DHM2020
L. Hanson et al. (Eds.)

© 2020 The authors and IOS Press.
This article is published online with Open Access by IOS Press and distributed under the terms of the Creative Commons Attribution Non-Commercial License 4.0 (CC BY-NC 4.0).
doi:10.3233/ATDE200011

Variational Interpolating Neural Networks for Locomotion Synthesis

Janis SPRENGER [a,b,1], Han DU [a,b], Noshaba CHEEMA [a,b,c], Klaus FISCHER [a] and Philipp SLUSALLEK [a,b]

[a] *German Research Center for Artificial Intelligence (DFKI)*
[b] *Saarland University, Saarbrücken, Germany*
[c] *Max-Planck Institute for Informatics, Saarbrücken, Germany*

Abstract. Natural human locomotion contains variations, which are important for creating realistic animations. Most of all when simulating a group of avatars, the resulting motions will appear robotic and not natural anymore if all avatars are simulated with the same walk cycle. While there is a lot of research work focusing on high-quality, interactive motion synthesis the same work does not include rich variations in the generated motion. We propose a novel approach to high-quality, interactive and variational motion synthesis. We successfully integrated concepts of variational autoencoders in a fully-connected network. Our approach can learn the dataset intrinsic variation inside the hidden layers. Different hyperparameters are evaluated, including the number of variational layers and the frequency of random sampling during motion generation. We demonstrate that our approach can generate smooth animations including highly visible temporal and spatial variations and can be utilized for reactive online locomotion synthesis.

Keywords. Neural Networks, Procedural Animation, Variational Motion Synthesis

1. Introduction

Human locomotion is composed out of two characteristics: consistency and variation. It is quite often considered as a cyclic problem (e.g., [1, 2, 3]), as consecutive steps are quite similar while walking with the same velocity and direction. On the other hand, it is very hard for humans to accurately replicate the same motion twice. Hence every step during locomotion is slightly different. In order to generate natural locomotion animations, it is not only important to accurately and fluently replicate the mean motion but to generate the intrinsic variation as well. In a third-person gaming context, this might not be a major issue, as the user is controlling the avatar and thus is incorporating a certain amount of variation due to his input. However, as soon as the avatar is perceived for a long time with the same input (e.g. walking straight ahead) or is displayed from an outside angle, the lack of variation becomes apparent. This problem increases with the number of avatars using the same type of locomotion variation. In this case, even a very natural locomotion clip will appear robotic and unnatural. In the gaming context, for example, different walk-cycles are generated and randomly assigned to different avatars in order

[1] Corresponding Author. janis.sprenger@dfki.de

to diminish this effect. Considering advanced motion synthesis techniques, this approach is rarely possible.

The recent publications for reactive motion synthesis (e.g. [2, 3, 4]) generate motions of very high quality, but these approaches usually lack variability due to their deterministic nature. Given the same initialization and user input, all motions will appear the same. In novel application fields, like a digital reality [5] simulating pedestrian agents for testing autonomous driving systems, the variation is not only preferred but highly required. If an autonomous vehicle is trained or validated in a virtual scene using avatars with only a single animation clip, the systems will overfit or only be proven to work with this single animation. Hence, it is necessary to accurately replicate the variation apparent in reality.

Although there are already approaches using variational autoencoders, they lack either the ability to control the motion synthesis [6] or the fidelity of the generated motion is not comparable with the results of state-of-the-art models [7]. Currently, the most promising work for locomotion synthesis are **Feedforward Interpolating Neural Networks (FINN)**. These networks consist of a core motion network and an interpolation method [2, 3, 8]. The core motion network is a fully connected neural network and performs the regression step from the current to the next frame (feedforward step). The interpolation method is used to generate the optimal network weights of the motion network for the current regression step.

In this work, we are presenting a novel approach adapting the core motion network of a FINN to learn and reproduce the variation intrinsic to the training data while retaining the motion quality and responsiveness. Similar to a variational autoencoder the parameters of a hidden layer are considered to correspond to the mean and variance of a normal distribution. During model training as well as during motion synthesis, a random variable is sampled from a normal distribution and scaled by the parameter. Hence the model can learn the distribution inherent to the motion capture training dataset inside the hidden layer. The approach is based on a phase-functioned neural network [2]. We evaluated the impact of different numbers and positions of variational layers inside the network against adding random noise and a vanilla phase-functioned neural network. Our approach greatly increases the inter- and intra-individual variation of locomotion, while retaining the quality and naturalness of the original motion. This work has the following contributions:

- a novel approach of Variational Interpolating Neural Networks (VINN) is presented,
- evaluation of the impact of multiple hidden distributions in a single variational network,
- novel insights for the usage of variational generative networks with a high frame rate,
- novel approach to the evaluation of the variation of generated motions.

2. Background

2.1. Data-Driven Motion Synthesis

Motion capturing produces natural and high-quality motions that can be used to train data-driven synthesis models. Kovar et al. [9] used motion clips from a dataset and build

a *motion graph* encoding the different possibilities to reassemble the original clips. A constrained graph walk can generate target-specific animations based on the control input. There are multiple extensions for motion graphs, e.g. [10, 11]. Lee et al. [4] encode individual frames as motion states and model all possible transitions using a motion field rather than a graph. A Markov decision process (MDP) is built to model the possible decisions in terms of actions for each motion state. Motion synthesis can be considered as a concatenation of individual discrete poses over time. Using simple fully-connected regression models to reproduce this time-series data suffers from pose averaging. Similar intermediate poses in different sequences, e.g. when both legs are crossing each other in a walk cycle, are averaged and prevent a natural transition of frames. Due to missing information on the recent past, the network can become unable to distinguish the foot going forward and does not accurately continue the walk cycle. There are different approaches, trying to solve this problem using recurrent neural networks (RNNs) [12, 13, 14]. However, training RNNs reportedly is very complex, time-consuming, and uncontrolled generation of motion using RNNs suffers from dying out of motion and accumulation of error [12, 13, 2].

Recently, [2] proposed a novel network architecture that interpolates the network weights depending on the phase of the walk cycle. This methodology was further extended by [3] to generate complex quadruplet motion of dogs. A prove of concept of the same idea for reach motions was recently published by [15]. All of these approaches have in common, that the network weights of the regression network are interpolated and use a similar, fully-connected network structure for motion synthesis. While these forward interpolating neural networks (FINN) are providing a natural motion synthesis that can be controlled responsively, they are deterministic by nature and unable to create variations in the generated motion. Although the animation of a single avatar appears convincing, the animation of multiple avatars appears highly unnatural and robotic. Therefore, including natural variations as a statistical model is attractive for crowd animation and realistic simulation.

2.2. Statistical Motion Modeling

Another trend of approaches for motion synthesis task is to learn the distribution from motion capture data. The new motions can be generated by sampling and optimizing the learned distribution based on constraints. Many conventional models have been used to model the motion distribution, for instance, the variants of Hidden Markov Models (HMMs) [16, 17, 18], Gaussian Mixture Models (GMMs) in low-dimensional space [10, 19, 11, 20] and Gaussian Processes (GP) and its many variants for temporal variations [21, 22, 23]. Min and Chai [11] combine a motion graph [9] with statistical motion modeling. The motion data is modeled as a directed graph with each node being a cluster of semantically and structurally similar motion clips. Each edge in the graph represents the possible transition between clips. Motion clips in each motion primitive are projected into low-dimensional space and a GMM is used to learn the distribution. The transitions between motion primitives are modeled by Gaussian Processes. Recently, variational autoencoders (VAE) have been proposed as non-parametric, unsupervised approaches for manifold learning. Since it is a non-parametric model, no prior knowledge about data distribution is assumed. It uses the computational power of neural networks to learn a transform from a normal distribution to any kind of complex distribution. Holden et al.

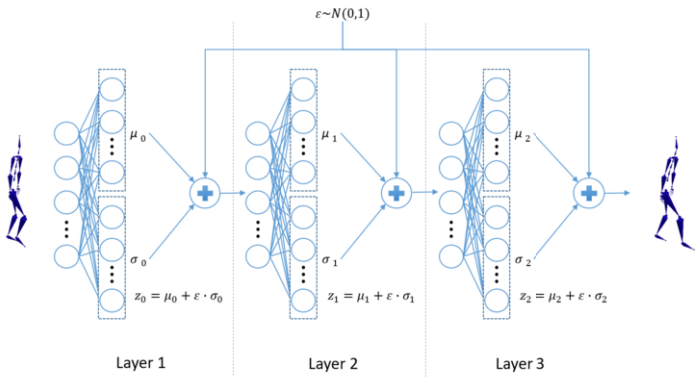


Figure 1. Network Structure using three Variational Layers (var1n2n3). A single sample is drawn from a normal distribution and used to scale the trained distribution in the hidden layers.

[24] first apply convolutional autoencoders to learn a continuous manifold space for a large, heterogeneous motion database. Other solutions are combining VAE with long short-term memory nodes (LSTM) [7] to model and predict motion sequences. These models, however, cannot model the high-level control of a user intuitively.

3. Variational Interpolating Neural Networks (VINN)

By considering animation as a concatenation of individual poses, we can consider the whole process as a time-series problem where each individual step can be predicted by a regression model. In order to allow a user to control the motion generation, a separate imperative controller is required, storing the global state of the avatar. Considering fully connected or feedforward interpolating neural networks, a single latent representation performs the regression steps deterministically. We propose a novel approach for learning not a single representation, but the distribution present in the training data. The approach is built upon a phase-functioned neural network ([2]), one of the state-of-the-art models for locomotion synthesis.

3.1. Neural Network Structure

For a single variational layer, the output of a fully connected layer is split into the mean (μ) and the standard deviation (σ) of a normal distribution. The output of the layer is computed by drawing a normal distributed variable $\varepsilon \sim N(0,1)$ and scaling it by the values computed in the fully connected layer:

$$v(\mu,\sigma,\varepsilon) = \mu + \varepsilon \cdot \sigma \tag{1}$$

Using a suitable split function to separate the mean and standard deviation (e.g. $split(x) = (x[0:0.5\cdot|x|], x[0.5\cdot|x|:|x|])$), a single variational layer for an input vector x, weights W and bias b can be defined as

$$\mathrm{VAR}(x,W,b) = v(split(Wx+b)) \tag{2}$$

Using the exponential rectified linear function [25] as an activation function, the variational layer can be written as:

$$\mathrm{VAR}^{(e)}(W,b,x) = v(split(ELU(Wx+b))) \tag{3}$$

Given an input parameter $x \in \mathbb{R}^n$, output parameters $y \in \mathbb{R}^m$ and a phase parameter $p \in \mathbb{R}$, we construct a three layer network as visualized in figure 1.
Mathematically, the network can be described as

$$\Phi(x;A) = \mathrm{VAR}_3(W_2,b_2,\mathrm{VAR}_2^{(e)}(W_1,b_1,\mathrm{VAR}_1^{(e)}(W_0,b_0,x))) \tag{4}$$

A network model A can be completely defined by $A = \{W_0 \in \mathbb{R}^{2h\times n}, W_1 \in \mathbb{R}^{2h\times h}, W_2 \in \mathbb{R}^{m\times h}, b_0 \in \mathbb{R}^{2h}, b_1 \in \mathbb{R}^{2h}, b_2 \in \mathbb{R}^m\}$. Here, $h = 512$ is the size of hidden state, $n = 301$ is the dimension of the input vector and $m = 245$ the dimension of the output vector. Similar to [2] we are employing a phase-functioned structure on this network. Hence, the network weights are computed by a phase function before each regression step, depending on the phase and the input parameters θ. In this case, a cyclic cubic Catmull-Rom spline is chosen as the phase function, using for control points $\theta = \{\alpha_0\ \alpha_1\ \alpha_2\ \alpha_3\}$. A definition of this function is shown in the appendix eq. (9).

3.2. Number of Variational Layers and Dimensionality of Random Sample

Variational autoencoders learn a single distribution in the latent space after encoding. Here, a fully connected network is utilized and hence, there are multiple possibilities to incorporate the variation. In practice, each conventional layer of a phase-functioned neural network (PFNN) [2] can be replaced with a variational layer. We configured network models for all different positions (e.g. *var1n3* starts with a variational layer, has a regular layer in between, and ends with a variational layer. For each pass, a single random sample is used to draw from the latent distributions to enable a correspondence between distributions. The dimensionality of the random sample is equal to the dimensionality of the hidden layer.

3.3. Input and Output Format

The network performs a single regression step from the past to the future frame. The input and output vectors are defined as a combination of control input and joint configurations. The input vector contains trajectory positions $p_i \in \mathbb{R}^{2f}$, trajectory directions $d_i \in \mathbb{R}^{2f}$ and gait controls $g_i \in \mathbb{R}^{f\times s}$ of $f = 12$ surrounding frames and the $s = 6$ target gaits (standing, walking, jogging, etc.). The past joint configuration is provided as joint positions $l_{i-1} \in \mathbb{R}^{3j}$ and joint velocities $v_{i-1} \in \mathbb{R}^{3j}$ for all $j = 31$ joints.

The output vector contains global translation $r_i^x \in \mathbb{R}^2$, the change of forward direction $r_i^a \in \mathbb{R}^2$ projected on the ground plane, the phase delta (elapsed phase) $\Delta_i \in \mathbb{R}$. In addition, it contains the predicted future trajectory positions $p_{i+1} \in \mathbb{R}^f$ and directions $d_i \in \mathbb{R}^f$. Last but not least the predicted next posture as joint positions $l_i \in \mathbb{R}^{3j}$ and velocities $v_i \in \mathbb{R}^{3j}$ for all $j = 31$ joints.

$$\begin{aligned} x_i &= \{p_i\ \ d_i\ \ g_i\ \ l_{i-1}\ \ v_{i-1}\} \in \mathbb{R}^{306} \\ y_i &= \{r_i^x\ \ r_i^a\ \ \Delta_i\ \ p_{i+1}\ \ d_{i+1}\ \ l_i\ \ v_i\} \in \mathbb{R}^{215} \end{aligned} \tag{5}$$

3.4. Random Sampling during Motion Synthesis

A simple motion controller similar to [2, 26] was implemented. During motion synthesis, the network is executed at 60 frames per second. Random sampling at such a high frequency is problematic for two reasons: first, drawing a random sample at each frame can produce jittering. Second, the more samples are drawn, the more will the average of these samples converge towards the mean of the distribution thus reducing the variability. Similar to Du et al. [20], a single step can be considered as motion primitive. Thus drawing a new random sample at each step rather than at each frame and keeping this sample constant throughout the step should generate variational steps rather than variational frames.

4. Evaluation

4.1. Dataset

All frames from the motion capture library published by Holden et al. [2] containing only the locomotion on a flat surface were utilized. After pre-processing, the raw motion capture data in the required format, 179,586 frame-pairs were extracted. The decision to focus on the reduced dataset was taken, to reduce the variability introduced by external factors, for example, the terrain.

4.2. Network Training

The network architecture was implemented using the TensorFlow framework [27]. Each network was trained on the same dataset for 50 epochs using the stochastic gradient descent algorithm Adam [28], already included in the framework.Dropout [29] was applied with a retention probability of 0.7. All models were trained in mini-batches of size 32. Cosine decay with warm restarts was utilized to adjust the learning rate over time. The initial learning rate was set to 0.0001. The restarts were performed at 10 and 30 epochs.

4.3. Measuring Variation

An experimental setup was utilized to compare the variation in locomotion generation. Using a vanilla PFNN, 10 seconds of straight forward walking were simulated to initialize a walk cycle. Based on this initialization all models generated 100 separate walk cycles of 4 seconds or about 5 steps. We measure three variables for each of the walk cycles. The **inter-individual variation**, comparing variation between the walk cycles for each time-step. For the baseline-method (PFNN) this variation should be zero. The **intra-individual variation**, describing variation throughout a single walk cycle. And the **error**, measuring violations of smoothness, foot drifting and large inconsistencies between steps (stumbling).

The **inter-individual variation** (V_{inter}) was computed by the sum

$$V_{\text{inter}} = V_{\text{inter pose}} + V_{\text{inter step}} \tag{6}$$

The *inter-pose variation* ($V_{\text{inter pose}}$) was measured as the median deviation of local joint positions of the mean joint configuration for each time step. The *inter step velocity* ($V_{\text{inter step}}$) was measured as the average standard deviation of step velocity of temporally aligned steps generated by the different models. The sum of both variables was used to measure the inter-individual variation.

The **intra-individual variation** (V_{intra}) was computed as the sum

$$V_{\text{intra}} = V_{\text{intra pose}} + V_{\text{intra step}} \tag{7}$$

The *intra-pose variation* ($V_{\text{intra pose}}$) was measured by first, grouping all generated local joint positions to their respective phase of the walk cycle (in 0.01 increments) and second computing the average deviation of joint positions for each phase increment in the same way, as the inter-pose variation. The *intra step velocity* ($V_{\text{intra step}}$) was measured as the average deviation of step velocity of all steps generated from the specific motion model.

The **error** was measured as the sum

$$\text{error} = e_{\text{smooth}} + e_{\text{drift}} + e_{\text{distance}} + e_{\text{duration}} \tag{8}$$

Smoothness (e_{smooth}) was computed as the average of the 95-percentile of velocities generated by each model. *Foot drifting* (e_{drift}) was computed for each generated walk cycle and averaged. *Step distance consistency* (e_{distance}) and *step duration consistency* (e_{duration}) are computed by taking the differences of distance and duration of consecutive steps. The resulting standard deviation of the differences should be reasonably small. The variables are computed by taking the standard deviation of all respective differences. Consistency and smoothness should be reasonably small but not zero, as there would be no dynamic movement in this case.

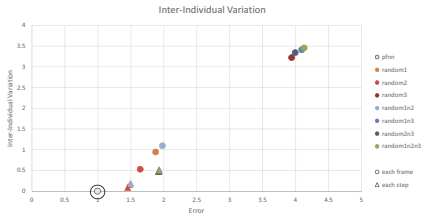
4.4. Numerical Results

We compare the VINN models against a vanilla PFNN implementation as well as random models, where instead of sampling from a distribution random noise is added to the layers of a PFNN, as this could generate variation as well. In addition, we compare sampling at each frame and sampling at each step. In figure 2 we present the results as scatter plots of variation measure versus the error. The baseline model produces zero inter-individual variation and only minor intra-individual variation. All random models generate motion with high error scores and do not necessarily generate more variation. Sampling at each step rather than each frame can double the variation scores without a large increase of error in the VINN models. The best performing model for inter- and intra-variation as well as with an error comparably with the baseline model contains only variational layers (var1n2n3).

4.5. Qualitative Results

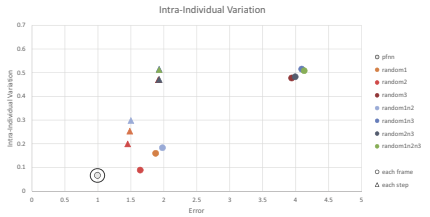
The different models were used to generate locomotion in different scenarios. The rendered videos of these scenarios can be found in the supplementary material.

(a) Scatterplot of the inter-individual variation vs error for variational models



(b) Scatterplot of the inter-individual variation vs error for random models

(c) Scatterplot of the intra-individual variation versus error for variational models



(d) Scatterplot of the intra-individual variation versus error for random models

Figure 2. Scatterplots of the variation versus error for inter- and intra-variation factors of all models. The baseline and the best performing model are encircled. All random models generated inconsistent and noisy motion.

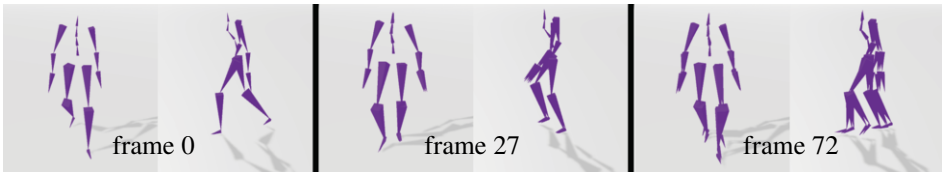


Figure 3. Development of motion variation within the first 72 frames (1.2 s). The display shows 5 overlayed skeleton walkers fixed at their hip position from front and side view. The motion is generated using single-step sampling with the vinn1n2n3 model.

Figure fig. 3 displays the result for the target model (vinn1n2n3). The qualitative analysis supports the numerical evaluation. Adding random noise creates a high amount of jitter when sampling at each frame. When sampling at each step, fewer variations in motion but a lot of floating can be observed, where the global translation did not match the body movements. In the case of VINN, the visible variation increases for all models, when sampling at each step. The body movement is corresponding to the global translation very accurately. Although there is a small numeric discontinuity at the specific frame of the change of random sample, the discontinuity is barely visible in the animations.

5. Discussion

The evaluation results suggest, that the proposed generative network can capture and replicate variation in motion best if a distribution is trained in each layer. In comparison to a vanilla PFNN, it generates a significant amount of variations. In comparison to

adding random noise at each layer, it generates consistent motions with respect to foot placement, step length, and smoothness.

We selected sufficiently straight walking examples from the training dataset and performed the same evaluation on these training samples. As the data was captured in a restricted space and not on a treadmill, there are only a few samples of straight-line walking. To find a sufficient amount of sequences for the evaluation, small deviations from a straight line were accepted. The results show an error of 0.93 and an intra-individual variation of 3.2. Although the variation is twice as large as generated by our model, it shows a comparable error. For future data capturing of locomotion, we suggest to include treadmill walking to have better means of evaluation, as the larger variation may be due to different trajectories.

Due to the high frame-rate (60fps) the variation is regressing to the mean if a random sample is drawn at each frame. Sampling at each step, however, creates natural motions with high variability. The discontinuity between two steps is comparably small and can be further reduced by smoothing the random sample. However, future work should investigate the usage of more temporally stable sampling methods.

The proposed approach demonstrates, that concepts of VAE can be integrated into a fully-connected interpolating network structure while retaining the controllability and visual quality of the original approach. This opens the potential for further experiments with similar network structures ([3, 8]). In addition, it builds a basis for future work in the direction of conditional variation and stylistic motion synthesis.

References

[1] Troje NF. Decomposing biological motion: A framework for analysis and synthesis of human gait patterns. Journal of Vision. 2002;2(5):2.

[2] Holden D, Komura T, Saito J. Phase-functioned Neural Networks for Character Control. ACM Trans Graph. 2017 Jul;36(4):42:1–42:13.

[3] Zhang H, Starke S, Komura T, Saito J. Mode-adaptive Neural Networks for Quadruped Motion Control. ACM Trans Graph. 2018 Jul;37(4):145:1–145:11.

[4] Lee Y, Wampler K, Bernstein G, Popović J, Popović Z. Motion Fields for Interactive Character Locomotion. ACM Trans Graph. 2010 Dec;29(6):138:1–138:8.

[5] Dahmen T, Trampert P, Boughorbel F, Sprenger J, Klusch M, Fischer K, et al. Digital reality: a model-based approach to supervised learning from synthetic data. AI Perspectives. 2019;1(1):1–12.

[6] Motegi Y, Hijioka Y, Murakami M. Human Motion Generative Model Using Variational Autoencoder. International Journal of Modeling and Optimization. 2018;8(1).

[7] Habibie I, Holden D, Schwarz J, Yearsley J, Komura T. A Recurrent Variational Autoencoder for Human Motion Synthesis. 2017.

[8] Starke S, Zhang H, Komura T, Saito J. Neural state machine for character-scene interactions. ACM Transactions on Graphics. 2019 Nov;38(6):1–14. Available from: https://dl.acm.org/doi/10.1145/3355089.3356505.

[9] Kovar L, Gleicher M, Pighin F. Motion Graphs. ACM Trans Graph. 2002 Jul;21(3):473–482.

[10] Lee J, Chai J, Reitsma PSA, Hodgins JK, Pollard NS. Interactive Control of Avatars Animated with Human Motion Data. ACM Trans Graph. 2002 Jul;21(3):491–500.
[11] Min J, Chai J. Motion Graphs++: A Compact Generative Model for Semantic Motion Analysis and Synthesis. ACM Trans Graph. 2012 Nov;31(6):153:1–153:12.
[12] Fragkiadaki K, Levine S, Felsen P, Malik J. Recurrent Network Models for Human Dynamics. In: Proceedings of the 2015 IEEE International Conference on Computer Vision (ICCV). ICCV '15. Washington, DC, USA: IEEE Computer Society; 2015. p. 4346–4354.
[13] Li Z, Zhou Y, Xiao S, He C, Li H. Auto-Conditioned LSTM Network for Extended Complex Human Motion Synthesis. CoRR. 2017;abs/1707.05363.
[14] Lee K, Lee S, Lee J. Interactive Character Animation by Learning Multi-objective Control. ACM Trans Graph. 2018 Dec;37(6):180:1–180:10.
[15] Gaisbauer F, Fröhlich P, Lehwald J, Agethen P, Rukzio E. Presenting a Deep Motion Blending Approach for Simulating Natural Reach Motions. In: Proceedings of the 39th Annual European Association for Computer Graphics Conference: Posters. EG '18. Goslar Germany, Germany: Eurographics Association; 2018. p. 5–6.
[16] Brand M, Hertzmann A. Style Machines. In: Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques. SIGGRAPH '00. New York, NY, USA: ACM Press/Addison-Wesley Publishing Co.; 2000. p. 183–192.
[17] Bowden R. Learning statistical models of human motion. In: IEEE Workshop on Human Modeling, Analysis and Synthesis, CVPR. vol. 2000; 2000. .
[18] Lehrmann AM, Gehler PV, Nowozin S. Efficient nonlinear markov models for human motion. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition; 2014. p. 1314–1321.
[19] Min J, Chen YL, Chai J. Interactive generation of human animation with deformable motion models. ACM Transactions on Graphics (TOG). 2009;29(1):9.
[20] Du H, Hosseini S, Manns M, Herrmann E, Fischer K. Scaled Functional Principal Component Analysis for Human Motion Synthesis. In: Proceedings of the 9th International Conference on Motion in Games. MIG '16. New York, NY, USA: ACM; 2016. p. 139–144.
[21] Lawrence ND, Moore AJ. Hierarchical Gaussian process latent variable models. In: Proceedings of the 24th international conference on Machine learning. ACM; 2007. p. 481–488.
[22] Wang JM, Fleet DJ, Hertzmann A. Gaussian Process Dynamical Models for Human Motion. IEEE Transactions on Pattern Analysis and Machine Intelligence. 2008 Feb;30(2):283–298.
[23] Zhou L, Shang L, Shum HP, Leung H. Human motion variation synthesis with multivariate Gaussian processes. Computer Animation and Virtual Worlds. 2014;25(3-4):301–309.
[24] Holden D, Saito J, Komura T, Joyce T. Learning motion manifolds with convolutional autoencoders. In: SIGGRAPH Asia 2015 Technical Briefs. ACM; 2015. p. 18.
[25] Clevert D, Unterthiner T, Hochreiter S. Fast and Accurate Deep Network Learning by Exponential Linear Units (ELUs). CoRR. 2015;abs/1511.07289.
[26] Clavet S. Motion matching and the road to next-gen animation. In: Proc. of GDC; 2016. .

[27] Abadi M, Barham P, Chen J, Chen Z, Davis A, Dean J, et al. TensorFlow: A System for Large-scale Machine Learning. In: Proceedings of the 12th USENIX Conference on Operating Systems Design and Implementation. OSDI '16. Berkeley, CA, USA:
USENIX Association; 2016. p. 265–283.

[28] Kingma DP, Ba J. Adam: A Method for Stochastic Optimization. CoRR. 2014;abs/1412.6980.

[29] Srivastava N, Hinton G, Krizhevsky A, Sutskever I, Salakhutdinov R. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. Journal of Machine Learning Research. 2014;15:1929–1958.

A. Appendix

A.1. Definition of a Cubic Catmull-Rom Spline

$$\begin{aligned}\Theta(p;\theta) &= \alpha_{k_1} + w(\frac{1}{2}\alpha_{k_2} - \frac{1}{2}\alpha_{k_0}) + w^2(\alpha_{k_0} - \frac{5}{2}\alpha_{k_1} + 2\alpha_{k_2} - \frac{1}{2}\alpha_{k_3}) \\ &\quad + w^3(\frac{3}{2}\alpha_{k_1} - \frac{3}{2}\alpha_{k_2} + \frac{1}{2}\alpha_{k_3} - \frac{1}{2}\alpha_{k_0}) \\ w &= \frac{4p}{2\pi} \mod 1 \\ k_n &= (\left\lfloor \frac{4p}{2\pi} \right\rfloor + n - 1) \mod 4\end{aligned} \tag{9}$$