Efficient Neural Networks for Real-time Motion Style Transfer

HARRISON JESSE SMITH, University of California, Davis, USA
CHEN CAO, Snap Inc., USA
MICHAEL NEFF, University of California, Davis, USA
YINGYING WANG, Snap Inc., USA

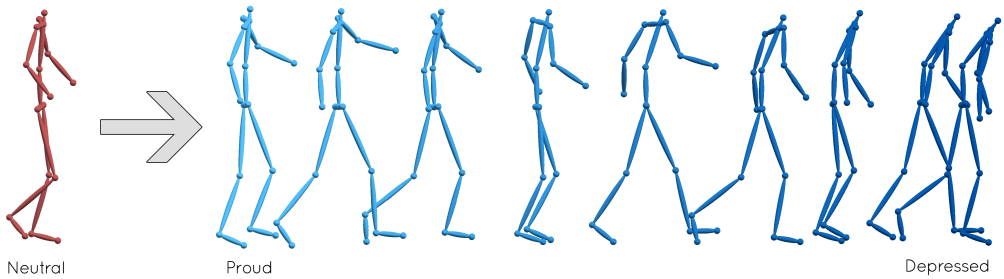

Fig. 1. Our motion style transfer method can stylize human motion in real-time. A single set of efficient networks can apply multiple different styles and interpolate between, as shown in the proud to depressed walking motion above.

Style is an intrinsic, inescapable part of human motion. It complements the content of motion to convey meaning, mood, and personality. Existing state-of-the-art motion style methods require large quantities of example data and intensive computational resources at runtime. To ensure output quality, such style transfer applications are often run on desktop machine with GPUs and significant memory. In this paper, we present a fast and expressive neural network-based motion style transfer method that generates stylized motion with quality comparable to the state of the art method, but uses much less computational power and a much smaller memory footprint. Our method also allows the output to be adjusted in a latent style space, something not offered in previous approaches. Our style transfer model is implemented using three multi-layered networks: a pose network, a timing network and a foot-contact network. A one-hot style vector serves as an input control knob and determines the stylistic output of these networks. During training, the networks are trained with a large motion capture database containing heterogeneous actions and various styles. Joint information vectors together with one-hot style vectors are extracted from motion data and fed to the networks. Once the network has been trained, the database is no longer needed on the device, thus removing the large memory requirement of previous motion style methods. At runtime, our model takes novel input and allows real-valued numbers to be specified in the style vector, which can be used for interpolation, extrapolation or mixing of

Authors' addresses: Harrison Jesse Smith, University of California, Davis, One Shields Ave. Davis, CA, 95616, USA, hjsmith@ucdavis.edu; Chen Cao, Snap Inc. Santa Monica, CA, 90405, USA, chen.cao@snap.com; Michael Neff, University of California, Davis, One Shields Ave, Davis, CA, USA, mpneff@ucdavis.edu; Yingying Wang, Snap Inc. Santa Monica, CA, 90405, USA, ywang@snap.com.

2577-6193/2019/7-ART13 $15.00
https://doi.org/10.1145/3340254

styles. With much lower memory and computational requirements, our networks are efficient and fast enough for real-time use on mobile devices. Requiring no information about future states, the style transfer can be performed in an online fashion. We validate our result both quantitatively and perceptually, confirming its effectiveness and improvement over previous approaches.

CCS Concepts: • **Computing methodologies** → **Animation**; *Motion processing*; *Motion capture*; *Machine learning*.

Additional Key Words and Phrases: deep learning, character animation, motion editing, style transfer

ACM Reference Format:
Harrison Jesse Smith, Chen Cao, Michael Neff, and Yingying Wang. 2019. Efficient Neural Networks for Real-time Motion Style Transfer. *Proc. ACM Comput. Graph. Interact. Tech.* 2, 2, Article 13 (July 2019), 17 pages. https://doi.org/10.1145/3340254

1 Introduction

Motion style- the manner in which an action is performed- is an important aspect of human movement that conveys personality, mood and identity. By conveying information about a person's internal state, a motion's style often provides more important information than the content itself. Motion capture is an effective mechanism for acquiring realistic, stylized motions. However, the vast number of combinations of motion content and styles means that the amount of data acquisition work required to span a reasonable style space is enormous. Motion style transfer is one effective solution that can greatly reduce the work load. It allows users to reuse captured motion by modifying it with new target styles while preserving its original content.

With the ubiquity of mobile digital devices such as phones, tablets, augmented reality glasses and headsets, there are ample opportunities to incorporate motion style transfer into mixed reality and game applications. However, motion style transfer algorithms must be improved if they are to run in real-time on such devices. These devices require algorithms that are not computationally intensive and have small memory footprints: these constraints present a major challenge for the motion style transfer task, and preclude the use of existing algorithms. Many previous style transfer techniques either rely on representative style features extracted from a large motion database [Aristidou et al. 2017; Yumer and Mitra 2016] or learn a transfer function following paired examples in the database [Hsu et al. 2005; Xia et al. 2015]. For these works, an all-inclusive database that covers all possible content and styles needs to be present on the device, which makes mobile application infeasible. Additionally, as motion data exists in a high dimensional space, searching motion examples [Xia et al. 2015], temporal alignment [Hsu et al. 2005], optimization [Brand and Hertzmann 2000; Holden et al. 2016] and fitting transfer function [Hsu et al. 2005; Xia et al. 2015] are all computationally expensive processes. Some of these methods employ powerful GPUs to achieve real-time performance, but this is not practical for use on mobile devices. Here, we deliberately overcome the aforementioned challenges, and thus significantly enhances the practical use of motion style transfer by supporting mobile applications.

Our fast and compact motion style transfer algorithm for mobile devices uses a network architecture that decouples logically distinct elements of the style transfer task. We separately train a *Pose Network* to predict spatial style variation, a *Timing Network* to predict temporal style variation, and a *Foot Contact Network* to assist in the removal of footskate artifacts. Our method is fast, as we handle the transfer in an online fashion using feed forward layers as building blocks. It is also compact, as only the trained parameters of the networks are needed during runtime. To specify the networks' transfer behavior, we use a one-hot style vector as the control knob, which allows the networks to perform valid interpolation, extrapolation, blending, and mixture in a latent style space. It also prevents us from needing to train and store multiple sets of network parameters, as

in [Holden et al. 2017b; Mason et al. 2018; Zhang et al. 2018], to achieve complicated transfers. For motion sequences containing heterogeneous, unlabeled actions, our transfer quality is on par with the previous state-of-the-art method [Xia et al. 2015], which requires 290MB and runs at 55 FPS with GPU acceleration. However, our approach is significantly faster with a lower memory requirement: 931KB; 115 FPS on CPU).

At runtime, any novel motion, even defined in a kinematic skeleton different from the training data, can be directly streamed into the networks as input for the transfer, as long as the relevant joint positions can be computed and the general scale matches. No retargetting or input action labels are required. As the expensive processes of temporal motion alignment and spatial transformation learning are handled through offline training, the runtime style transfer is straightforward and fast. Also during runtime, real-valued data is allowed in the style vector to manipulate the influence of the networks' operations. Adjusting style output in this manner minimizes the memory requirements of our approach, as no additional network parameters are necessary to achieve complex transfer behavior. We also argue that using a real-valued style vector to interpolate, extrapolate, and mix styles in a latent space can generate higher quality results that the direct blending of motions of different styles.

We summarize the main contribution of our work as follows:

- We propose a fast and compact style transfer network-based method suitable for mobile device, with quality comparable to state of the art work;
- We decouple the complex motion style transfer function into three controllable network modules: a pose network, timing network and foot contact network. They collaboratively achieves the motion style transfer;
- We develop a one-hot style vector as a user control, which allows generation of effective style interpolation, extrapolation, and mixtures.

2 Related Work

Realistic human motion is often acquired through motion capture, which involves a laborious data processing pipeline. Many previous research efforts have focused on how to edit and synthesize novel motion by efficiently reusing an existing motion capture databases.

Stylistic Motion Analysis and Generation

Substantial research has sought to understand the movement factors that lead to variations in style perception, including personality, emotion, distinctiveness, and attractiveness. Studies in [McDonnell et al. 2008] confirmed that motion provides reliable emotional cues that are unaffected by a character's appearance. Perceptual experiments in [Neff et al. 2010; Smith and Neff 2017; Wang et al. 2016] indicate that low-level motion features such as direction, amplitude, speed, frequency, and general posture significantly impact the perceived personality of the performer. Thus, procedurally modifying these key motion features can effectively generate different styles of motions. In [Aristidou et al. 2017], the authors compute emotion coordinates from dance movement based on Laban Movement Analysis features, and stylize output dances with desired emotions by editing the associated motion features. [Hoyet et al. 2013] indicates that individual motion characteristics can be transferred between similar actions (e.g. walking vs. jogging) but that, in general, style transferability is limited between different types of motion. However, these works rely heavily on manually designed experiments, which do not fully utilize the information from the motion data.

Early research regarded motion style as "secondary" movement that can be laid on top of the "primary" motion content [Amaya et al. 1996]. Identifying style transform models, which are the

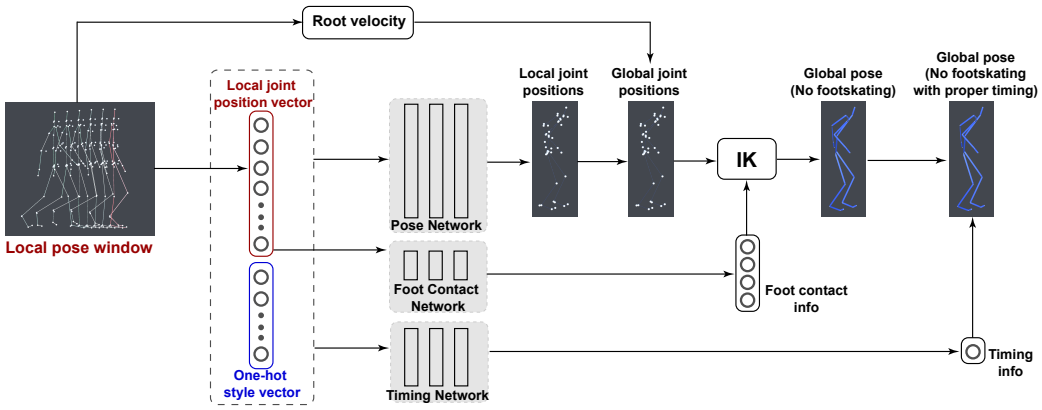

Fig. 2. Workflow Overview. We have three networks in our pipeline: a pose network, a foot contact network and a timing network. The input of the foot contact network is a local joint position vector, while the inputs to the pose network and timing network are the same position vector concatenated with a one-hot style vector. The pose network predicts local joint positions which, when combined with the input motion sequence's root velocity, can be used to recover global joint positions. The foot contact network outputs the probabilities of the four foot contact states (both in contact, neither in contact, left in contact only, right in contact only). The timing network outputs a single value representing the timing difference between the current and previous pose in the output style.

differences in time and space between example motion pairs, and applying them to new neutral motions is one solution. More sophisticated transform models are developed in [Hsu et al. 2005; Xia et al. 2015]. Hsu et al.[2005] use Iterative Motion Warping (IMW) to find temporal correspondences between motion pairs and applied a Linear Time-Invariant (LTI) model to transform an input motion into a new style. To better handle long sequences of unlabeled, heterogeneous motion content, Xia et al. [2015] adopted local mixtures of autoregressive models, which are trained on the fly by the nearest motion examples in the database. Instead of modeling the style transform, Brand and Hertzmann [2000] learned a set of stylistic parameters as control knobs for their stylistic Hidden Markov Models (SHMM), where new motion styles can be synthesized by tuning these parameters. Kim and Neff [2012] took a component-based approach to style, allowing users to mix components from different style motions in order to generate novel output. Nunes et al. [2012] used modal vibrations to create stylized physics-based controllers. . Other work interprets motion style in spectral domain [Unuma et al. 1995; Yumer and Mitra 2016]. In recent work [Yumer and Mitra 2016], styles are identified as spectral intensity and can be transferred to heterogeneous motion content that is unseen in the database.

Deep Learning for Human Motion

With the rapid development of deep learning, researchers have started to show promising results when applying deep learning to synthesizing human motion. [Taylor and Hinton 2009] built a factored Conditional Restricted Boltzmann Machine (CRBM) to model stylistic human motions. The three-way interactions gated by stylistic features allow natural interpolation, extrapolation and smooth transition of styles in the training data. [Holden et al. 2015; Wang and Neff 2015] extracted high-level motion features through deep autoencoders, where features can be further applied to motion retrieval, interpolation and denoising. [Du et al. 2019] train a conditional variational autoencoder to learn a style transfer distribution and generate novel versions of stylized locomotion

given a few target style examples and a neutral database. Holden et al. [2016] proposed a deep learning framework by stacking task-specific networks on top of the motion autoencoder. This framework is capable of editing and controlling locomotion, as well as some simple stylization tasks using a Gram matrix. However, in their work, the temporal variation is determined by motion content not style, and no style mixture or extrapolation is considered. Their follow-up work [Holden et al. 2017a] speeds up style transfer by using a feed forward transformation network. Artifacts in the output are removed by re-projecting motion to latent manifold through a second convolutional autoencoder. Phase-Functioned Neural Networks (PFNN) [Holden et al. 2017b] and Mode-Adaptive Neural Networks [Zhang et al. 2018] are successful examples of using deep neural networks for controlling biped or quadruped locomotion, where locomotion with different foot patterns on varied terrains can be generated by blending sets of parameters of a trained network. Built upon PFNN, [Mason et al. 2018] learns style-agnostic general locomotion through PFNN, and further adds residual adapters to capture stylistic features from limited amount of exemplar data; however, this approach does not extend to heterogeneous sequences containing non-cyclic actions.

Recurrent neural networks (RNN) and long short-term memory models (LSTM) represent human motion as time series. For the RNN proposed in [Pavllo et al. 2018], human motion is represented in quaternion parameterization and penalized by positional loss through forward kinematics. Their two-layered gated recurrent unit networks handle both short term motion prediction and long term locomotion generation. By using an encoder RNN and a decoder RNN with a forward kinematics layer, [Villegas et al. 2018] is capable of online motion retargetting between characters with different skeletal bone lengths. [Li et al. 2017] overcame the issue of error accumulation in LSTMs and proposed a novel auto-conditioned LSTM structure, which can generate continuous and complex human motion with different styles. Deep reinforcement learning has also been applied to learning physics-based locomotion [Peng et al. 2017], balancing skills [Liu and Hodgins 2017], basketball dribbling [Liu and Hodgins 2018], and acrobat stunts [Peng et al. 2018]. In our work, we propose a novel deep neural network architecture specifically for the motion stylization tasks, with flexible style transfer and better generalization capabilities.

3 Overview

As we intend our algorithm to be used in real-time on mobile devices, it must be fast, have a small memory footprint, require no future information, and work on heterogeneous action sequences containing non-cyclic motion. We therefore propose a series of networks that take, as input, a heterogeneous motion sequence and output that same sequence in a specific style. The general overview of our workflow is shown in Fig. 2.

Data Preparation

To train such a network architecture, we prepared a large human motion database including heterogeneous action sequences perform in different styles. In this work we make use of motion data gathered by [Xia et al. 2015], but our approach is not restricted to their database. Each pose (skeletal configuration of the performer during a single frame of motion) in the database is annotated with foot contact information, registered with similar poses in different styles, and further augmented for training the networks as described in Section 4.2.

While it is computationally intensive to create this large database, it is only necessary for network training and can afterwards be discarded. Delegating these expensive computations to a preprocessing step allows our algorithm to be fast at runtime.

Networks

Style changes the positions of a performer's joints and the speed at which a performer moves; therefore, we train a *Pose Network* and a *Timing Network*. To assist in removing footskate, we also train a *Foot Contact Network*. A window of poses is fed to the Foot Contact Network. This pose window is concatenated with a one-hot style vector and fed into the Pose Network and Timing Network. Conditioning outputs on a one-hot style vector reduces the memory footprint of our approach, as one set of parameters can be used to predict multiple styles. The three networks output pose, foot contact and timing information respectively, which can be further integrated to construct a stylized version of the input motion sequence.

Runtime Style Transfer

During runtime, the networks can take novel pose input and create stylized output. Different from the training phase, in which the style vector weights are restricted to values of zero or one, real valued numbers can be used throughout the style vector during runtime. Doing so allows the networks to output rich motion styles that can be interpolated, extrapolated and mixed.

4 Methods

In this section, we describe our approach in detail. Sec. 4.1 describes how the training data is prepared. Sec. 4.2, describes the structure of the network inputs and outputs. Sec. 4.3 explains how the networks are designed and trained. Sec. 4.4 describes how the different network outputs are integrated together to construct the final, styled motion sequence. In Sec. 4.5 we introduce the style interpolation, extrapolation and mixing capabilities of our method.

4.1 Data Preparation

The database we use to train this network architecture was derived from previous motion style transfer work [Xia et al. 2015]. The motion database contained examples of walking, running, jumping, kicking, punching, and transitions between these actions in eight different styles (*neutral*, *proud*, *angry*, *depressed*, *strutting*, *childlike*, *old* and *sexy*). There are 11 minute of motion in the database. It was captured with a Vicon optical system at 120 Hz, resulting in roughly 79,000 individual frames. All the motions were retargeted to a single skeleton with 25 joints. Eighteen virtual joints were added to the skeleton, which are useful for later deriving joint rotations from predicted joint positions.

Similar actions in the database are registered with each other across styles [Kovar and Gleicher 2003]. Temporal registration helps build the correspondence between poses of different styles in the motion sequences, which will be further organized as the training sample pairs used by our networks to learn supervised style transfer functions.

We further augmented this data by feeding it into the system of [Xia et al. 2015], producing seven different synthetically stylized versions of the motion (*neutral* versions of the motion were not produced). The end result of the process was roughly 550,000 tuples of input pose, styled output pose, styled output timing, and foot contact state.

Our method is not restricted to the specific representation used in the database. It only requires the database to contain a sufficient quantity of various styled motion sequences with corresponding poses for training.

4.2 Input and Output Formats

Next, we process the tuples created in the previous step to derive the training data used by our model.

To be invariant to global translation and orientation, we represent the input poses in their local root space, following the procedure in [Holden et al. 2017b]. Each pose is translated such that its root joint is located above the origin; the height of the root joint is not adjusted. The pose is then rotated such that its forward vector (taken as the average of the vectors perpendicular to the vector connecting the hip joints and the vector connecting the shoulder joints) faces along the positive z axis.

We take as input a **local pose window** $\boldsymbol{w}_i = \{\boldsymbol{t}_i, \boldsymbol{t}_{i-s}, ..., \boldsymbol{t}_{i-n*s}\}$, where $\boldsymbol{t}_i$ is the local joint position vector of the pose at time i, translated and rotated as described above. Because motion is a time series and the velocities and accelerations of joints contain important information about the state of the motion, we include $\{\boldsymbol{t}_{i-s}...\boldsymbol{t}_{i-n*s}\}$, subsampled historical joint positions represented in the local space of $\boldsymbol{t}_i$ (see left of Figure 2 for a visual example). Here n is the number of previous poses to include and s is the step size between poses. In our implementation, $n = 5$ and $s = 6$, which covers the previous 0.25 seconds of motion. $\boldsymbol{t} \in \mathbb{R}^{3j}$, where j is 43, the number of real and virtual joints comprising the skeleton. Thus $\boldsymbol{w} \in \mathbb{R}^{3\times43\times6}$ contains the xyz positions of the 43 joints of the local pose and the five historical poses.

We define the input **style vector** at frame i as $\boldsymbol{s}_i \in \mathbb{R}^{|L|}$, where L is the set of all output styles present in the training samples. In our implementation $|L| = 7$. During the training phase, $\boldsymbol{s}_{i,l}$ is set to 1.0, where $l \in L$ is the output style of the predicted pose. All other dimensions of $\boldsymbol{s}$ are set to 0.0. At run-time, $\boldsymbol{s}$ takes real-valued input and serves as a control handle. This allows the user to specify the desired style or style mixture for the networks' output.

The **output pose** is defined as $\boldsymbol{p} = \{\boldsymbol{t}'_i\}$. Similar to poses inside the local pose window, $\boldsymbol{t}'_i \in \mathbb{R}^{3j}$,where j is 43. We represent the output pose in its local root space by transforming it, such that the values along the x and z axis of its root joint are zero and the pose faces towards the positive z axis.

An alternative approach we initially used involved predicting the output pose in the local space of the input pose's root joint. However, we found this representation problematic: when the stylized version of a long cyclic motion sequence (such as walking) is faster or slower than the input version, the global positions of corresponding poses within the sequences may diverge over time. For example, a depressing walk is often slower than a neutral walk; over the course of the motion, the depressed pose will fall further and further behind the neutral walk. This can cause problems for the pose prediction model: a pose window taken from the beginning of the sequence looks very similar to a pose window taken from the end of the sequence, but the desired joint position predictions would be very different. Therefore we use a local space representation, in which the output pose's root joint is always located above the origin.

The output **foot contact** information for frame i is defined as $\boldsymbol{c}_i \in \mathbb{R}^4$. This output vector predicts the probability of the pose's foot contact states: left foot in contact only, right foot in contact only, both feet in contact, or neither foot in contact. The state with the greatest probability is recorded and later used to remove foot skate.

The output timing prediction $\boldsymbol{k}_i \in \mathbb{R}$ is a real-valued scalar indicating the timing difference between the current pose and previous pose in the output style. This is determined by the temporal correspondence of pose pairs in the motion database and is computed through motion registration [Kovar and Gleicher 2003].

Together, $\{\boldsymbol{w}_i, \boldsymbol{p}_i, \boldsymbol{s}_i, \boldsymbol{c}_i, \boldsymbol{k}_i\}$ is the complete training information for the frame i training sample. Specifically $\{\boldsymbol{w}_i, \boldsymbol{s}_i\}$ describe the input, and $\{\boldsymbol{p}_i, \boldsymbol{c}_i, \boldsymbol{k}_i\}$ are for integrating output motion.

4.3 Network Design and Training

In the network architecture, there are three components: *Pose Network* (Θ_{Pose}), *Foot Contact Network* ($\Theta_{\text{FootContact}}$) and *Timing Network* ($\Theta_{\text{Timing}}$). The networks are all composed of three fully connected layers as illustrated in Fig.3, and their functionality can be defined as:

$$\Theta_{\mathbf{n}}(x_n;\alpha) = W_3\mathbf{ReLU}(W_2\mathbf{ReLU}(W_1\mathbf{ReLU}(W_0x_n + b_0) + b_1) + b_2) + b_3, \tag{1}$$

$$\Theta_{\text{Pose}}(x_{Pose};\alpha) = \boldsymbol{p}, \tag{2}$$

$$\Theta_{\text{Timing}}(x_{Timing};\alpha) = \boldsymbol{k}, \tag{3}$$

$$\Theta_{\text{FootContact}}(x_{FootContact};\alpha) = \boldsymbol{c}. \tag{4}$$

Inputs to the Pose Network (x_{Pose}) and Timing Network (x_{Timing}) are $\{\boldsymbol{w}, \boldsymbol{s}\}$ which contain 781 dimensions. Input to the Foot Contact Network ($x_{FootContact}$) is $\{\boldsymbol{w}\}$ which contains 774 dimensions: this is because the input motion's foot contact state is not a function of the desired style.

All hidden layer units use rectified linear activation functions (ReLU):

$$\mathbf{ReLU}(x) = max(0, x) \tag{5}$$

The goal of network training is to make the network generate y_n, given input x_n.

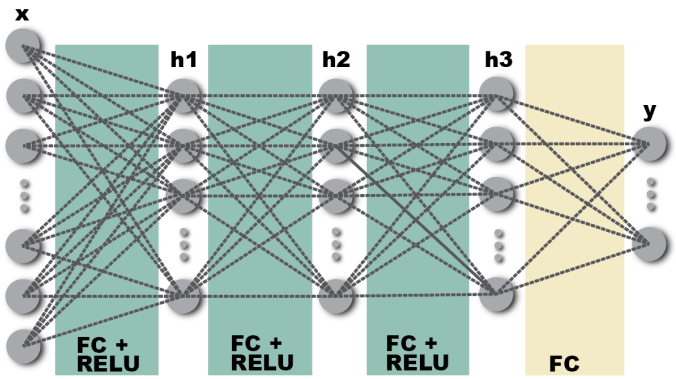

Fig. 3. Three-layered network structure for Pose Network, Foot Contact Network and Timing Network.

For the inputs and outputs of the pose network, Gaussian normalization is used. The structure of the network is three fully connected hidden layers employing 128 ReLUs. The model was trained over 20 epochs using minibatches of size 64 and a learning rate of 0.01. The mean squared error of the position predictions was used as a loss function. In order to improve the models ability to interpolate/extrapolate style, we adjusted half the samples in each minibatch. In these samples, the one-hot style vector was replaced with $s = \mathbf{0}^{|L|}$ and the model was trained to predict the the input pose.

The timing model takes input identical to the pose model. Its output is a single value indicating the timing prediction. This model's hidden layers have 64 units each, but it is otherwise identical to the pose model. Mean squared error of the timing prediction is used as the loss function. It was trained over 20 epochs with minibatches of size 64 and a learning rate of 0.01. In half the samples of each minibatch, the one-hot style vector $s = \mathbf{0}^{|L|}$ was used and the model was trained to predict a value 1.0.

The foot contact model was trained similarly, but it took as input only the joint window pose $\mathbf{w}_i$ and not the style vector. This model had three hidden layers with 32 units each. The model was trained over 20 epochs with minibatches of size 64 and used a binary cross entropy loss function and a learning rate of 0.01.

We implemented and trained the model on an i7 3.50 GHz, four core PC with a Geforce GTX 1070 graphics card. Models were implemented in PyTorch and training took approximately three hours to complete on the GPU. After training was complete, the database can be discarded entirely: only the models are necessary to store in memory. The total size of the three trained models is 931 KB.

4.4 Integration Pipeline

The output of our models are foot contact information, timing information, and the joint positions of the output pose in local space. It now remains to combine these pieces into a format that can be used to drive a mesh, such as skeleton joint angles.

The first step is to recover the global stylized joint positions. To do this we make use of the original trajectory of the input motion. For each input pose, we take the root velocity along the x and z axis and apply it to the root of the output pose. Similarly, we obtain the angular velocity of the input pose and apply it to the output pose. By keeping track of the current position and orientation of the output pose at each frame for use in the next frame, the global position of the output pose can be fully recovered. In this way we ensure that our styled output will always follow the same trajectory and face the same direction as the input pose, regardless of trajectory direction, speed, or duration.

After recovering the global positions of the stylized joints, they must be converted into joint angles. For this we use a pseudoinverse Jacobian-based inverse kinematics solver. It is then necessary to post-process the poses to remove footskate. This is done by incorporating the foot contact state predictions produced by the foot contact prediction network. When the foot first comes in contact with the ground, the target position of the toe joint is held constant until contact ceases. To prevent a jumping artifact after contact ends, the position of the toe joint is interpolated between the previous contact point and the model's prediction. Another iteration of Jacobian inverse kinematics is applied to obtain the final skeleton joint angles.

The final step is to incorporate the timing predictions into the animation. The timing prediction represents the timing difference between the current pose and the previous pose in the output style.

To do this, we keep track of the timing predictions for every pose in the output sequence. Then, we use spherical interpolation to derive the correct joint rotations for every frame. We use linear interpolation to derive the correct root position for every frame.

4.5 Style Interpolation and Extrapolation

After the model has been trained, novel motions can be used as input and the requested style specified by the values of the style vector. Although, during model training, the style vector weights are restricted to values of zero or one, this constraint does not apply after training is complete. During runtime prediction, style vector weights between zero and one can be used to transfer a smaller degree of style to the motion. By varying the values of the style vector over time, a walk which begins in a proud style can be smoothly blended into a depressed style (see Figure 1), for example. Additionally, there is no need to limit the values to between zero and one. In fact, we find that using values greater than one often results in reasonable poses with plausible and recognizable exaggerated styles (see Figure 5).

Table 1. We compare our approach to two pre-existing state-of-the-art style transfer approaches. We display the memory footprints and frames-per-second (FPS) reported in the original works. Our reported FPS is the average FPS over 2250 frames of walking data. Note that [Xia et al. 2015] achieved their FPS using a GPU, whereas the other two methods only use the CPU.

	Memory Footprint	**FPS Achieved**
[Xia et al. 2015]	290 MB	55*
[Yumer and Mitra 2016]	Not Reported	50
Our Method (PC)	931 KB	115

5 Experiments

We validate our method in three different ways. First, we compare the speed and memory footprint of our method with previous style transfer approaches (Table 1) and other neural-network based motion algorithms (Table 2). Second, we present comparative visual results to showcase the ability of our network to transform heterogeneous motion, to transform novel input styles, and to interpolate and extrapolate particular styles. Third, we present two perceptual user studies comparing our results to the original style algorithm of Xia et al. [2015]. The first study is intended to show that motion styles generated by our method are as recognizable as those generated by the original algorithm, and therefore that our method does not sacrifice style accuracy in exchange for its high speed and low memory footprint. The second study is intended to show that, when the values of the input style vector are extrapolated beyond 1.0, the resulting styles are perceived as more expressive than those output by Xia et al. [2015].

5.1 Performance

During performance testing, a series of joint positions were fed into the model and animations (in BVH format) were generated from the predictions. While the speed of the algorithm depends upon the desired quality of the inverse kinematic results, we found that suitable, jitter-free results could be produced very quickly. Using a 2,250 frame walking animation as input, we obtained an average runtime speed of 115 FPS using a single thread on an i7 3.5GHz CPU with four cores.

Table 1 compares the speed and memory footprint of our method to the approaches of Xia et al. [2015] and Yumer and Mitra [2016], a *spectral style transfer* technique. Our algorithm achieves the highest FPS and, because it does not need the motion database at runtime, has a memory footprint less than 0.5% the size of [Xia et al. 2015]'s.

We tested the speed of a forward pass of the neural networks on an iPhone 7 Plus and using a single thread on an i7 3.5GHz CPU with four cores. Table 2 compares the memory footprint and forward pass speed to other neural-network based motion algorithms.

5.2 Visual Results

While we refer the reader to the accompanying video for a more exhaustive demonstration of the capabilities of our network, we present several figures showcasing our results. Figure 4 demonstrates the ability of our model to stylize heterogeneous actions. Clips taken from the input motion are shown in the top row. The bottom three rows show *depressed* versions of those poses. The second row shows the *depressed* style of [Xia et al. 2015] for comparison. The third row shows the results of our method when the depressed value of the style vector is set to 1.0. The third row show the results of our method when the depressed value of the style vector is set to 2.0.

Our method provides reasonable versions of stylized poses which are clearly different from the input poses. It does so using far less computation and memory than prior approaches (see Table 1). However, some subtleties of the motion, such as the upward arm direction in Figure 4.D, are

Table 2. We compare the speed and memory footprint of our networks to other state-of-the-art neural network-based motion algorithms. Note that the values reported for [Mason et al. 2018] are for the CP decomposition of a single residual style adaptor and does not include the full phase-functioned neural network [Holden et al. 2017b] that must be present in memory during runtime.

	Memory Footprint	**Neural Network Forward Pass**
Our Method (PC)	931 KB	0.0008s
Our Method (Mobile)	931 KB	0.002s
[Holden et al. 2017b] Cubic	10MB	0.0018s
[Holden et al. 2017b] Constant	125MB	0.0008s
[Mason et al. 2018]	126KB*	0.0011s
[Zhang et al. 2018]	22MB	0.002s

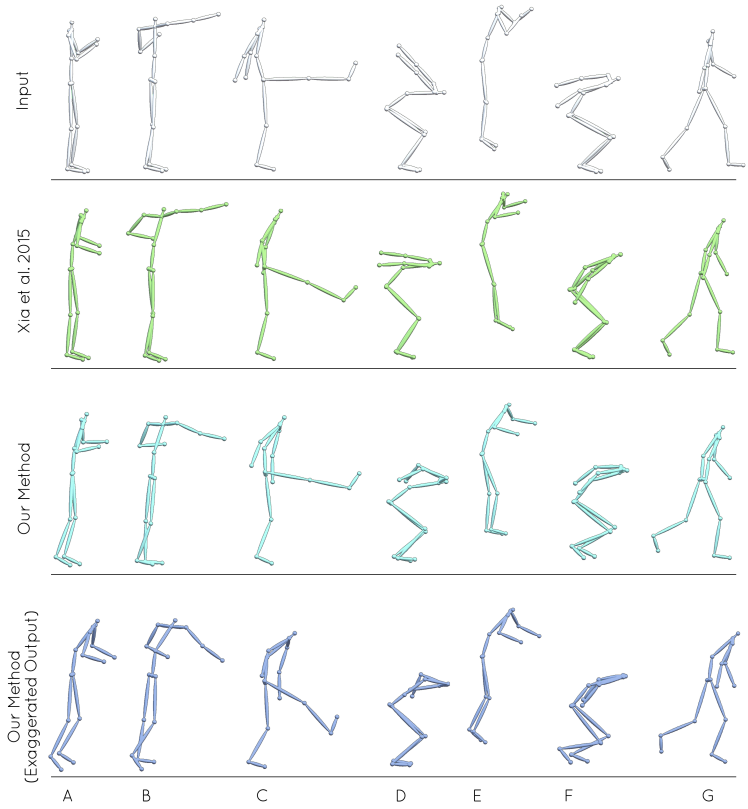

Fig. 4. Visual comparison between input data, a previous state-of-the-art approach, and our method. The examples above show a depressed style applied to standing (A), punching (B), kicking (C), pre-jump crouching (D), jumping (E), post-jump landing (F), and walking (G).

not well preserved by our method. When the value in the style vector is increased beyond 1.0, the network appears to likewise increase the spatial transformation that is being applied to the pose.

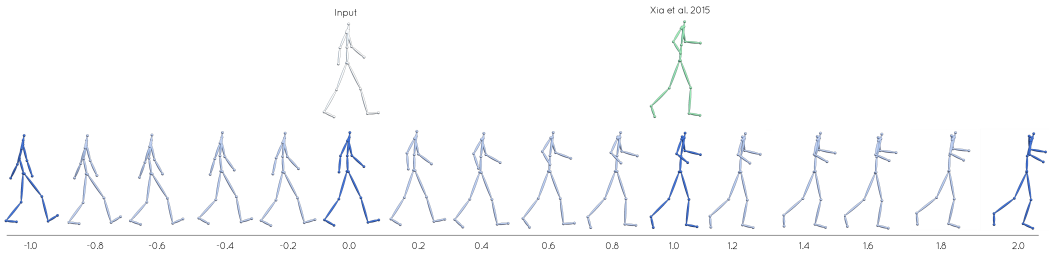

Fig. 5. Demonstration of our model's ability to interpolate and extrapolate stylistic transformations. The lower row of poses shows the output of our network when the *childlike* dimension of the style vector contains values between -1.0 and 2.0. For reference, the top row contains the neutral input pose and the *childlike* prediction of [Xia et al. 2015].

This can be seen in Figure 4.A: a slight forward hunch is exaggerated into a more pronounced hunch.

To further demonstrate this effect, Figure 5 shows the predictions of our network on the same pose when the style vector's *childlike* value is scaled between -1.0 and 2.0. Values between 0.0 and 1.0 can be viewed as interpolation between neutral and stylized, and values between 1.0 and 2.0 can be viewed as style extrapolation.

Although the network is not trained with any style vector with negative values, it is possible to input these after training is complete. Much as extrapolation exaggerates the style transforms applied by our network, inputting negative values appears to apply the opposite of the style transforms. Positive *childlike* values cause the character to extend its arms and lift its torso, while negative values cause it to slouch and move its arms further back. This style transfer operation is not so easily accomplished by other approaches and is a side-benefit of the one-hot style vector architecture.

Our style transfer algorithm can be applied to motions that are dissimilar to those used in the training data. Figure 6 shows our approach used to apply the *old* style to a *zombie* walk. The characteristic forward-raised arms of the zombie are retained by our algorithm, even as the torso is hunched over and the elbows bent.

By inserting non-zero values into multiple dimensions of the style vector, styles can be mixed together. Examples of *strutting-old*, *angry-proud*, and *childlike-depressed* styles are shown in the accompanying video.

5.3 User Experiment I

While the main benefits of our method are of speed and memory efficiency, it is nevertheless important to show that our method preserves the uniqueness of the different styles. If our method does, viewers ought to be able to correctly identify our method's output styles and Xia et al. [2015]'s output styles at a similar rate. Therefore, we designed the first experiment to directly compare the identification rates of the two methods.

The stimuli for this experiment were 14 videos, each depicting a neutral input motion followed by a stylized version. Seven of the stylized motions were created by our method (with style vector weights set to 1.0) and seven were created by Xia et al. [2015]'s method. The seven styles used are shown in the left-most column of Table 3. For all 14 stimuli, the same neutral walking motion was used as input.

We recruited 55 US-based, master-level workers from Amazon Mechanical Turk. Each was given a short set of instructions explaining that they were to identify the applied style from a list of seven

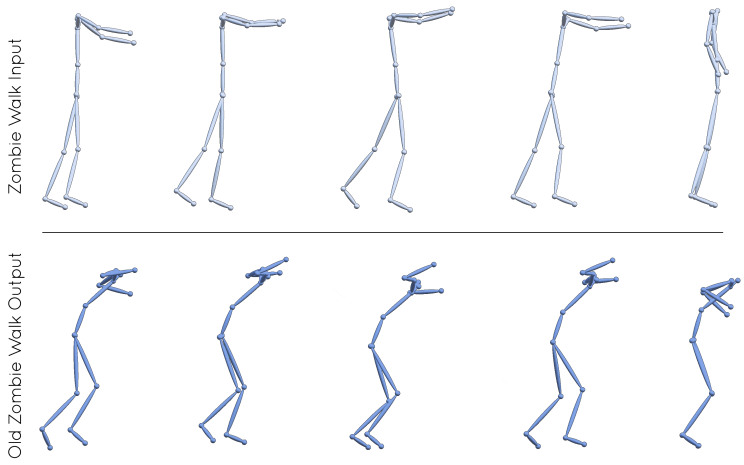

Fig. 6. Demonstration of our network creating an old style zombie walk. Although there were no zombie walks present in the training data, the model retains the characteristic forward-raised arms of the zombie walk, while bending the elbows and hunching the torso to create the old style.

Table 3. This table shows the identification counts collected in experiment 1. For each style (identified in the left-most column), workers were shown a motion styled by our algorithm and one styled by Xia et al. [2015]. The values presented here indicate how many workers correctly identified the style in both cases, in one of the two cases, or in neither case. The p-values, calculated using McNemar test for paired nominal data, show that the probability of misidentifying styles generated by the two methods's method are never significantly different.

		Identified (Xia's)	Misidentified (Xia's)	**p-value**
Angry	Identified (Ours)	25	13	0.79
	Misidentified (Ours)	9	18	
Childlike	Identified (Ours)	22	8	0.81
	Misidentified (Ours)	10	15	
Depressed	Identified (Ours)	40	2	0.18
	Misidentified (Ours)	7	6	
Old	Identified (Ours)	32	10	0.45
	Misidentified (Ours)	6	7	
Proud	Identified (Ours)	6	11	0.81
	Misidentified (Ours)	16	22	
Sexy	Identified (Ours)	2	8	0.58
	Misidentified (Ours)	5	40	
Strutting	Identified (Ours)	15	13	0.52
	Misidentified (Ours)	9	18	

choices. Each worker was then shown the series of 14 videos in a randomized order to prevent ordering effects. Workers were allowed to watch a video as many times as they wanted before selecting the style, but could not go back to rewatch previous videos or change previous answers.

Table 3 shows, grouped by style, the number of workers who correctly identified the stylized output from both methods (upper left), neither method (bottom right), our method only (upper right), or Xia et al. [2015]'s method only (lower left). In the majority of cases, workers either

Table 4. This table shows the data collected in User Experiment 2. In a paired comparison setting, workers were asked to indicate whether an extrapolated motion style generated using our method or a motion style generated using Xia et al.'s method was more expressive of the style.

Style	Ours	Xia's	Don't Know
Angry	27 (63%)	15 (34%)	1 (3%)
Childlike	33 (77%)	6 (14%)	4 (9%)
Depressed	31 (72%)	12 (28%)	0
Old	37 (86%)	6 (14%)	0
Proud	3 (7%)	40 (93%)	0
Sexy	17 (39%)	25 (58%)	1 (3%)
Strutting	27 (63%)	15 (34%)	1 (3%)

correctly identified the style from both methods or misidentified the style from both methods. In total, there were 65 cases in which the style from our system was correctly identified while that style from Xia et al. [2015] was misidentified. The opposite case occurred 62 times. To test whether the style transfer algorithm significantly influenced a style's misidentification rate, we employed a series of McNemar tests [McNemar 1947]. In the realm of machine learning, this test has been used to compare two classification models [Raschka 2018], but it is broadly suitable for data sets containing paired, dichotomous data [McCrum-Gardner 2008]. In our situation, the test determines whether the independent variable, style generation method, significantly affects the rate at which the style is misidentified.

The resulting p-value of each style's McNemar test is shown in Table 3. They range from 0.18 to 0.81. None of these values indicate significance at the 0.05 level: styles generated by our method are correctly identified at a rate not significantly different from those generated by Xia et al. [2015]. This indicates that our algorithm provides comparable style performance to Xia et al. [2015].

It is interesting to note the rate at which workers misidentified the different styles. It is important to note that workers were given no example style motions or style definitions on which to base their answers. Rather, they relied upon their own experiences and understandings of the style labels when making their selections. Correct identification rates were extremely low for the *Proud*, *Strutting*, and *Sexy* styles. These styles were frequently mistaken for each other; all three styles share elements of confidence and positivity, which may have led to the confusion.

5.4 User Experiment II

We seek to show that, by increasing the weights in the input style vectors, our algorithm is capable of extrapolating the style and creating more expressive versions of it. We therefore designed this experiment to test whether viewers thought our extrapolated styles were more expressive than those of the original algorithm. The stimuli for this experiment were seven video pairs. Both videos in a pair displayed the same motion with the same style. One of the videos was created using the technique of Xia et al. [2015], and the other was created using our method with an input style vector weight of 1.5. For all videos, the same neutral walking motion was used as input.

We recruited 43 US-based, master-level, Amazon Mechanical Turk workers. Each was given a short set of instructions indicating that they were to choose the video that was most expressive of a particular style. The workers were then shown a series of seven video pairs in a randomized order to prevent ordering effects. The side on which our method's video appeared was also randomized. Both videos were synchronized and played simultaneous. A prompt below the video identified the

style that was being shown and asked the users to select the more expressive video ('Left', 'Right', or 'Dont't Know'). Workers were allowed to watch the videos as many times as they wanted before answering, but could not go back to watch previous videos or change previous answers.

The results of the study are shown in Table 4. The majority of workers selected our extrapolated styles as 'more expressive' for the *Old*, *Depressed*, *Strutting*, *Childlike*, and *Angry* styles. This suggests that, for five of the seven styles studied, the method by which our algorithm extrapolates joint position transformations results in a more expressive motion style.

Our method was not successful in creating more expressive version of the *Proud* and *Sexy* style. Both the *Proud* style and the *Sexy* style share strong elements of torso axial rotation. Our method has difficulty reproducing such pose changes, as we reuse the input pose's forward vector (perpendicular to the vectors connecting the hips and the shoulders) to reconstruct the global positions of the output motion. Doing so removes any stylized change of the forward vector. Therefore, styles that rely on significant changes to the pose's forward vector cannot be adequately extrapolated by our system.

The *Proud* styles shares this characteristics of torso axial rotation. However, it also relies on extremely high arm swings as well. These swings position the hand joint positions much higher than in any other stylized pose. Our algorithm has difficulty predicting such extreme positions, even when the style weight is extrapolated.

Taken together, the results of this experiment suggest the types of styles that our method could be used to extrapolate: styles that rely heavily to torso flexion or extension (Such as *Old*, *Childlike*, *Depressed*, or *Strutting*), or less extreme changes in arm position (such as *Childlike* and *Strutting*). However, styles with very extreme changes in limb position, such as *Proud*, or with significant torso rotation, such as *Proud* and *Sexy* , cannot easily be extrapolated by our system.

6 Conclusion and Future Work

In this paper, we present a method for stylizing unlabeled heterogeneous human motion. The approach is fast enough to be performed in real time and requires a very small run-time memory footprint. This makes it the first heterogeneous human motion style transfer technique suited for use on a mobile device. An additional novel contribution of our approach is one-hot style vector-conditioned training, which allows the user to easily control the run-time output style, interpolating, extrapolating and blending styles as desired.

Our second user study showed that, while our algorithm can be successfully used for extrapolation, there are limitations to the types of styles for which this succeeds. Because our method transfers the input motion's root velocity and angular rotation directly to the output, these aspects of the motion will remain unedited. This ensures that both motions follow the same trajectory, but styles which rely on increased swaying or swaggering, such as *proud* and *sexy*, will be more difficult to exaggerate. If necessary, the root motion could be stylistically warped in a post-processing step.

The speed and memory improvements of our algorithm do come at a cost. While the stylized outputs are recognizable, the more extreme variations within a particular style may get lost. As can be seen in the accompanying video, the exaggeratedly high arm swings of the proud walk are not well reproduced by our method. Additionally, certain actions may be warped to more closely resemble a mean pose. This can be seen in the bent arms in Figure 4.D. This limitation will constrain the quality of the output and the number of different actions that should be present in the training data.

Our approach is slowed by the post-processing necessary to recover valid skeleton joint angles from the predicted joint positions. Ideally, the inverse kinematic solver quickly and easily converges to a stable and error-free solution. However, this is not always possible, as bone length constraints are not enforced in the model's predictions. In practice, we often found the Pose Prediction Model

to output poses with low bone length error. However, this error increases when large values are used in the style vector; this provides an upper limit on our model's style extrapolating capabilities. In the future, we would like to predict joint rotations directly (similar to [Pavllo et al. 2018]), as doing so would reduce the need for an inverse kinematics step.

Our method does not explicitly handle self-intersection artifacts. For styles which involve bringing the limbs closer together (such as the legs of the *sexy* style), can be problematic- especially as the style is extrapolated. This must be identified and removed in a post-processing step.

There are several important limitations and constraints to our method. While our approach can apply styles to motions, it cannot remove them: a *zombie* walk can be turned into a *childlike zombie* walk, but it cannot turned into a *neutral* walk. Additionally, our approach requires a large database of registered motion in different styles to create our training samples. Creating such a database from scratch or adding additional styles may be very costly. . Finally, novel input actions (such as jumping jacks or dancing) not in training data may not produce desirable output. In the future, our method could be improved by identifying such inputs and handling them appropriately. This could be accomplished by inputting the pose window along with a style vector of zeros, and checking whether the reconstructed output pose is sufficiently similar to the output pose before attempting to stylize it.

Acknowledgments

We would like to thank the authors of [Xia et al. 2015] for generously sharing their data and code with us.

References

Kenji Amaya, Armin Bruderlin, and Tom Calvert. 1996. Emotion from Motion. In *Proceedings of the Conference on Graphics Interface '96 (GI '96)*. Canadian Information Processing Society, Toronto, Ont., Canada, Canada, 222–229. http://dl.acm.org/citation.cfm?id=241020.241079

Andreas Aristidou, Qiong Zeng, Efstathios Stavrakis, KangKang Yin, Daniel Cohen-Or, Yiorgos Chrysanthou, and Baoquan Chen. 2017. Emotion Control of Unstructured Dance Movements. In *Proceedings of the ACM SIGGRAPH / Eurographics Symposium on Computer Animation (SCA '17)*. ACM, New York, NY, USA, Article 9, 10 pages. https://doi.org/10.1145/3099564.3099566

Matthew Brand and Aaron Hertzmann. 2000. Style Machines. In *Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '00)*. ACM Press/Addison-Wesley Publishing Co., New York, NY, USA, 183–192. https://doi.org/10.1145/344779.344865

Han Du, Erik Herrmann, Janis Sprenger, Noshaba Cheema, Somayeh Hosseini, Klaus Fischer, and Philipp Slusallek. 2019. Stylistic Locomotion Modeling with Conditional Variational Autoencoder. In *Eurographics 2019 - Short Papers*, Paolo Cignoni and Eder Miguel (Eds.). The Eurographics Association. https://doi.org/10.2312/egs.20191002

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 (SA '15)*. ACM, New York, NY, USA, Article 18, 4 pages. https://doi.org/10.1145/2820903.2820918

Ludovic Hoyet, Kenneth Ryall, Katja Zibrek, Hwangpil Park, Jehee Lee, Jessica Hodgins, and Carol O'Sullivan. 2013. Evaluating the distinctiveness and attractiveness of human motions on realistic virtual bodies. *ACM Transactions on Graphics (TOG)* 32, 6 (2013), 204.

Eugene Hsu, Kari Pulli, and Jovan Popović. 2005. Style Translation for Human Motion. *ACM Trans. Graph.* 24, 3, 1082–1089. https://doi.org/10.1145/1073204.1073315

Yejin Kim and Michael Neff. 2012. Component-based Locomotion Composition. In *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation (SCA '12)*. Eurographics Association, Goslar Germany, Germany, 165–173. http://dl.acm.org/citation.cfm?id=2422356.2422381

Lucas Kovar and Michael Gleicher. 2003. Flexible Automatic Motion Blending with Registration Curves. In *Proceedings of the 2003 ACM SIGGRAPH/Eurographics Symposium on Computer Animation (SCA '03)*. Eurographics Association, Aire-la-Ville, Switzerland, Switzerland, 214–224. http://dl.acm.org/citation.cfm?id=846276.846307

Zimo Li, Yi Zhou, Shuangjiu Xiao, Chong He, and Hao Li. 2017. Auto-Conditioned LSTM Network for Extended Complex Human Motion Synthesis. *CoRR* abs/1707.05363 (2017). arXiv:1707.05363 http://arxiv.org/abs/1707.05363

Libin Liu and Jessica Hodgins. 2017. Learning to schedule control fragments for physics-based characters using deep q-learning. *ACM Transactions on Graphics (TOG)* 36, 3 (2017), 29.

Libin Liu and Jessica Hodgins. 2018. Learning Basketball Dribbling Skills Using Trajectory Optimization and Deep Reinforcement Learning. *ACM Trans. Graph.* 37, 4, Article 142 (July 2018), 14 pages. https://doi.org/10.1145/3197517.3201315

Ian Mason, Sebastian Starke, He Zhang, Hakan Bilen, and Taku Komura. 2018. Few-shot learning of homogeneous human locomotion styles. *Computer Graphics Forum* 37, 7, 143–153.

Evie McCrum-Gardner. 2008. Which is the correct statistical test to use? *British Journal of Oral and Maxillofacial Surgery* 46, 1 (2008), 38–41.

Rachel McDonnell, Sophie Jörg, Joanna McHugh, Fiona Newell, and Carol O'Sullivan. 2008. Evaluating the Emotional Content of Human Motions on Real and Virtual Characters. In *Proceedings of the 5th Symposium on Applied Perception in Graphics and Visualization (APGV '08)*. ACM, New York, NY, USA, 67–74. https://doi.org/10.1145/1394281.1394294

Quinn McNemar. 1947. Note on the sampling error of the difference between correlated proportions or percentages. *Psychometrika* 12, 2 (1947), 153–157.

Michael Neff, Yingying Wang, Rob Abbott, and Marilyn Walker. 2010. Evaluating the Effect of Gesture and Language on Personality Perception in Conversational Agents. In *Intelligent Virtual Agents*, Jan Allbeck, Norman Badler, Timothy Bickmore, Catherine Pelachaud, and Alla Safonova (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 222–235.

Rubens F Nunes, Joaquim B Cavalcante-Neto, Creto A Vidal, Paul G Kry, and Victor B Zordan. 2012. Using natural vibrations to guide control for locomotion. In *Proceedings of the ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games*. ACM, 87–94.

Dario Pavllo, David Grangier, and Michael Auli. 2018. QuaterNet: A Quaternion-based Recurrent Model for Human Motion. *CoRR* abs/1805.06485 (2018). arXiv:1805.06485 http://arxiv.org/abs/1805.06485

Xue Bin Peng, Pieter Abbeel, Sergey Levine, and Michiel van de Panne. 2018. DeepMimic: Example-Guided Deep Reinforcement Learning of Physics-Based Character Skills. *CoRR* abs/1804.02717 (2018). arXiv:1804.02717 http://arxiv.org/abs/1804.02717

Xue Bin Peng, Glen Berseth, KangKang Yin, and Michiel Van De Panne. 2017. Deeploco: Dynamic locomotion skills using hierarchical deep reinforcement learning. *ACM Transactions on Graphics (TOG)* 36, 4 (2017), 41.

Sebastian Raschka. 2018. Model Evaluation, Model Selection, and Algorithm Selection in Machine Learning. *CoRR* abs/1811.12808 (2018). arXiv:1811.12808 http://arxiv.org/abs/1811.12808

Harrison Jesse Smith and Michael Neff. 2017. Understanding the impact of animated gesture performance on personality perceptions. *ACM Transactions on Graphics (TOG)* 36, 4 (2017), 49.

Graham W. Taylor and Geoffrey E. Hinton. 2009. Factored Conditional Restricted Boltzmann Machines for Modeling Motion Style. In *Proceedings of the 26th Annual International Conference on Machine Learning (ICML '09)*. ACM, New York, NY, USA, 1025–1032. https://doi.org/10.1145/1553374.1553505

Munetoshi Unuma, Ken Anjyo, and Ryozo Takeuchi. 1995. Fourier Principles for Emotion-based Human Figure Animation. In *Proceedings of the 22Nd Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '95)*. ACM, New York, NY, USA, 91–96. https://doi.org/10.1145/218380.218419

Ruben Villegas, Jimei Yang, Duygu Ceylan, and Honglak Lee. 2018. Neural Kinematic Networks for Unsupervised Motion Retargetting. *CoRR* abs/1804.05653. arXiv:1804.05653 http://arxiv.org/abs/1804.05653

Yingying Wang and Michael Neff. 2015. Deep Signatures for Indexing and Retrieval in Large Motion Databases. In *Proceedings of the 8th ACM SIGGRAPH Conference on Motion in Games (MIG '15)*. ACM, New York, NY, USA, 37–45. https://doi.org/10.1145/2822013.2822024

Yingying Wang, Jean E Fox Tree, Marilyn Walker, and Michael Neff. 2016. Assessing the impact of hand motion on virtual character personality. *ACM Transactions on Applied Perception (TAP)* 13, 2 (2016), 9.

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.

He Zhang, Sebastian Starke, Taku Komura, and Jun Saito. 2018. Mode-adaptive neural networks for quadruped motion control. *ACM Transactions on Graphics (TOG)* 37, 4 (2018), 145.