words and Phrases: motion analysis, style transfer

Fig. 1. Style transfer from video to animation. Our network, which is trained with unpaired motion sequences, learns to disentangle content and style. Our trained generator is able to produce a motion sequence that combines the content of a 3D sequence with the style extracted directly from a video.

1 INTRODUCTION

The style of human motion may be thought of as the collection of motion attributes that convey the mood and the personality of a character. Human observers are extremely perceptive to subtle style variations; we can, for example, often tell whether a person is happy or sad from the way they walk. Consequently, for games

and movies that pursue realistic and expressive character animation, there is a long-standing interest in generating diverse stylized motions. However, capturing all desired motions in a variety of styles is practically infeasible. A much more promising option is to perform motion style transfer: modify the style of an existing motion into one taken from another. Furthermore, it is particularly attractive to use video clips to specify target motion styles.

Since motion style eludes a precise definition, hand crafted representations are not well-suited to cope with style transfer, and most recent works attempt to infer style from examples. However, despite years of progress in data-driven motion style transfer, two main obstacles remain in practice: (i) avoiding the need for paired and registered data, and (ii) extracting style from only a few examples. Both of these hurdles arise because of the difficulty to collect and process sufficient motion data. In order to capture paired and registered data, the same actor must, for example, perform a walking

---

*equal contribution
†corresponding author

Authors' addresses: Kfir Aberman, kfiraberman@gmail.com; Yijia Weng, halfsummer11@gmail.com; Dani Lischinski, danix3d@gmail.com; Daniel Cohen-Or, cohenor@gmail.com; Baoquan Chen, baoquan@pku.edu.cn.

sequence in different styles with identical steps and turns, which is tedious and, more importantly, unscalable to the huge training sets required by today's deep learning models. As for the extraction of styles, clearly, a large number of style examples might better characterize the style and facilitate the transfer. However, in reality, one can often obtain only a few examples for each (uncommon) style.

Additionally, we also see great potential in videos as a massive source for motion styles. As the primary format for recording human activities, available videos contain a much wider range of motions and styles compared to 3D motion capture data. And if the desired style needs to be captured on-the-spot, shooting a video is a much easier and cheaper alternative to performing motion capture.

As an important research topic, the problem of motion style transfer has been approached in several ways, all of which we find limited, especially with regard to the challenges above. Some works model style using hand-crafted representations, such as physical parameters or spectral characteristics, which may fail to fully capture complex and subtle properties. Other works adopt a data-driven approach. Their reliance on paired training data or large numbers of target style examples, however, hinders their applicability to real world settings. Recently, Mason et al. [2018] proposed a few-shot learning scheme to cope with the shortage of style references. However, they employ a specialized network that targets only locomotion. Also, like other previous works, their model can only extract style from 3D MoCap data.

In this work, we circumvent the need for paired training data by learning without supervision, characterize unseen styles even from a single reference clip, and provide the ability to extract style from a video. To achieve these goals, we adopt a generative scheme, using temporal convolutional neural networks as the backbone of our model. Our network encodes content and style inputs into corresponding latent codes, which are then recombined and decoded to obtain a re-stylized result (see Figure 1). We argue that during training, our network learns a universal style extractor, which is then applicable to new styles, specified via a few examples at test time. Furthermore, our style extractor is applicable to both 3D motions and 2D motions observed in ordinary video examples, without requiring 3D reconstruction.

Our main technical contribution lies in the architecture of the deep neural network outlined above, in which the content and style codes affect the generated motions via two different mechanisms. The content code is decoded into a motion by applying a sequence of temporal convolutional layers, each resulting in a set of temporal signals that represent joint rotations in a high-dimensional feature space. The style code is used to modify the second order statistics of these generated deep features via temporally-invariant adaptive instance normalization (AdaIN). These temporally-invariant affine transformations amplify or attenuate the temporal signals, while preserving their shape. Consequently, the motion content is also preserved. AdaIN has been used with great effect in image generation and style transfer (see Section 2.1), but to our knowledge we are the first to apply it in the context of motion.

Our network is trained by optimizing a *content consistency loss*, which ensures that the content input to our network is reconstructed whenever the style input has the same style label as the content one.

This loss forces the network to extract only those attributes that are shared among samples of the same style class. Simply copying the content input to the output is prevented by instance normalization during the encoding and by restricting the dimensionality of the latent content code.

In order to extract style from videos we learn a joint embedding space for style codes extracted from either 3D or 2D joint positions. During training, we require that 3D motions and their 2D projections, are both mapped by a pair of corresponding encoders into the same style code. In addition to enabling style extraction from videos, this joint embedding supports style interpolation, as well as measuring "style distance" between videos and/or 3D motions.

In summary, our contributions consist of a novel data-driven approach for motion style transfer that: (i) does not require paired training data; (ii) is able to transfer styles unseen during training, extracted from as little as a single example clip; and (iii) supports style extraction directly from ordinary videos.

We discuss various insights related to the mechanism of the network and the analogy to some related works. Our results show that by leveraging the power of our new framework, we can match state-of-the-art motion style transfer results, by training only on unpaired motion clips with style labels. Furthermore, we outperform other methods for previously unseen styles, as well as styles extracted from ordinary videos.

2 RELATED WORK

2.1 Image Style Transfer

Our work is inspired by the impressive progress in image style transfer, achieved through the use of deep learning machinery. The pioneering work of Gatys et al. [2016] showed that style and content can be represented by statistics of deep features extracted from a pre-trained classification network. While their original approach required optimization to transfer style between each pair of images, Johnson et al. [2016] later converted this approach to a feed-forward one by training a network using a perceptual loss.

Later on, Ulyanov et al. [2016] showed that the style of an image can be manipulated by modifying the second order statistics (mean and variance) of channels of intermediate layers and proposed an instance normalization layer which enables to train a network to modify the style of arbitrary content images into a single specific target style. This idea was further extended by Huang et al. [2017], who demonstrated that various target styles may be applied simply by using the Adaptive Instance Normalization (AdaIN) layer to inject different style statistics into the same network.

The AdaIN mechanism has proved effective for various tasks on images, such as image-to-image translation [Huang et al. 2018] and image generation [Karras et al. 2019]. Recently, Park et al. [2019] proposed a spatially adaptive normalization layer, for multi-modal generation of images, based on semantic segmentation maps, while Liu et al. [2019] introduced FUNIT, a few-shot unpaired image-to-image translation method, where only a few examples of the target class are required.

Inspired by these recent achievements in image style transfer, our work makes use of temporally invariant AdaIN parameters, thus enabling manipulating a given motion sequence to perform in

arbitrary styles. To our knowledge, we are the first to employ such a mechanism in the context of motion processing.

2.2 Motion style transfer

Motion style transfer is a long standing problem in computer animation. Previous works relied on handcrafted features, in frequency domain [Unuma et al. 1995], or in time [Amaya et al. 1996], to represent and manipulate style or emotion [Aristidou et al. 2017] of given 3D motions, or used physics-based optimizations [Liu et al. 2005] to achieve a similar goal. Yumer and Mitra [2016] showed that difference in spectral intensities of two motion signals with similar content but different styles enables the transfer between these two styles on arbitrary heterogeneous actions.

Since style is an elusive attribute that defies a precise mathematical definition, data-driven approaches that infer style features from examples, might have an advantage compared to attempting to hand-craft features to characterize style and content. Indeed, several works used machine learning tools to perform motion style transfer [Brand and Hertzmann 2000; Hsu et al. 2005; Ikemoto et al. 2009; Ma et al. 2010; Wang et al. 2007; Xia et al. 2015]. However, these methods either assume that there are explicit motion pairs in the training data that exhibit the same motion with different styles, or limited to the set of styles given in the dataset.

For example, Hsu et al. [2005] learned a linear translation model that can map a motion to a target style based on pairwise dense correspondence (per frame) between motions with similar content but different styles. Xia et al. [2015] used a KNN search over a database of motions to construct a mixture of regression models for transferring style between motion clips, and Smith et al. [2019] improved the processing using a neural network that is trained on paired and registered examples. Both latter methods represent style by a vector of fixed length, which is limited to the set of styles in the dataset, and can not be applied to unseen styles. In contrast, our approach only assumes that each motion clip is labeled by its style, without any requirement for content labels or correspondences between motion clips. In addition, our method enables extraction of unseen styles from 3D motion examples, and even from video, and is not limited to the styles in the dataset.

Besides transferring motion style, other methods exploited machine learning approaches to cope with various closely related tasks. These include generating motion with constraints using inverse kinematics (IK) when the style is taken from a dataset [Grochow et al. 2004], performing independent component analysis to separate motion into different components and perform a transfer [Shapiro et al. 2006], or using restricted Boltzmann machines, conditioned on a style label, to model human motion and capture style [Taylor and Hinton 2009].

With the recent rapid progress in deep learning methods for character animation [Holden et al. 2017b, 2016, 2015], the flourishing image style transfer techniques were quickly adopted into the character animation domain. Holden et al. [2016], proposed a general learning framework for motion editing that enables style transfer. In analogy to Gatys et al. [2016], they optimize a motion sequence that satisfy two conditions; the activations of the hidden units should be similar to those of the content motion, while their Gram matrices should match those of the style input motion. Differently from Gatys et al. [2016], here the features are extracted using a pretrained autoencoder for motion, rather than a pretrained image classification network. Later on, Holden et al. [2017a] and Du et al. [2019] proposed to improve performance by replacing optimization with a feed-forward network that is trained to satisfy the same constraints. In both of these works, the pretrained network is not explicitly designed for style transfer. The features extracted by the pretrained autoencoder contain information on both content and style, which leads to a strong dependency between the two properties, as demonstrated in Section 5.

Recently, Mason et al. [2018] proposed a method to transfer the style of character locomotion in real time, given a few shots of another stylized animation. In contrast to this approach, which is limited to locomotion, our approach can extract and transfer styles regardless of the motion content.

2.3 Motion from Videos

Motion reconstruction and pose estimation from monocular videos are long-standing fundamental tasks in computer vision, and are beyond the scope this paper. Along the years, various methods have dealt with the extraction of different motion properties directly from video, such as 2D poses [Cao et al. 2018], 3D poses [Pavllo et al. 2019b], and 3D motion reconstruction [Mehta et al. 2017]. Recently, Aberman et al. [2019b] extracted character-agnostic motion, view angle, and skeleton, as three disentangled latent codes, directly from videos, bypassing 3D reconstruction. Existing methods that extract motion from videos [Kanazawa et al. 2019; Mehta et al. 2017] are typically not concerned with style, while our work is the first to extract the style, rather than the motion, from video-captured human motion samples, bypassing 3D reconstruction.

Another stream of work is focused on motion transfer in video using deep learning techniques. The existing approaches provide a different perspective on how to extract motion in 2D [Aberman et al. 2019a; Chan et al. 2019] or 3D [Liu et al. 2018] from one video, and apply to it the appearance of a target actor from another video.

3 MOTION STYLE TRANSFER FRAMEWORK

Our framework aims at translating a motion clip of an animated character with a given *content* to another motion that exhibits the same content, but performed using a different *style*. The desired *target style* may be inferred from a few (or even a single) 3D motion clips, or a video example. Importantly, the target style might not be one of those seen during training. We only assume that each of the motion clips in the training set is assigned a style label; there's no pairing requirement, i.e., no need for explicit pairs of motions that feature the same content performed using two different styles.

We treat style transfer as a conditional translation model, and propose a neural network that learns to decompose motion into two disentangled latent codes, a temporal latent code that encodes motion content, and a temporally-invariant latent code that encodes motion style. These two codes affect the generated motions via two different mechanisms. The content code is decoded into a motion by applying a sequence of temporal convolutional layers, each yielding
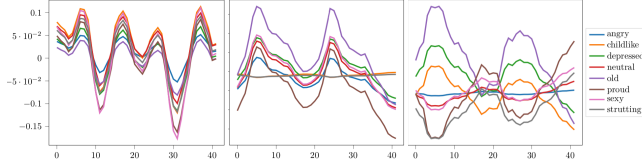
Fig. 2. Visualization of deep features in our decoder. Each plot visualizes a specific channel of a deep feature as eight different styles are applied to the same motion content. It can be seen that the signals differ only by a temporally-invariant affine transform and that the signal shape (per channel) is preserved. Thus, the motion content is preserved as well.

a set of temporal signals that determine the joint rotations in a high-dimensional feature space. The style code is used to modify only the means and variances of the generated temporal deep features via temporally-invariant adaptive instance normalization (AdaIN), thereby preserving the content of the original motion. Figure 2 visualizes three channels of deep features generated by our network's decoder, showing the effect of different styles applied to the same motion content.

In addition, in order to support style extraction from videos, we learn a joint embedding space for style by extracting style codes from either 3D or 2D joint positions, using two different encoders, which are encouraged to map 3D motions and their 2D projections into the same code.

Figure 3 describes the high-level architecture of our framework, whose various components are described in more detail below.

3.1 Architecture

Our framework, depicted in Figure 3, consists of a conditional motion translator that takes as input two motion clips: the content motion $\mathbf{m}^s$, with a source style $s \in \mathcal{S}$, as well as a style motion $\mathbf{n}^t$, with a target style $t \in \mathcal{S}$. The output motion $\tilde{\mathbf{m}}^t$ is supposed to consist of the content of $\mathbf{m}^s$, performed using style t. We next describe the design and role of the different components in our framework.

Motion Representation. In our setting, a motion clip $\mathbf{m} \in \mathbb{R}^{T \times d}$ is a temporal sequence of T poses where each pose is represented by d channels. We choose to represent the content input $\mathbf{m}^s$ and the style input $\mathbf{m}^t$ differently from each other. Since a motion is well defined by the joint rotations (as opposed to joint positions), and since the content input is strongly correlated with the output motion, we represent $\mathbf{m}^s$ using rotations (unit quaternions), i.e., $\mathbf{m}^s \in \mathbb{R}^{T \times 4J}$, where J is the number of joints. In contrast, since style can be inferred from the relative motion of joint positions, and to facilitate learning a joint embedding space for 3D and 2D motions, we represent the style input using joint positions ($\mathbf{n}^t \in \mathbb{R}^{T \times 3J}$). The output motion $\tilde{\mathbf{m}}^t$ is represented using joint rotations, which enables the extraction of standard character animation files without any further post-processing. Note that the global root positions are discarded from the representation of the network's input/output, and are treated separately during test time, as explained later in this section.

Motion Translator. The motion translator consists of a content encoder E_C, a style encoder E_S, and a decoder F, where E_C and

E_S encode the input into two latent codes, $\mathbf{z}_c$ and $\mathbf{z}_s$, respectively. E_C consists of several temporal, 1D convolutional layers [Holden et al. 2015], followed by several residual blocks that map the content motion to a temporal content latent code $\mathbf{z}_c$. During encoding, the intermediate temporal feature maps undergo instance normalization (IN), which effectively ensures that the resulting content code is "stripped" of style.

The encoding of style is performed by one of the two encoders E_S^{2D}, E_S^{3D}, depending on whether the joint coordinates are in 2D (extracted from a video) or in 3D. We assume that the style does not change in mid-motion, and use a sequence of 1D convolutional layers to map $\mathbf{n}^t$ into a fixed-size (independent on the temporal length of the clip) latent code $\mathbf{z}_s$. We'd like 2D and 3D motions performed with the same style to be mapped to the same latent vector $\mathbf{z}_s$. Thus, at each iteration during training we use a 3D motion clip, along with a 2D perspective projection of that same clip, and feed both clips into the corresponding encoders. The latent code $\mathbf{z}_s$ is then obtained by averaging the output of the two encoders. At test time, the style code is extracted by only one of the two encoders, depending on the type of style input.

The decoder F consists of several residual blocks with adaptive instance normalization (AdaIN) [Huang and Belongie 2017], followed by convolutional layers with stride that upsample the temporal resolution of the clip. The AdaIN layer constitutes a normalization layer that applies an affine transform to the feature activations (per channel). For each AdaIN layer with c channels, the network learns a mapping (a multilayer perceptron, or MLP) of the style code $\mathbf{z}_s$ into $2c$ parameters that modify the per-channel mean and variance.

Note that the affine transformation is temporally invariant and hence only affects non-temporal attributes of the motion. Thus, while the content encoder effectively removes the source style s by normalizing the non-temporal attributes with IN, the decoder injects the target style t by using AdaIN to scale and shift the feature channels to target values inferred from the style code.

In summary, using the notation introduced above, our conditional motion translator G may be formally expressed as:

$$\tilde{\mathbf{m}}^t = G\left(\mathbf{m}^s|\mathbf{n}^t\right) = F\left(E_C(\mathbf{m}^s)|E_S(\mathbf{n}^t)\right). \tag{1}$$

Multi-Style Discriminator. Our discriminator D follows the multi-class discriminator baseline proposed at [Liu et al. 2019]. D is a single component that is trained to cope with $|\mathcal{S}|$ adversarial tasks simultaneously, where each task aims to determine whether an input motion is a real motion of a specific style $i \in \mathcal{S}$, or a fake output of G. When updating D for a real motion of source style $i \in \mathcal{S}$, D is penalized if its i-th output is false. For a translation output yielding a fake motion of source style i, D is penalized if the i-th output is positive. Note that D is not penalized for not predicting false for motions of other styles. When updating G, it is penalized only if the i-th output of D is false.

Global Velocity. At test time, we aim to extract the desired style even from a single example. However, since the content of the two input motions (style and content) may be different, the translation of global velocity, a property which is often correlated with style, is a challenging task. For example, in order to convert neutral walking to old walking, the global velocity should decrease. However,
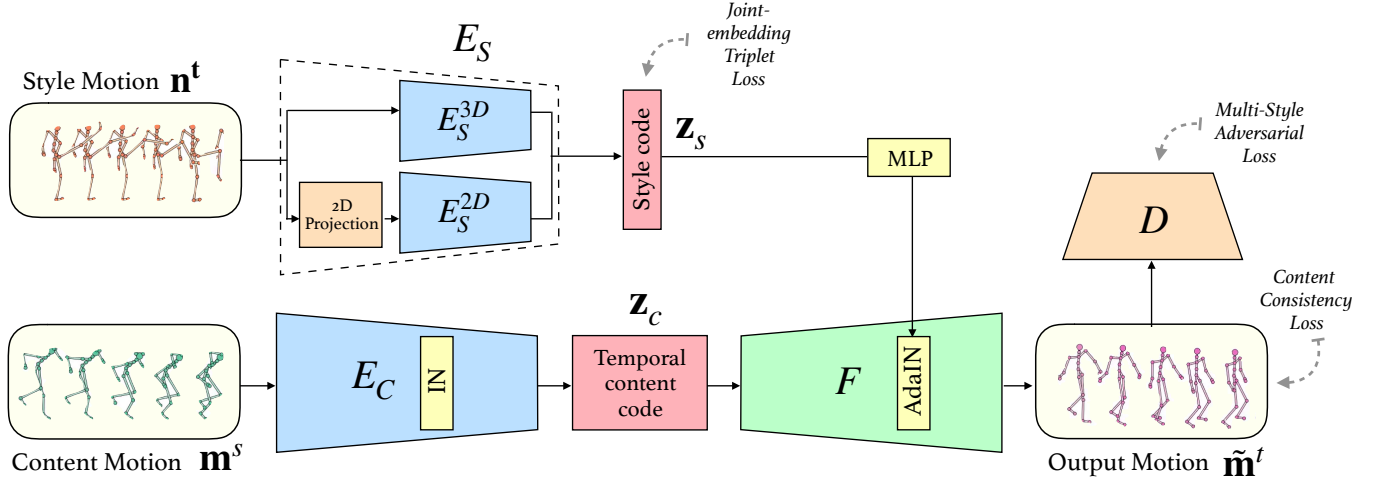
Fig. 3. Our motion-style transfer architecture consists of encoders for content (E_C) and style (E_S), which extract a latent content code ($\mathbf{z}_c$) and a latent style code ($\mathbf{z}_s$), respectively, where the style code can be extracted from either 3D motions (via E_S^{3D}) or 2D projections (E_S^{2D}). During encoding, the content code is stripped of style by instance normalization (IN) layers. The content code is then used to reconstruct a motion by a decoder F, which contains AdaIN layers that modify temporally-invariant second order statistics of the intermediate deep features decoded by F. The output motions are fed into a multi-style discriminator D that judges whether it belongs to a certain style.

inferring such a property form a single clip of old kicking, where the root position is nearly static, is practically impossible, especially when the style is previously unseen. A principled solution to this problem is outside the scope of this work; below, we describe a heuristic solution that has worked well in our experiments. In this solution, the root positions of our output motion are directly taken from the content input sequence. However, since global velocity is correlated with style, we perform a dynamic time warping on the global velocity, based on the velocity ratio between the two input motions. More precisely, for each motion sequence, we measure the velocity factor as the temporal average of the maximal local joint velocity,

$$V = \frac{1}{T} \sum_{\tau=1}^{T} \max_{j \in J} \{v^j(\tau)\}, \qquad (2)$$

where $v^j(\tau)$ is the local velocity of the j-th joint in time τ. Next, we warp the temporal axis by the factor $V^{\text{sty}}/V^{\text{con}}$, where V^{sty} and V^{con} are the velocity factors for the style and content inputs, respectively. We find that in most of our examples local joint velocity captured style information better than global velocity, and for that reason decided to use the above definition for the velocity factor.

Note that this global transformation is reversible, such that if we use the output motion as the content input, and the original content motion as the style input, we recover the global velocity of the original motion.

Foot Contact. As our network is built upon 1D temporal convolution layers, raw outputs tend to suffer from foot skating artifacts. In order to cope with the issue, we extract foot contact labels from the content input, use them to correct the feet positions and apply IK to fix the corresponding output poses (before the global velocity warping). As a result, we get visually plausible outputs, with no foot skating. While this fix works well in most cases, note that it

assumes that the foot contact timing is part of the content, and not of the style. However, this is not always true: consider, for example, a zombie walking while dragging one of the feet. This aspect should be further addressed in future work.

3.2 Training and Loss

Our dataset is trimmed into short overlapped clips, which comprise our motion collection $\mathcal{M}$. However, note that at test time the length of the input sequences can be arbitrary, since the networks are fully convolutional.

Note that although the content input motion and the output motion are both represented using joint rotations, all of the losses are applied to joint positions as well. This is done by applying a forward kinematics layer [Pavllo et al. 2019a; Villegas et al. 2018] on the aforementioned components, which for simplicity, is not explicitly mentioned in the following equations.

Content Consistency Loss. In case that the content input $\mathbf{m}^s$ and the style input $\mathbf{n}^t$ share the same style ($t = s$), it is expected that the translator network will constitute an identity map, regardless of the content of $\mathbf{n}^t$. Thus, in every iteration we randomly pick two motion sequences from our dataset $\mathcal{M}$, with the same style label, and apply the content consistency loss which is given by

$$\mathcal{L}_{\text{con}} = \mathbb{E}_{\mathbf{m}^s, \mathbf{n}^s \sim \mathcal{M}} \| F\left(E_C(\mathbf{m}^s) | E_S(\mathbf{n}^s)\right) - \mathbf{m}^s \|_1, \qquad (3)$$

where $\|\cdot\|$ represents the L_1 norm. Note that when $\mathbf{n}^s = \mathbf{m}^s$, Equation (3) becomes a standard reconstruction loss.

Adversarial Loss. Since our training is unpaired, the adversarial loss is the component which is responsible, in practice, to manipulate

the style of $\mathbf{m}^s$ via

$$
\begin{aligned}
\mathcal{L}_{\text{adv}} = \quad & \mathbb{E}_{\mathbf{n}^t \sim \mathcal{M}} \| D^t(\mathbf{n}^t) \|^2 \\
+ \quad & \mathbb{E}_{\mathbf{m}^s, \mathbf{n}^t \sim \mathcal{M} \times \mathcal{M}} \| D^t(F(E_C(\mathbf{m}^s) | E_S(\mathbf{n}^t)) - 1 \|^2,
\end{aligned} \tag{4}
$$

where $D^t(\cdot)$ represents the discriminator output which corresponds to the style class $t \in \mathcal{S}$. In order to stabilize the training of the generator in the multi-class setting, we apply feature matching loss as regularization [Wang et al. 2018]. This loss minimizes the distance between the last feature of the discriminator when fed by a real input of a specific style (averaged over the set) to the same feature when fed by a fake output of the same target style, via

$$
\mathcal{L}_{\text{reg}} = \mathbb{E}_{\mathbf{m}^s, \mathbf{n}^t \sim \mathcal{M} \times \mathcal{M}} \| D_f(\tilde{\mathbf{m}}^t) - \frac{1}{|\mathcal{M}_t|} \sum_{i \in \mathcal{M}_t} D_f(\mathbf{n}_t^i) \|_1, \tag{5}
$$

where $\mathcal{M}_t$ is a subset of motions with style $t \in \mathcal{S}$ and D_f is a sub-network of D that doesn't include the prediction (last) layer.

3.3 Joint 2D-3D Style Embedding

Intuitively, the style of motion can be identified by observing the character 3D positions in space-time as well as their 2D projections using a view with a reasonable elevation angle. While handcrafted representations are designed to treat a single form of input, we exploit the power of deep learning to learn a joint embedding space for extracting style from both 3D and 2D motion representations. The learned common latent space can be used for various tasks, such as video-based motion style retrieval, style interpolation and more. In our context, we exploit the properties of this space to extract style directly from real videos during test time, while bypassing the need for 3D reconstruction, an error-prone process which adds noise into the pipeline.

Joint Embedding Loss. In order to construct a common latent space for style, our loss encourages pairs of 3D-2D motions to be mapped into the same feature vector by

$$
\mathcal{L}_{\text{joint}} = \mathbb{E}_{\mathbf{n}^t \sim \mathcal{M}} \| E_S^{3D}(\mathbf{n}^t) - E_S^{2D}(P(\mathbf{n}^t; p)) \|^2, \tag{6}
$$

where P is a weak perspective projection operator that projects the input to a camera plane, with camera parameters p, that consist of scale s and the Euler angles $v = (v^{\text{pitch}}, v^{\text{yaw}}, v^{\text{roll}})$. For each motion clip, we define the local Z-axis to be the temporal average of per-frame forward directions, which are computed based on the cross product of the Y-axis and the average of vectors across the shoulders and the hips. During training $v^{\text{roll}} = v^{\text{pitch}} = 0$ are fixed while $v^{\text{yaw}} \in [-90°, 90°]$ and $s \in [0.8, 1.2]$ are randomly sampled five times in each iteration, to create five projections that are transferred to E_S^{2D}.

Style Triplet Loss. In order to improve the clustering of the different styles in the latent space, we exploit the style labels and use the technique suggested by Aristidou et al. [2018] to explicitly encourage inputs with similar style to be mapped tightly together, by applying a triplet loss on the style latent space via

$$
\begin{aligned}
\mathcal{L}_{\text{trip}} = \quad & \mathbb{E}_{\mathbf{n}^t, \mathbf{x}^t, \mathbf{w}^s \sim \mathcal{M}} [\| E_S(\mathbf{n}^t) - E_S(\mathbf{x}^t) \| - \\
& \| E_S(\mathbf{n}^t) - E_S(\mathbf{w}^s) \| + \delta]_+,
\end{aligned} \tag{7}
$$

where $\mathbf{x}^t$ and $\mathbf{w}^s$ are two motions with different styles $s \neq t$, and $\delta = 5$ is our margin. This loss encourages the distance between

feature vectors of two motion inputs that share the same style to be smaller, at least by α, than the distance between two motions with different styles.

Our final loss is given by a combination of the aforementioned loss terms:

$$
\mathcal{L} = \mathcal{L}_{\text{con}} + \alpha_{\text{adv}} \mathcal{L}_{\text{adv}} + \alpha_{\text{reg}} \mathcal{L}_{\text{reg}} + \alpha_{\text{joint}} \mathcal{L}_{\text{joint}} + \alpha_{\text{trip}} \mathcal{L}_{\text{trip}}, \tag{8}
$$

where in our experiments we use $\alpha_{\text{adv}} = 1$, $\alpha_{\text{reg}} = 0.5$, $\alpha_{\text{joint}} = 0.3$ and $\alpha_{\text{trip}} = 0.3$. The network is trained to minimize $\mathcal{L}$ with the parameters of E_S, E_C, F and to maximize it with the parameters D.

4 DISCUSSION

As discussed in Section 2, adjusting the mean and variance of deep feature channels in a neural network proved to be effective for manipulating the style of 2D images. Although motion style is a conceptually and visually different notion, we have shown that a similar mechanism can be used to manipulate the style of motion sequences.

Below we show that our technique may be seen as a generalization of the style transfer technique of Yumer and Mitra [2016]. Their technique is based on pairs, while ours is unpaired, and uses learning to deal with unseen styles. However, we present a derivation that shows the commonality in the building blocks, the analysis of the motion, and how style is transferred. These commonalities contribute to the understanding of our method.

In their work, Yumer and Mitra [2016] propose a method for motion style transfer in the frequency domain. They show that given two motions $\mathbf{y}^s$ and $\mathbf{y}^t$ with similar content and different styles $s \neq t$, a new, arbitrary, motion $\mathbf{x}^s$ with style s can be transferred to style t. The transfer result is given in the frequency domain by

$$
\tilde{\mathbf{x}}^t(\omega) = |\tilde{\mathbf{x}}^t(\omega)| e^{i \angle \tilde{\mathbf{x}}^t(\omega)}, \tag{9}
$$

where the magnitude of the output is given by

$$
|\tilde{\mathbf{x}}^t(\omega)| = |\mathbf{x}^s(\omega)| + |\mathbf{y}^t(\omega)| - |\mathbf{y}^s(\omega)|
$$

and the phase function is taken from the original input signal $\angle \tilde{\mathbf{x}}^t(\omega) = \angle \mathbf{x}^s(\omega)$. Applying the inverse Fourier transform to Equation (9), we get

$$
\tilde{\mathbf{x}}^t(\tau) \approx \mathbf{x}^s(\tau) + g^{s \rightarrow t}(\tau) * \mathbf{x}^s(\tau), \tag{10}
$$

where $g^{s \rightarrow t}(\tau)$ is a convolution kernel (in time, τ) whose weights depend on the source and target styles.

Equation (10) implies that the approach of Yumer and Mitra [2016] may be implemented using a convolutional residual block to modify the style of a motion sequence, where the target style is defined by the weights of the kernel $g^{s \rightarrow t}$, which depend on the source style s and well as the target style t.

Similarly, our style translation framework also uses convolutional residual blocks as its core units, where the kernel weights effectively depend on the source and target style. Although the kernels are fixed at test time, their weights are effectively modified by the IN and AdaIN layers, as a function of the styles s and t. Convolution in time (with kernel $k(\tau)$ and bias b) followed by an IN/AdaIN layer can be expressed as:

$$
\tilde{\mathbf{x}}(\tau) = \beta [\mathbf{x}(\tau) * k(\tau) + b] + \gamma = \mathbf{x}(\tau) * \beta k(\tau) + \beta b + \gamma, \tag{11}
$$

where β and γ, are the IN or AdaIN parameters. Equation (10) is, in fact, equivalent to a single convolution with an effective kernel weights $\hat{k}(\tau) = \beta k(\tau)$ and bias $\hat{b} = \beta b(\tau) + \gamma$, that are modified as a function of some input. For the IN case, β and γ depend on the input signal $\mathbf{x}^s$, since they are calculated such that the output has zero mean and unit variance. In the AdaIN case, the parameters are produced by an MLP layer as a mapping of the target style's latent code. Thus, the use of IN and AdaIN in our architecture effectively controls the convolutions by the source and target styles.

Rather than directly transferring the motion style from s to t with a single convolution, as in Equation (10), our process may be viewed as consisting of two steps: first, the source style is removed by the encoder E_C, which depends only on s, and then the target style is applied by the decoder F, via the AdaIN layers, which depend only on t. Thus, our network performs style transfer in a modular way, making it easier to accommodate new styles. In contrast, using a single kernel makes it necessary to learn how to translate each style in $\mathcal{S}$ to every other style (all pairs).

5 EXPERIMENTS AND EVALUATION

In this section we evaluate our method and perform various experiments and comparisons that demonstrate some interesting insights and interpretation of our style transfer mechanism.

Firstly, several samples of our style transfer results are shown in Figure 4. The full motion clips are included in the accompanying video. Note that our network outputs joint rotations, hence, our results do not require any further processing such as IK, and can be directly converted to the commonly used motion representation files, and visualized. Although our joint rotations are represented by unit quaternions, which may lead to discontinuities within neural networks [Zhou et al. 2019], our output quaternions tend to be smooth due to the temporal 1D convolutions performed by our network. Our results demonstrate that our system can transfer styles that are extracted from various sources, such as 3D animated characters, 2D projection of 3D motions and real videos, within a unified framework. The examples demonstrating style transfer from a video use only a short (3-second) video clip as the sole (and previously unseen) style example. We are not aware of any other style transfer method with this capability.

Implementation Details. We used two different datasets to perform our experiments. The first dataset, supplied by Xia et al. [2015], contains motion sequences that are labeled with eight style labels. The second, is our own newly captured dataset, which contains various motions performed by a single character in 16 distinct styles. For convenience, we refer to the datasets as A and B, respectively. The motion sequences within each of the datasets are trimmed into short overlapping clips of $T = 32$ frames with overlap of $T/4$, resulting in about 1500 motion sequences for dataset A and 10500 for B. In addition, the motions in each dataset are split into two disjoint train and test sets, with the test set consisting of 10% of the samples.

Our framework is implemented in PyTorch and optimized by the Adam optimizer. A training session takes about about 8 hours for dataset A, and double the time for dataset B, using an NVIDIA GeForce GTX Titan Xp GPU (12 GB).

5.1 Latent Space Visualization

In this experiment we project the content codes and style parameters of some random motion samples from dataset A onto a 2D space by using t-distributed stochastic neighbor embedding (t-SNE), and plot the results in order to gain a better understanding of how the network interprets content and style in practice.

Style Code. Figure 5 shows the 2D projection of our style parameters (AdaIN), where each sample is marked with a color corresponding to its style label. It can be seen that our network learns to cluster the style parameters, which means that style inputs that share the same style will manipulate the motion content in a similar way. This result demonstrates that the extracted style parameters mostly depend on the style label.

As previously discussed, our framework treats style as a set of properties, shared by motions in a group, which can be manipulated (added/removed) by an affine, temporally invariant, transformation (AdaIN) applied to deep features. When such common properties exist within the group, the clusters are naturally formed even without the need for triplet loss (Figure 5(a)). However, since a given style may be described by different nuances for different content motions (e.g., proud boxing has some hand gestures that do not exist in proud walking), a triplet loss encourages (but does not enforce) style codes of the same group to be closer to each other. This loss emphasizes commonalities within the group, making the clusters tighter, as can be observed in Figure 5(b), and leads to better content-style disentanglement.

Figure 6 visualizes style codes parameters (AdaIN) extracted from 3D motions together with ones extracted from video. It may be seen that the latter codes, for the most part fall into the same clusters as the former ones.

Unseen Styles. Generally speaking, our network enables to extract styles from arbitrary motion clips during test time. However, in practice, when the number of seen styles is small, the network may overfit to the existing styles, as one might suspect when observing the well-separated clusters in Figure 5. We retrained our model with dataset A, excluding the motions that are labeled by the "old" style label, and then tested it using the motions within this group. Although the network successfully clusters the samples (Figure 7(a)), our results show that the style properties of the output are adapted from visually similar styles among those that were seen during training. For example, the "old walking" style code is close that that of "depressed walking", and the style transfer result indeed resembles that of "depressed walking".

We performed the same experiment with dataset B (which includes 16 styles), excluding the "heavy" style, and then tested the trained system with these motions. As can be seen in Figure 7 (b), the network again learns to cluster the test samples by their style labels. However, in this case, the output motions successfully adapted style properties from the new unseen clip, which means that the overfitting is significantly reduced when training on dataset B. This demonstrates that for the same framework with fixed dimensions, style can be generalized better, when there are more style classes, and that the network learns to identify properties that may be related to style, and to extract them even from unseen style inputs.

(a)   (b)   (c)
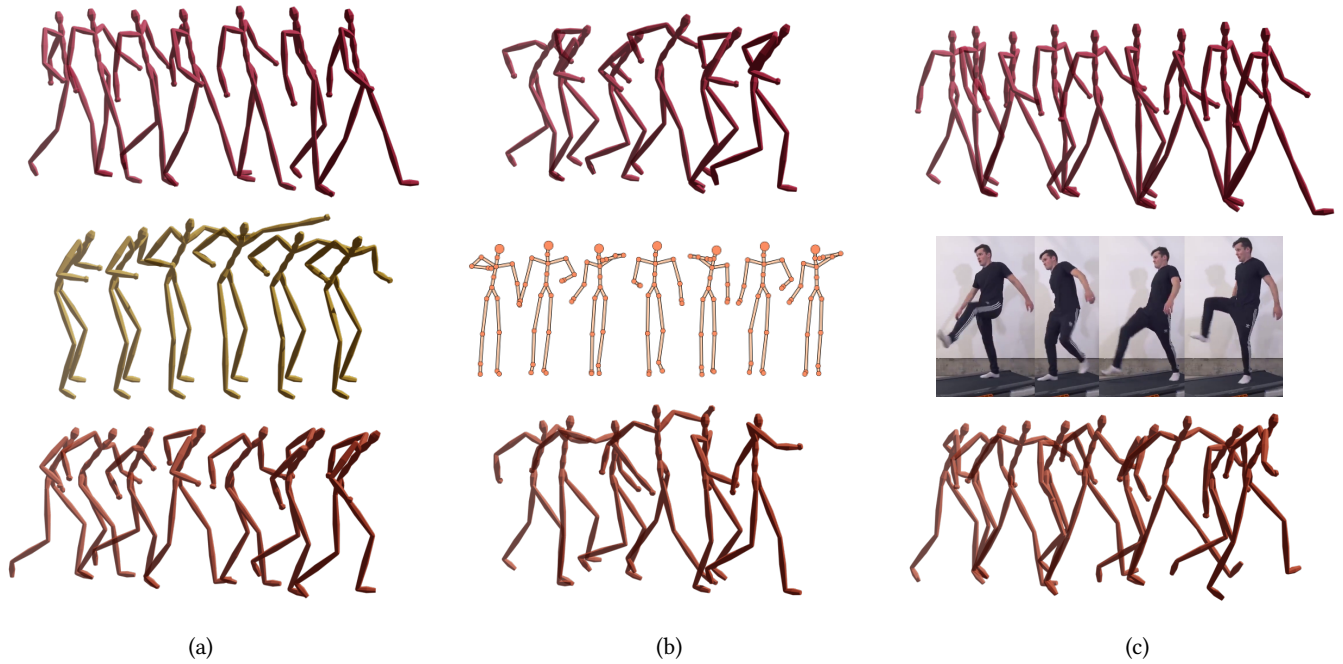
Fig. 4. Samples of our style transfer results. The motion of the content input (top row) is transferred to a motion with similar content and different style (bottom row), while the style can be extracted from various sources (middle row) such as 3D animated characters (a) 2D projection of 3D motions (b) and video sequences (c).
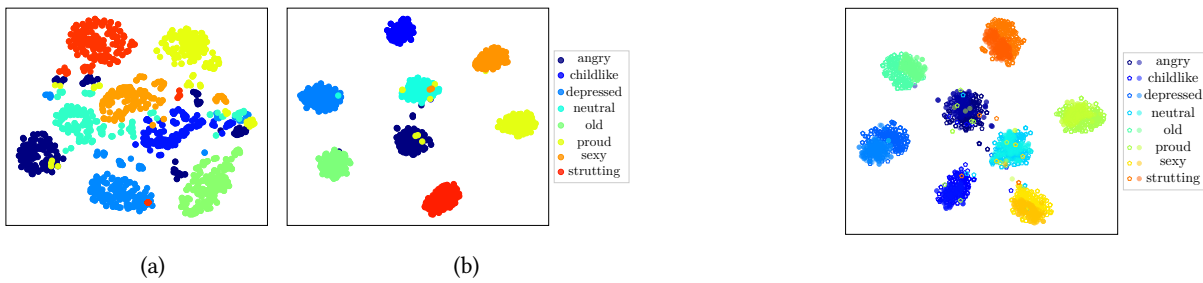

(a)   (b)

Fig. 5. The AdaIN parameters extracted from the style codes are projected onto 2D space using t-SNE and colored based on their style labels. The system is trained without triplet loss (a) and with triplet loss (b). It can be seen that our framework learns to cluster the AdaIN parameters as a function of style label in both cases, while the addition of the triplet loss results in tighter clusters.

However, when the number of styles is small, or the characteristic style properties are very different from those encountered during training, the network fails to generalize style.

The output motion clips of two experiments with unseen settings are shown in our supplemental video, next to an output that demonstrates how the same pair of inputs in each experiment is translated once the network has seen that style during training. As can be seen, although the outputs are different, in both cases the motion is plausible and the target style can be identified.

Fig. 6. Joint embedding of style codes parameters extracted from 3D motions as well as directly from 2D videos.
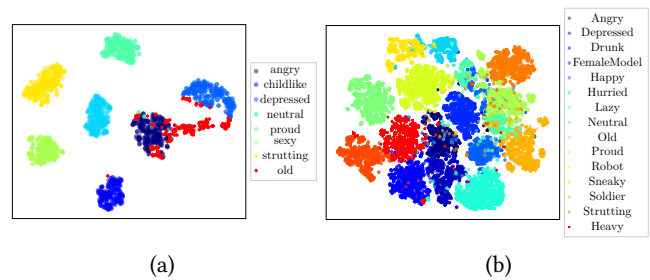

(a)   (b)

Fig. 7. Unseen styles. (a) Trained on dataset A excluding the "old" style. (b) Trained on dataset B excluding the "heavy" style. It can be seen that a larger number of style classes enables the network to better generalize styles in the latent space
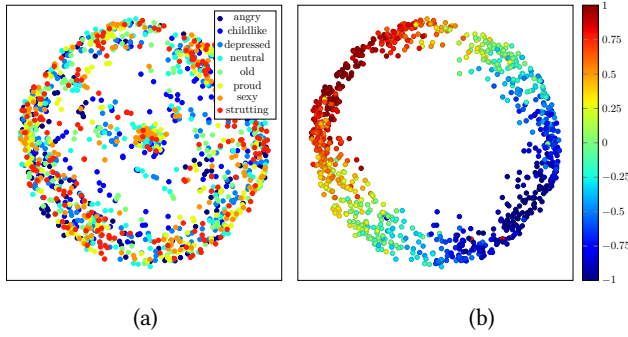
(a)   (b)

Fig. 8. Content codes samples are projected onto 2D space using PCA. (a) The samples are labeled by the style label. No clustering based on style label may be observed, suggesting that style information has been removed. (b) When visualizing only walking motions, while labeling samples by the phase of walking, it may be seen that our content code effectively parameterizes the motions using a single parameter – the phase.

Content Code. Figure 8(a) visualizes 2D projections of our content codes (dataset A), colored by their style labels. It can be seen that there is no clear correlation between the spatial positions of the points and their labels, which suggests that the content code probably does not contain significant style information.

Surprisingly, by observing the spatial distribution of the 2D points it can be seen that a subset of the samples forms a circle. The circle becomes nearly perfect by filtering out all the non-walking motions (using content motion labels that exist in the original dataset) and scaling the 2D space (the unscaled projection is elliptical). For walking motion samples, the reduced space achieved by PCA captures 97.4% of the original variation, which means that our projected content code preserves the information well.

The nearly perfect circle is achieved due to 3 main reasons: (i) Walking motions in dataset A exhibit approximately the same velocity. (ii) The network discards global velocity and orientation (iii) Our motion samples are represented by a fixed size temporal window. Thus, the content of these periodic motions can be parameterized with a single parameter: the phase. In order to confirm this interpretation we calculate the period of each walking motion sample, extract the phase Θ of the middle frame, and color the corresponding point with $\sin(\Theta)$ in Figure 8(b). The continuous variation of the color along the circle suggests that our network effectively strips the style and represents walking motions with a single phase parameter.

The phase representation for locomotion is well-known in character animation and is used, for example, to dictate the state or mode of a phase-based neural network that generates animations of humans [Holden et al. 2017b].

5.2 Comparison

In this section we compare our approach to the method of Holden et al. [2016] that performs style transfer by optimizing a motion sequence to satisfy two constraints, one for motion and one for style. Similarly to the seminal work of Gatys et al. [2016] for image style transfer, the content is described by a set of deep features, and the style is represented by the Gram matrix of those features. However,

MoCap	Holden et al. [2016]	Ours
79.17%	12.5%	75%

(a)

	Holden et al. [2016]	Ours
Content Preservation - 3D	38.89%	61.11%
Content Preservation - video	25%	75%
Style Transfer - 3D	5.56%	94.44%
Style Transfer - video	8.33%	91.67%

(b)

Table 1. User study results. (a) Realism ratios. (b) Content preservation and style transfer ratings ([Holden et al. 2016] vs. Ours). The style inputs were either from 3D motion, or from video.

while in the image domain the features are extracted by a classification network, here they are extracted by a motion autoencoder.

In order to perform the comparison the approaches are qualitatively evaluated by a user study that measures a few aspects of style transfer approaches. The results are evaluated with styles extracted from 3D motions, as well as from videos. However, since Holden et al. [2016] extract styles only from 3D motions, we use a state-of-the-art 3D pose estimation algorithm [Pavllo et al. 2019b] to recover 3D poses, when the provided style input is a video. For a fair comparison we use dataset A, which is part of the CMU dataset [CMU 2019], which Holden et al. [2016] used to train their model.

A few results extracted from the full comparison given in our supplementary video are depicted in Figure 9.

5.2.1 User Study. We performed a user study to perceptually evaluate the realism, style expressiveness and content preservation of our transfer results, while the style is extracted both from 3D motions and videos. 22 subjects were asked to answer a questionnaire with three types of questions, which we describe below.

Realism. In this part, we evaluate the realism of different motions. Users were presented with a pair of motions, both depicting the same type of content and style (e.g., angry jump). The motions were taken from three different sources: (1) Our original MoCap dataset, (2) Results of Holden et al. [2016] (3) Our results. Note that (2) and (3) are generated with similar inputs. Users were asked questions of the form: "Which of the above motions look more realistic?", and had to choose one of the four answers: Left, Right, Both, or None.

132 responses were collected for this question type. Table 1 reports the realism ratios for each motion source. It may be seen that 75% of our results were judged as realistic, which is a significantly higher ratio than the one measured for Holden et al. [2016], and not far below the realism ratio of real MoCap motions.

Content Preservation and Style Transfer. In this part, we compare our style transfer results to those of Holden et al. [2016] in terms of two aspects: the preservation of content and the transfer of style. Users were presented with a content input, a style input, and two transferred results, one by Holden et al. [2016] and the other by our method. They were asked to first select the motion whose content is closer to the content input ("Which of the motions on the right is more similar to the motion on the left in content?"), and then select

the motion whose style is closer to the style input ("Which of the motions on the right is more similar to the motion on the left in style?").

110 responses were collected for each of these two questions. The results are reported in Table 1. The results indicate that our method was judged far more successful in both aspects (content preservation and style transfer), both when using a 3D motion as the style input, and when using a video. The reasons to which we attribute the large gaps in the ratings are discussed in detail later in this section.

It can be seen that the user study states that our method yields results which are more faithful to the task of style transfer. In particular, it can be seen that the approach of Holden et al. [2016] struggles to transfer style when the content of the two input motions is different (for example, when the input content motion is "proud walking" and the input style is "depressed kicking"). The main reason is that both content and style representations are derived from the same deep features, which leads to a dependency of content and style. In order to get a better understanding of their style representation, we projected the styles extracted by both methods into 2D, using t-SNE. Figure 10 shows the resulting maps. It can be seen that while our samples are clustered by the style labels (right plot), this cannot be observed for Holden's representation, which results in a multitude of small clusters, scattered over the 2D plane. Thus, while Gram matrices of features extracted by an autoencoder enable some degree of style transfer, they are clearly affected by other information present in the motion samples.

Moreover, we use video style examples, where a person demonstrates different walking styles, while walking on a treadmill. When poses are extracted from such a video, the root velocity is very small. In contrast, most of the content inputs have significant root velocity. This discrepancy poses no problem for our approach, but it adversely affects the method of Holden et al.[2016], which is limited to work with input pairs that share the same content.

Our method explicitly attempts to extract a latent style code from an input style motion, which enables clustering of motions with different content but similar style in the latent style space, thereby disentangling style from content. In contrast, Holden et al. [2016] represent style using Gram matrices, similarly to the seminal work of Gatys et al. [2016] for image domain style transfer.

5.3 Ablation Study and Insights

Effect of Adversarial Loss. In this experiment we discarded the adversarial loss $\mathcal{L}_{adv}$ from our training. Surprisingly, our experiments show that the discriminator does not play the key role in the transferring of style. Furthermore, a single content consistency loss is sufficient to train the network to extract shared property with from labeled styles, and to cluster the style code samples by their style labels. However, we found that without the attendance of the adversarial loss, the perceived realism of the output motions is degraded, and artifacts such as some shaking can be observed. The comparison can be found in the supplementary video.

Style Code and Neutral Style. In order to gain a better understating of the impact of the style code and the structure of its space, we neutralized the style branch by setting the AdaIN output to

identity parameters (zero mean, unit variance). With these settings, the network outputs pure noise. The reason is that the network is trained in an end-to-end fashion, the scale and translation are also responsible for modifying the features such that the network outputs valid motions. In addition, in order to understand whether the neutral style is more centralized in the style latent space than other styles, for every style label, we calculated the mean distance between its average style code to all the other average style codes. We found that in both of the datasets the neutral style is among the top three styles in terms of that mean distance, which might suggest that the network learns to branch from neutral style into the other styles. However, we are not able to reach a definitive conclusion based on this experiment.

5.4 Style Interpolation

Our learned continuous style code space can be used to interpolate between styles. Style interpolation can be achieved by linearly interpolating between style code, and then decoding the results through our decoder. Our video demonstrates motions where the content input is fixed (neutral walking) and the style is interpolated between two different style codes (depressed to proud and neutral to old). Figure 11 shows a key-frame from each interpolated motion sequence.

6 CONCLUSIONS AND FUTURE WORK

We have presented a neural network that transfers motion style from one sequence into another.

The key novelty is that the network is trained without paired data, and without attempting to explicitly define either motion content or motion style. Nevertheless, the results show that the network succeeds to implicitly disentangle style and content and combine even previously unseen styles with a given content. We partly attribute the success of the approach to the asymmetric structure of the network, where the style is represented and controlled by instance normalization layers, while the content by deep convolutional layers. Instance normalization layers have the innate tendency to control mainly local statistics, or in other words, details, while the convolutional layers preserve the content. Our training protocol, with various motion styles, encourages this asymmetric network to disentangle the latent style from the motion's content.

Although there is no universally accepted definition of motion style, it may be argued that our framework defines style as the set of properties, shared by motions in a group, which can be manipulated (added/removed) by an affine, temporally invariant, transformation (AdaIN) applied to deep features. As a consequence, the complementary part of style that enables the reconstruction of local motion (without the root position), is defined as the content. In addition, the global positions, which are taken directly from the content input, are considered as part of the content, while the global velocity, which is temporally warped based on the style input, is defined to be part of the style in our case.

Our mechanism aims at disentangling style and content of arbitrary motions based on style labels. However, if the two input motions (content input and style input) during test time are different (lacking commonalities) and the target style departs too much from

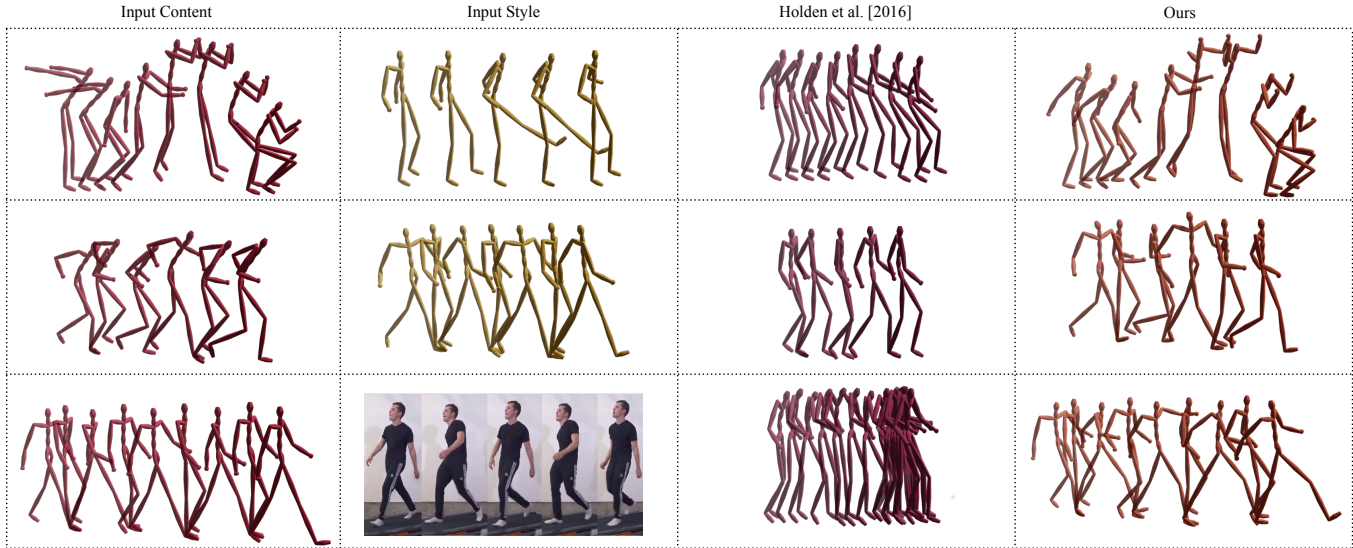| Input Content | Input Style | Holden et al. [2016] | Ours |

Fig. 9. Qualitative comparison of our method to the approach of Holden et al. [2016]. The content input is shared across all the examples (each column shows a different example), the input style is depicted in the first row, while the results of Holden et al. [2016] and ours are given in the second and last row, respectively. We picked a fixed set of key frames of each motion to demonstrate the results. The full video sequences and more results can be found in the supplemental video.
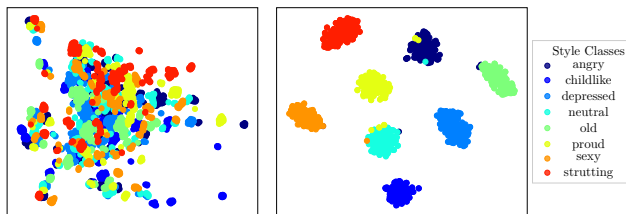

Fig. 10. Style codes extracted by our method (right) compared to the style representation of Holden et al. [2016] (left). While our style codes are clustered by style labels, while in Holden's representation the style representation for many small scattered clusters, which implies dependencies between style and content. While our style codes are clustered by style labels, the style representation of Holden et al. [2016] forms many small scattered clusters, which imply dependencies between style and content.
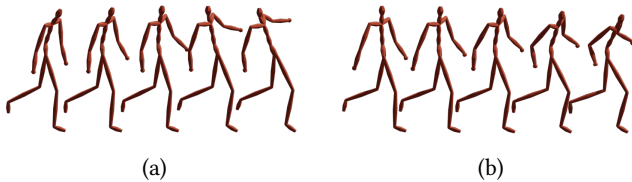

(a)   (b)

Fig. 11. Style interpolation. Our style space code enables motion style interpolation. A neutral walking is transferred to an interpolated style. (a) Depressed to proud. (b) neutral to old.

the ones used for training, the network will not be able to infer which style properties should be transferred. Moreover, due to the fact that the majority of the motions in our datasets depict locomotion (walking and running) the network tends to output motions of higher-quality with such samples during test time. In turn, this motivates us to use our generative system to produce more data and new styles by possibly mixing styles or amplifying (or attenuating) available styles or mixes of styles.

Another notable limitation is that testing the system with characters that have different body proportions from those which were seen during training, may lead to implausible results. In order to cope with such cases, motion retargeting should be performed prior to the style transfer pass. Motion retargeting is a challenging problem in its own right, and is outside the scope of this work. In order to support style transfer of various unseen skeletons in an end-to-end fashion, a different solution would have to be proposed. We leave this issue to future work.

In our current implementation, a given pair of input content and style motions yields a deterministic output. We would like to consider extending the system by injecting noise to produce slight variations. This will allow a temporal prolongation of the input sequence without noticeable repetitions or discontinuous transitions. In the future, we would also consider segmenting the sequence temporally and transferring different styles to different segments.

We believe that the role of instance normalization in motion processing and animation is likely to increase, especially for generative models. The work we presented, is only a first step in that direction.

ACKNOWLEDGMENTS

REFERENCES

Kfir Aberman, Mingyi Shi, Jing Liao, Dani Lischinski, Baoquan Chen, and Daniel Cohen-Or. 2019a. Deep Video-Based Performance Cloning. In *Computer Graphics Forum*, Vol. 38. Wiley Online Library, 219–233.

Kfir Aberman, Rundi Wu, Dani Lischinski, Baoquan Chen, and Daniel Cohen-Or. 2019b. Learning Character-Agnostic Motion for Motion Retargeting in 2D. *ACM Transactions on Graphics (TOG)* 38, 4 (2019), 75.

Kenji Amaya, Armin Bruderlin, and Tom Calvert. 1996. Emotion from motion. In *Graphics Interface*, Vol. 96. Toronto, Canada, 222–229.

Andreas Aristidou, Daniel Cohen-Or, Jessica K. Hodgins, Yiorgos Chrysanthou, and Ariel Shamir. 2018. Deep Motifs and Motion Signatures. *ACM Trans. Graph.* 37, 6, Article 187 (Nov. 2018), 13 pages. https://doi.org/10.1145/3272127.3275038

Andreas Aristidou, Qiong Zeng, Efstathios Stavrakis, KangKang Yin, Daniel Cohen-Or, Yiorgos Chrysanthou, and Baoquan Chen. 2017. Emotion control of unstructured dance movements. In *Proc. ACM SIGGRAPH/Eurographics Symposium on Computer Animation*. ACM, 9.

Matthew Brand and Aaron Hertzmann. 2000. Style machines. In *Proc. SIGGRAPH 2000*. ACM Press/Addison-Wesley Publishing Co., 183–192.

Zhe Cao, Gines Hidalgo, Tomas Simon, Shih-En Wei, and Yaser Sheikh. 2018. OpenPose: realtime multi-person 2D pose estimation using Part Affinity Fields. *arXiv preprint arXiv:1812.08008* (2018).

Caroline Chan, Shiry Ginosar, Tinghui Zhou, and Alexei A Efros. 2019. Everybody dance now. In *Proceedings of the IEEE International Conference on Computer Vision*. 5933–5942.

CMU. 2019. CMU Graphics Lab Motion Capture Database. http://mocap.cs.cmu.edu/

Han Du, Erik Herrmann, Janis Sprenger, Noshaba Cheema, Klaus Fischer, Philipp Slusallek, et al. 2019. Stylistic Locomotion Modeling with Conditional Variational Autoencoder. In *Proc. Eurographics*. The Eurographics Association.

Leon A Gatys, Alexander S Ecker, and Matthias Bethge. 2016. Image style transfer using convolutional neural networks. In *Proc. CVPR*. 2414–2423.

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

Daniel Holden, Ikhsanul Habibie, Ikuo Kusajima, and Taku Komura. 2017a. Fast neural style transfer for motion data. *IEEE computer graphics and applications* 37, 4 (2017), 42–49.

Daniel Holden, Taku Komura, and Jun Saito. 2017b. Phase-functioned neural networks for character control. *ACM Transactions on Graphics (TOG)* 36, 4 (2017), 42.

Daniel Holden, Jun Saito, and Taku Komura. 2016. A deep learning framework for character motion synthesis and editing. *ACM Transactions on Graphics (TOG)* 35, 4 (2016), 138.

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

Eugene Hsu, Kari Pulli, and Jovan Popović. 2005. Style translation for human motion. In *ACM Transactions on Graphics (TOG)*, Vol. 24. ACM, 1082–1089.

Xun Huang and Serge Belongie. 2017. Arbitrary style transfer in real-time with adaptive instance normalization. In *Proc. ICCV*. 1501–1510.

Xun Huang, Ming-Yu Liu, Serge Belongie, and Jan Kautz. 2018. Multimodal unsupervised image-to-image translation. In *Proc. ECCV*. 172–189.

Leslie Ikemoto, Okan Arikan, and David Forsyth. 2009. Generalizing motion edits with gaussian processes. *ACM Transactions on Graphics (TOG)* 28, 1 (2009), 1.

Justin Johnson, Alexandre Alahi, and Li Fei-Fei. 2016. Perceptual losses for real-time style transfer and super-resolution. In *Proc. ECCV*. Springer, 694–711.

Angjoo Kanazawa, Jason Y Zhang, Panna Felsen, and Jitendra Malik. 2019. Learning 3d human dynamics from video. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 5614–5623.

Tero Karras, Samuli Laine, and Timo Aila. 2019. A style-based generator architecture for generative adversarial networks. In *Proc. CVPR*. 4401–4410.

C Karen Liu, Aaron Hertzmann, and Zoran Popović. 2005. Learning physics-based motion style with nonlinear inverse optimization. In *ACM Transactions on Graphics (TOG)*, Vol. 24. ACM, 1071–1081.

Lingjie Liu, Weipeng Xu, Michael Zollhoefer, Hyeongwoo Kim, Florian Bernard, Marc Habermann, Wenping Wang, and Christian Theobalt. 2018. Neural Rendering and Reenactment of Human Actor Videos. *arXiv preprint arXiv:1809.03658* (2018).

Ming-Yu Liu, Xun Huang, Arun Mallya, Tero Karras, Timo Aila, Jaakko Lehtinen, and Jan Kautz. 2019. Few-shot unsupervised image-to-image translation. *arXiv preprint arXiv:1905.01723* (2019).

Wanli Ma, Shihong Xia, Jessica K Hodgins, Xiao Yang, Chunpeng Li, and Zhaoqi Wang. 2010. Modeling style and variation in human motion. In *Proc. 2010 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*. Eurographics Association, 21–30.

Ian Mason, Sebastian Starke, He Zhang, Hakan Bilen, and Taku Komura. 2018. Few-shot Learning of Homogeneous Human Locomotion Styles. In *Computer Graphics Forum*, Vol. 37. Wiley Online Library, 143–153.

Dushyant Mehta, Srinath Sridhar, Oleksandr Sotnychenko, Helge Rhodin, Mohammad Shafiei, Hans-Peter Seidel, Weipeng Xu, Dan Casas, and Christian Theobalt. 2017. VNect: Real-time 3D Human Pose Estimation with a Single RGB Camera. *ACM Trans. Graph.* 36, 4, Article 44 (July 2017), 44:1–44:14 pages. https://doi.org/10.1145/3072959.3073596

Taesung Park, Ming-Yu Liu, Ting-Chun Wang, and Jun-Yan Zhu. 2019. Semantic image synthesis with spatially-adaptive normalization. In *Proc. CVPR*. 2337–2346.

Dario Pavllo, Christoph Feichtenhofer, Michael Auli, and David Grangier. 2019a. Modeling Human Motion with Quaternion-based Neural Networks. *arXiv preprint arXiv:1901.07677* (2019).

Dario Pavllo, Christoph Feichtenhofer, David Grangier, and Michael Auli. 2019b. 3D human pose estimation in video with temporal convolutions and semi-supervised training. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 7753–7762.

Ari Shapiro, Yong Cao, and Petros Faloutsos. 2006. Style components. In *Proc. Graphics Interface 2006*. Canadian Information Processing Society, 33–39.

Harrison Jesse Smith, Chen Cao, Michael Neff, and Yingying Wang. 2019. Efficient Neural Networks for Real-time Motion Style Transfer. *PACMCGIT* 2, 2 (2019), 13:1–13:17.

Graham W Taylor and Geoffrey E Hinton. 2009. Factored conditional restricted Boltzmann machines for modeling motion style. In *Proc. ICML*. ACM, 1025–1032.

Dmitry Ulyanov, Andrea Vedaldi, and Victor Lempitsky. 2016. Instance normalization: The missing ingredient for fast stylization. *arXiv preprint arXiv:1607.08022* (2016).

Munetoshi Unuma, Ken Anjyo, and Ryozo Takeuchi. 1995. Fourier principles for emotion-based human figure animation. In *Proc. SIGGRAPH '95*. ACM, 91–96.

Ruben Villegas, Jimei Yang, Duygu Ceylan, and Honglak Lee. 2018. Neural kinematic networks for unsupervised motion retargetting. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 8639–8648.

Jack M Wang, David J Fleet, and Aaron Hertzmann. 2007. Multifactor Gaussian process models for style-content separation. In *Proc. ICML*. ACM, 975–982.

Ting-Chun Wang, Ming-Yu Liu, Jun-Yan Zhu, Andrew Tao, Jan Kautz, and Bryan Catanzaro. 2018. High-resolution image synthesis and semantic manipulation with conditional gans. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 8798–8807.

Shihong Xia, Congyi Wang, Jinxiang Chai, and Jessica Hodgins. 2015. Realtime style transfer for unlabeled heterogeneous human motion. *ACM Transactions on Graphics (TOG)* 34, 4 (2015), 119.

M Ersin Yumer and Niloy J Mitra. 2016. Spectral style transfer for human motion between independent actions. *ACM Transactions on Graphics (TOG)* 35, 4 (2016), 137.

Yi Zhou, Connelly Barnes, Jingwan Lu, Jimei Yang, and Hao Li. 2019. On the continuity of rotation representations in neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 5745–5753.