Fast Neural Style Transfer for Motion Data

Fast Neural Style Transfer for Motion Data

Daniel Holden*
University of Edinburgh

Ikhsanul Habibie †
University of Edinburgh

Taku Komura‡
University of Edinburgh

Ikuo Kusajima §
University of Tokyo

April 11, 2017

Abstract

We present a fast, efficient technique for performing neural style transfer of human motion data using a feedforward neural network. Typically feedforward neural networks are trained in a supervised fashion; both specifying the input and desired output simultaneously. For tasks such as style transfer this data may not always be available and so a different training method is required. We present a method of training a feedforward neural network making using of a loss network; in this case a convolutional autoencoder trained on a large motion database. This loss network is used to evaluate a number of separate error terms used in training the feedforward neural network. We compute a loss function in the space of the hidden units of the loss network that is based on style difference and motion-specific constraints such as foot sliding, joint lengths, and the trajectory of the character. By back-propagating these errors into the feedforward network we can train it to perform a transformation equivalent to neural style transfer. Using our framework we can transform the style of motion thousands of times faster than previous approaches which use optimization. We demonstrate our system by transforming locomotion into various different styles.

Keywords: motion capture, style transfer, deep learning, machine learning

*email:s0822954@sms.ed.ac.uk
†email:abie.ikhsan@gmail.com
‡email:tkomura@ed.ac.uk
§email:kusajima@ynl.t.u-tokyo.ac.jp

1 Introduction

Motion style transfer is a technique for converting the motion of an actor to that of a different character, for example to a character that is old, depressed, happy, or hurt etc. Automating this process can save animators a lot of time since they do not need to create many different variations of motions based upon which character is performing such a motion. Instead they can produce a single set of motions which are automatically adapted for use with different characters.

Many previous techniques for style transfer have been developed. Most of these methods are data-driven and require a set of corresponding motions both in the neutral style and the characterized style. These motions must be temporally aligned so that associated poses can be computed and a regression of some form learned. Constructing such a data-set can be tedious for artists and even with automatic methods, obtaining a good temporal alignment between motions can be difficult or require significant manual intervention by a technical developer.

Neural style transfer, first introduced by [1] and adapted for motion data by [2] uses a deep neural network to perform the style transfer task, solving an optimisation problem over the neural network hidden units to produce motion in the style of one clip, but with the content of another. This method over-

comes a number of the issues with conventional style transfer methods. Firstly, it only requires a single exemplar motion to represent the style rather than a database of corresponding clips. Secondly, it does not require any kind of alignment between content and style clips, instead calculating the style implicitly by taking an average over all the frames of the motion's *Gram matrix*. Both of these things are appealing to developers as they reduce the amount of data preparation required.

One of the disadvantages of neural style transfer has conventionally been the speed. Instead of performing a regression, neural style transfer requires solving an optimisation problem which is computationally expensive and may take a long time.

In this paper we present a technique inspired by the work of [3] that removes this disadvantage of speed while retaining the other advantages of the original style transfer scheme [1] . To do this we train a fast feedforward neural network to perform a regression task satisfying the constraints of the original optimisation problem, as well as new motion-specific constraints. We also replace the pre-trained classification network with an auto-encoding convolutional network as used in [2]. This acts as a loss network which has the advantage of being trained unsupervised, and is additionally used to fix any artifacts or noise that may be present in the produced motion.

In summary, our contribution is the following:

- A fast, efficient neural style transfer technique designed for motion data.

2 Related Work

In this section we first review methods in computer graphics about motion style transfer. Then we review the use of deep learning methods in transformation of data with an emphasis on style transfer. Finally, we will discuss previous motion synthesis methods based on deep learning techniques.

2.1 Motion Style Transfer

Motion style transfer is an old problem in computer animation, with the idea being to import a style from some motion clip and apply it to some other motion. Motion style transfer is very useful especially for applications such as computer games, where we wish to minimize the amount of motion data in the package.

One approach is to handle the motion in the frequency domain: Unuma et al. [4] and Bruderlin et al [5]. propose to transform the motions in the frequency domain. Neutral motions are converted into motions with different emotions by transferring the difference of the Fourier coefficients. Pullen and Bregler [6] define the style of the motion by the high frequency elements of the motion data and add them to a novel motion to transfer the style. Yumer and Mitra [7] slide a window along the motion and apply a FFT to extract the high frequency data that represents the style.

Another stream of work for motion style transfer is to use dynamic models: Hsu et al. [8], use a linear time invariant model to produce a time series model where the style is embedded. Min et.al. [9] propose a multilinear analysis approach to synthesize and transfer motion styles between actors. This method can be used to reduce ambiguity as the models can represent the motion data in a low-dimensional space. Another advantage of this method is the omission of the foot contact definition to prevent the foot sliding problem as the model is constructed from motion registration. The work by Xia et al. [10] shows that style transfer of human motion can be performed in real-time by constructing a local mixture of auto-regressive models of styles and contents that can be used to stylise a sequence of different motion contents such as a walking motion immediately followed jumping. The idea is to calculate the closest possible style from a given current motion using the trained model which is then used to predict the motion of the next frame.

2.2 Feedforward Transformation and Style Transfer

Using feedforward neural networks for data transformation has recently been applied to various image processing tasks. Related works include the use of convolutional neural networks to transform low-resolution images to produce high-resolution images [11], coloring of grayscale images [12], and segmentation or semantic scene understanding of an image [13, 14, 15]. Like other neural network approaches, once the models are trained they can easily and quickly be used to perform similar transformations to new input data.

Gatys et al. [1] show that the concept of image transformation using deep neural networks can be used to combine distinct elements of an image by using method known as style transfer. Their work shows that convolutional neural networks can be used to combine two images by transferring the style of one image to the content of another image. The combined image manages to successfully capture the visual style of one image while preserving the content of the other image. As shown by Zeiler and Fergus [16], each convolutional layer of the network captures the shape representation of the objects in the image with increasing levels of detail. This feature of the network can then be used to capture the content representation of the data. On the other hand, the style in the image data is represented as the texture characteristics such as color and local features, generally encoded by the Gram matrix.

Recently Johnson et al. [3] proposed a deep neural network structure with a perceptual loss function that can be used to carry out image transformation tasks such as style transfer and production of high-quality super-resolution images. They suggest that a loss function based on perceptual difference can be used to improve the performance compared to the common per-pixel mean squared error comparison. Their method produces style-transferred images comparable to [1] but is faster by a three orders of magnitude. We adopt this approach, but use motion specific constraints and an additional manifold projection step to make it suitable for use with motion data.

2.3 Deep neural network for human motion analysis

Before the widespread success in various research areas such as image transformation, the popularity of deep neural networks was from their achievement in obtaining state-of-the-art performances in recognition problems, especially for image and speech recognition. These successes can be generalized and used for other problems including classification and recognition of human motion data [17].

One of the earliest human motion synthesis techniques using neural networks is the conditional Restricted Boltzmann Machine (cRBM) [18] and the Recurrent Temporal Restricted Boltzmann Machine (RTRBM) [19]. While both approaches successfully produce human motion data, the motions are very noisy due to the per-frame sampling and additionally often converge to an average pose due to the ambiguity in the mapping. Fragkiadaki et al. [20] propose a variant of Recurrent Neural Networks to learn and synthesize human motion data known as the Encoder-Recurrent-Decoder network. Work by Holden et al. [21] shows that a convolutional autoencoder can be used to learn a manifold of human motion. This has many purposes in research including reconstructing, cleaning, or denoising motion data. Furthermore, Holden et al. [2] combine this autoencoder with a feedforward neural network to regress from high-level user inputs to full body motion data. They then provide a framework to edit the generated motions using the motion manifold and by optimizing the motion in the hidden unit space to satisfy constraints such as bone-length and foot sliding. This editing process can also be used to convert the motion style using the Gram matrix in a scheme similar to Gatys et al. [1].

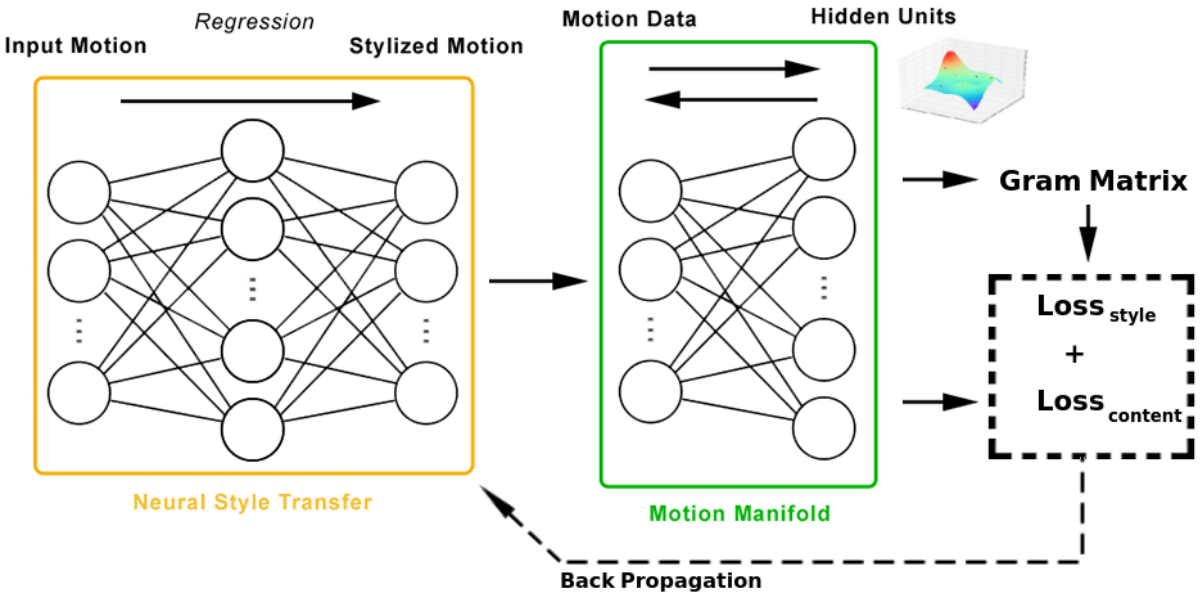

Figure 1: An overview of the two networks used by our system. Shown at the left in orange is the *transformation network* $\mathcal{T}$, a feed forward convolutional network which performs the style transformation. Shown at the right in green is the *loss network* $\mathcal{L}$, a convolutional autoencoder which represents a manifold over human motion and has two tasks - firstly to help compute the loss between motion content and style, and secondly to re-project motion onto the manifold to fix any small artefacts resulting from the style transfer.

3 Methodology

Our proposed network architecture is similar to the image transformation technique proposed by Jonhnson et al. [3]. An overview of our system is shown in Fig. 1. It consists of two separate neural network structures, a convolutional autoencoder network which serves as the *loss network* $\mathcal{L}$ and a feed forward convolutional network to perform motion style transformation which we refer as the *transformation network* $\mathcal{T}$.

3.1 Data Representation

Each pose is represented by the 21 joint positions in three-dimensional Euclidean space local to the character root projected onto the floor plane. Appended to this are three additional variables for turning velocity, and forward and sideways velocities. This gives us 66 degrees of freedom (DOF) in total per pose. While our proposed framework can be used to process motion data of any length - for training purposes motion data is split into overlapping windows of 240 frames at 60 frames per second. Any motions with less than 240 frames are padded with the first and last frames.

3.2 Loss Network

Contrary to Johnson et al. [3] who use a pre-trained image classification network, the *loss network* $\mathcal{L}$ in our system is a convolutional autoencoding network trained to reconstruct the motion data $\mathbf{X}$ which it receives as input. The purpose of using such a net-

work is to learn the latent, manifold structure of the motion database [21]. It allows us to partially emulate the classification network used in Johnson et al. [3] but in an unsupervised fashion. In Holden et al. [2], it was shown that a single convolution and pooling layer was enough to effectively capture motion content style in the latent variables, and as such a deeper or more complex network is not required. We therefore follow their construction which we will now briefly revise. The loss network is built from a single convolution and pooling layer that uses a one-dimensional convolutional calculated over the temporal dimension. The encoding operation can be written as:

$$\mathcal{L}(\mathbf{X}) = ReLU(\mathbf{\Psi}(\mathbf{X} * \mathbf{W_0} + \mathbf{b_0})), \quad (1)$$

where the weight matrix $\mathbf{W_0}$ represents a tensor composed of m hidden units (in our case $m = 256$), d features (in our case $d = 66$), and a temporal filter width of w_0 (in our case $w_0 = 25$). The vector $\mathbf{b_0}$ represents the biases of the layer and the operation $\mathbf{\Psi}$ represents a max-pooling operation used to reduce the number of features over the temporal axis by taking the maximum value between two consecutive parameters. We use Rectified Linear Units (ReLU) [22] as an activation function to provide non-linearity to the network. Having a filter width of 25 corresponds to around half a second of motion after the pooling has taken place, which is a reasonable length of time to represent most movements or components of movement. As our initial input has 66 dimensions, using 256 hidden units allows us to represent motion using sparse basis, which are more natural than the fully correlated basis that are often found via methods such as PCA. Similarly, the decoding operation of the convolutional autoencoder is defined as follows:

$$\mathcal{L}^{\dagger}(\mathbf{H}) = (\mathbf{\Psi}^{\dagger}(\mathbf{H}) - \mathbf{b_0}) * \tilde{\mathbf{W}}_\mathbf{0} \quad (2)$$

where $\mathbf{H}$ are the hidden units produced by the forward operation, $\mathbf{\Psi}^{\dagger}$ is the depooling operation (in our case average depooling), and $\tilde{\mathbf{W}}_\mathbf{0}$ is the deconvolution operation.

To train the convolutional autoencoder, the input motion data $\mathbf{X}$ is encoded into the the hidden unit space, and then decoded back to reconstruct $\tilde{\mathbf{X}}$.

The loss function

$$Loss(\mathbf{X}, \theta) = \|\mathbf{X} - \mathcal{L}^{\dagger}(\mathcal{L}(\mathbf{X}))\|_2^2 + \alpha\|\theta\|_1 \quad (3)$$

is then back-propagated to optimize the network parameters $\theta = \{\mathbf{W_0}, \mathbf{b_0}\}$ using stochastic gradient descent, where the second term is the L_1 sparsity regularization term.

Training takes around 6 hours on a NVIDIA GeForce GTX 660 GPU but since we use the pre-trained network from [2], for the purposes of this paper, training is not required.

3.3 Transformation Network

The *transformation network* $\mathcal{T}$ is a simple three-layer convolutional neural network where each layer has a similar construction to the encoding layer of the loss network without the pooling operation. Each layer contains 128 hidden units and all but the final output layer use *ReLU* units as a non-linearity.

3.4 Training

The goal of this work is to train the transformation network $\mathcal{T}$ without explicitly providing the desired inputs and outputs. To do this we make use of the given loss network $\mathcal{L}$ to encode various constraints about how we wish for the transformed motion to be.

During training we define the loss function of the transformation network using the following terms:

Content To ensure the output of the *transformation network* contains the *content* of the input motion $\mathbf{I}$ we define content loss to be the difference between hidden unit values of the input motion when passed to the *loss network* $\mathcal{L}$ and the hidden unit values of the *transformed* motion $\mathcal{T}(\mathbf{I})$ when passed to the *loss*

network. This comparison can be thought of as encoding the distance along the motion manifold [21] between the transformed and un-transformed motion. This is scaled by some user specified weight c, in our case 1.0:

$$Loss_{content} = c\|\mathcal{L}(\mathbf{I}) - \mathcal{L}(\mathcal{T}(\mathbf{I}))\| \tag{4}$$

.

Style To ensure the output of the *transformation network* contains the *style* of the given style clip $\mathbf{S}$ we define the style loss to be the difference between the Gram matrix of the hidden unit values of the *transformed* motion when passed to the *loss network* and the Gram matrix of the hidden unit values of the *style* motion when passed to the *loss network*. This follows the work of Gatys et al. [1, 23] and comparison ensures the style of the motion produced by the transformation network matches that of the given style clip $\mathbf{S}$. This is scaled by some user specified weight s in our case 0.01:

$$Loss_{style} = s\|Gram(\mathcal{L}(\mathbf{S})) - Gram(\mathcal{L}(\mathcal{T}(\mathbf{I})))\|. \tag{5}$$

The Gram matrix is given by the following and represents the sum over the temporal axis i of the inner product of the hidden unit values:

$$Gram(\mathbf{H}) = \sum_i \mathbf{H}_i \mathbf{H}_i^T. \tag{6}$$

By pairing stylized clips to their associated content clips during training, a transformation network that applies the style in different contexts can be learned. For example, given an "angry" style, all training clips where the content contains running are paired with the "angry run" style, while all content clips containing walking are paired with the "angry walk" style. In this way, although a different network is required for each style, a single network can be trained to work for a variety of contents.

Constraints As well as the content and style, it is important that the transformed motion $\mathcal{T}(\mathbf{I})$ respects the constraints of human motion. This means there should be no foot sliding artifacts, the bone lengths must not vary, and the trajectory should be similar to the input motion in shape. We therefore calculate additional loss functions relating to constraints of human motion. All of these constraints must be scaled with respect to the units of the input data.

To remove foot sliding, when the feet are considered in contact with the ground (a variable previously labelled in the training data) we ensure that for each of the foot joints j, the local translational velocity $\mathbf{v}_r^{\mathcal{T}(\mathbf{I})}$ plus the local rotational velocity $\omega^{\mathcal{T}(\mathbf{I})} \times \mathbf{p}_j^{\mathcal{T}(\mathbf{I})}$ negates the velocity of the root of the character relative to the forward direction $\mathbf{r}'$:

$$Loss_{foot} = \sum_j \|\mathbf{v}_r^{\mathcal{T}(\mathbf{I})} + \omega^{\mathcal{T}(\mathbf{I})} \times \mathbf{p}_j^{\mathcal{T}(\mathbf{I})} + \mathbf{v}_j^{\mathcal{T}(\mathbf{I})} - \mathbf{r}'\|^2. \tag{7}$$

For each bone b consisting of joints j_1 and j_2 we calculate the distance between the joint positions in the joint space of the transformed motion given by $\mathbf{p}_{b_{j_1}}^{\mathcal{T}(\mathbf{I})}$ and $\mathbf{p}_{b_{j_2}}^{\mathcal{T}(\mathbf{I})}$, and calculate the mean squared difference of this with the given bone length l_b:

$$Loss_{bone} = \sum_b |\|\mathbf{p}_{b_{j_1}}^{\mathcal{T}(\mathbf{I})} - \mathbf{p}_{b_{j_2}}^{\mathcal{T}(\mathbf{I})}\| - l_b|^2. \tag{8}$$

Given desired trajectory velocities $\mathbf{v}_r'$ and desired turning angle velocities ω' we calculate the mean squared error of these and the trajectory velocities and turning angle velocities the transformed motion given by $\mathbf{v}_r^{\mathcal{T}(\mathbf{I})}$ and $\omega^{\mathcal{T}(\mathbf{I})}$:

$$Loss_{traj} = \|\omega^{\mathcal{T}(\mathbf{I})} - \omega'\|^2 + \|\mathbf{v}_r^{\mathcal{T}(\mathbf{I})} - \mathbf{v}_r'\|^2. \tag{9}$$

Training Thus the final loss function that takes into account the content, style, and motion constraints is finally computed as follows:

$$\begin{aligned} Loss = Loss_{content} + Loss_{style} + Loss_{foot} \\ + Loss_{bone} + Loss_{traj}. \end{aligned} \tag{10}$$

Given this loss function the *transformation network* $\mathcal{T}$ is trained using stochastic gradient descent

Figure 2: Images generated by deep neural networks often contain high frequency noise and other artifacts (Top). Motion generated by our transformation network (Bottom) also sometimes has these problems (Grey), but we fix this using an additional step of projecting back onto the motion manifold previously found using the convolutional autoencoder (Blue).

with automatic derivative calculation performed using Theano. We use the momentum based optimization algorithm Adam [24] to improve learning speed. The *transformation network* is trained for 100 epochs using a locomotion dataset of about 20 minutes of locomotion data. Training for a single style takes around 20 minutes on a NVIDIA GeForce GTX 970. Once trained the transformation network can perform the style transfer task very quickly.

Manifold Projection Data produced using deep neural networks often contains small amounts of noise or other artifacts (See Fig. 2). For images these small errors may be acceptable but for motion data this greatly impacts the visual quality of the result. To solve this problem we re-purpose the pre-trained convolutional autoencoder used as the loss network and, following the work of [21], pass the transformation motion through it to project it back onto the motion manifold, effectively removing any undesirable noise or artifacts.

Method	**Frames**	**Runtime**	**FPS**
Our Method	240	0.012 sec	21818.18
Our Method	480	0.014 sec	33802.81
Our Method	960	0.017 sec	55813.95
Holden et al.	240	15.219 sec	15.76
Holden et al.	480	20.341 sec	23.59
Holden et al.	960	33.371 sec	28.76

Table 1: Performance comparison between our method and [2].

4 Experimental Results

In this section we demonstrate some of the results of our method. We test our method on ten different styles and various kinds of locomotion including walking, jogging, and running. As well as a a qualitative evaluation we compare our results to the most similar previous work - the optimisation based style transfer technique presented in Holden et al. [2]. For a more detailed look at the results, the readers are referred to the supplementary video.

In Fig. 3 we show a selection of style transfer results for different styles of locomotion including *old man*, *zombie*, *injured*, and *depressed*.

In Fig. 4 we show a comparison to [2]. Our method produces results which are in most cases visually very similar but which can be generated in a much shorter runtime.

In Table 1 we show a breakdown of the runtimes of our method and that of [2]. Our method has a vastly superior runtime and better scaling characteristics due to the implicit parallelism of the transformation network. All performance characteristics were recorded on a NVIDIA GeForce GTX 660 GPU.

5 Limitations & Future Work

Unlike [2] our method requires for training a database of motion data similar in content to the kind of motions that will be supplied at runtime. For example

training our method to perform style transfer for locomotion data requires a database of locomotion. For more specific motions it may be difficult or undesirable to acquire such a database.

Our method still requires the whole motion to be specified up-front and as such will require further adaptations for use in interactive applications. Additionally, while neural style transfer requires very little manual intervention and no alignment of data, compared to previous style transfer techniques it can be difficult to control from an artistic standpoint and often there is no clear way to fix undesirable results.

Experimentally we found our method does produce as good results on motion that is not primarily locomotion E.G. punching and kicking. Since Neural Style Transfer is difficult to understand and control, future experimentation is required to see how it can most effectively be applied to different kinds of motion.

The manifold projection operation used to remove artefacts from the stylized motion may introduce foot sliding. In our results we don't make an attempt to remove this - showing the raw output of the system - but for use in production this must be removed via some fast post process such as analytical IK.

The framework presented in this paper is specific to style transfer but it may be useful to learn many more other transformations of motion data given a single exemplar (I.E. in our case this is the single style clip). For example it may be possible to use this framework to transform motions for use on characters of different sizes, weights, or structures.

6 Conclusion

In conclusion we present a fast neural style transfer algorithm that can be used to perform style transfer on a variety of styles with a much faster runtime than previous methods.

References

[1] L. A. Gatys, A. S. Ecker, and M. Bethge, "A neural algorithm of artistic style," *arXiv preprint arXiv:1508.06576*, 2015.

[2] D. Holden, J. Saito, and T. Komura, "A deep learning framework for character motion synthesis and editing," *ACM Trans. Graph.*, vol. 35, pp. 138:1–138:11, 2016.

[3] J. Johnson, A. Alahi, and L. Fei-Fei, "Perceptual losses for real-time style transfer and super-resolution," *arXiv preprint arXiv:1603.08155*, 2016.

[4] M. Unuma, K. Anjyo, and R. Takeuchi, "Fourier principles for emotion-based human figure animation," in *Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*, pp. 91–96, ACM, 1995.

[5] A. Bruderlin and L. Williams, "Motion signal processing," in *Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*, pp. 97–104, ACM, 1995.

[6] K. Pullen and C. Bregler, "Motion capture assisted animation: Texturing and synthesis," in *ACM Transactions on Graphics (TOG)*, vol. 21, pp. 501–508, ACM, 2002.

[7] M. E. Yumer and N. J. Mitra, "Spectral style transfer for human motion between independent actions," *ACM Trans. Graph.*, vol. 35, pp. 137:1–137:8, July 2016.

[8] E. Hsu, K. Pulli, and J. Popović, "Style translation for human motion," in *ACM Transactions on Graphics (TOG)*, vol. 24, pp. 1082–1089, ACM, 2005.

[9] J. Min, H. Liu, and J. Chai, "Synthesis and editing of personalized stylistic human motion," in *Proceedings of the 2010 ACM SIGGRAPH symposium on Interactive 3D Graphics and Games*, pp. 39–46, ACM, 2010.

[10] S. Xia, C. Wang, J. Chai, and J. Hodgins, "Realtime style transfer for unlabeled heterogeneous

human motion," *ACM Transactions on Graphics (TOG)*, vol. 34, no. 4, p. 119, 2015.

[11] C. Dong, C. C. Loy, K. He, and X. Tang, "Learning a deep convolutional network for image super-resolution," in *European Conference on Computer Vision*, pp. 184–199, Springer, 2014.

[12] Z. Cheng, Q. Yang, and B. Sheng, "Deep colorization," in *Proceedings of the IEEE International Conference on Computer Vision*, pp. 415–423, 2015.

[13] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3431–3440, 2015.

[14] D. Eigen and R. Fergus, "Predicting depth, surface normals and semantic labels with a common multi-scale convolutional architecture," in *Proceedings of the IEEE International Conference on Computer Vision*, pp. 2650–2658, 2015.

[15] C. Farabet, C. Couprie, L. Najman, and Y. LeCun, "Learning hierarchical features for scene labeling," *IEEE transactions on pattern analysis and machine intelligence*, vol. 35, no. 8, pp. 1915–1929, 2013.

[16] M. D. Zeiler and R. Fergus, "Visualizing and understanding convolutional networks," in *European Conference on Computer Vision*, pp. 818–833, Springer, 2014.

[17] F. G. Harvey and C. Pal, "Semi-supervised learning with encoder-decoder recurrent neural networks: Experiments with motion capture sequences," *arXiv preprint arXiv:1511.06653*, 2015.

[18] G. W. Taylor and G. E. Hinton, "Factored conditional restricted boltzmann machines for modeling motion style," in *Proceedings of the 26th annual international conference on machine learning*, pp. 1025–1032, ACM, 2009.

[19] I. Sutskever, G. E. Hinton, and G. W. Taylor, "The recurrent temporal restricted boltzmann machine," in *Advances in Neural Information Processing Systems*, pp. 1601–1608, 2009.

[20] K. Fragkiadaki, S. Levine, P. Felsen, and J. Malik, "Recurrent network models for human dynamics," in *Proceedings of the IEEE International Conference on Computer Vision*, pp. 4346–4354, 2015.

[21] D. Holden, J. Saito, T. Komura, and T. Joyce, "Learning motion manifolds with convolutional autoencoders," in *SIGGRAPH Asia 2015 Technical Briefs*, p. 18, ACM, 2015.

[22] V. Nair and G. E. Hinton, "Rectified linear units improve restricted boltzmann machines," in *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, pp. 807–814, 2010.

[23] L. Gatys, A. S. Ecker, and M. Bethge, "Texture synthesis using convolutional neural networks," in *Advances in Neural Information Processing Systems*, pp. 262–270, 2015.

[24] D. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

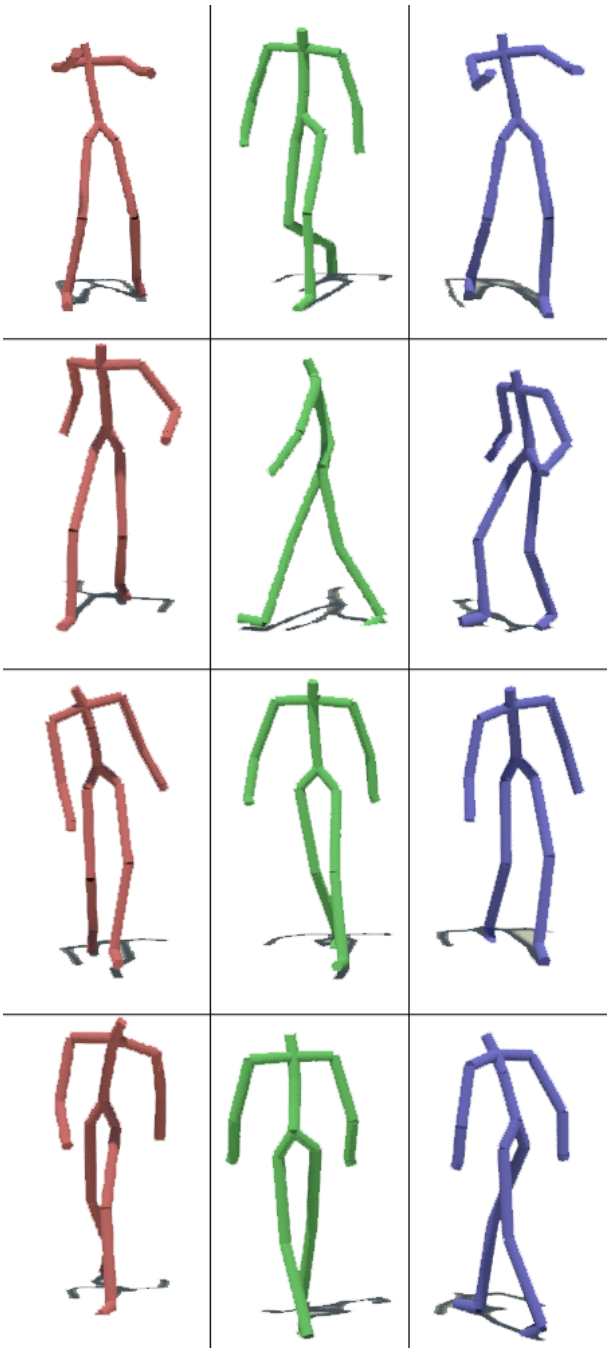

Figure 3: Some results of our method. Red: Style, Green: Content, Blue: Transfered. From top to bottom *zombie*, *old man*, *injured*, *depressed*.

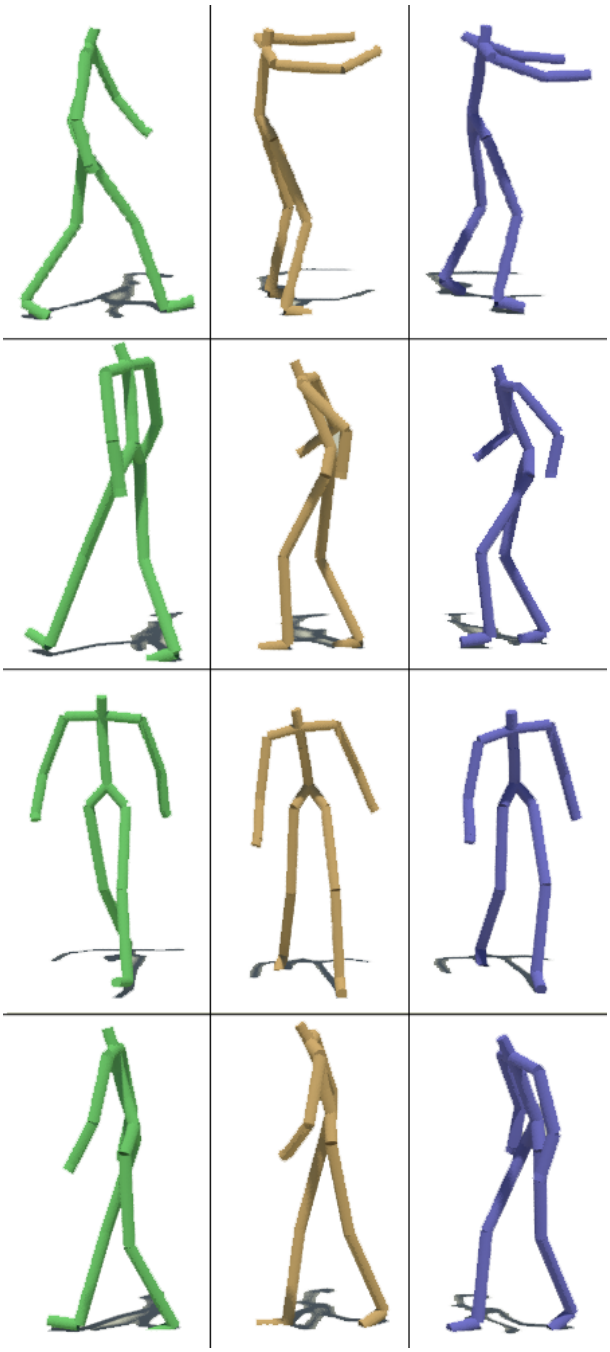

Figure 4: Visual comparison of our method to [2]. Green: Content, Orange: Holden et al., Blue: Our Method. From top to bottom *zombie*, *old man*, *injured*, *depressed*.