Natural Posture Blending Using Deep Neural Networks

Felix Gaisbauer
Daimler AG
Ulm, Germany

Jannes Lehwald
Daimler AG
Ulm, Germany

Janis Sprenger
German Research Center for Artificial Intelligence (DFKI)
Saarbrücken, Germany

Enrico Rukzio
Ulm University
Ulm, Germany

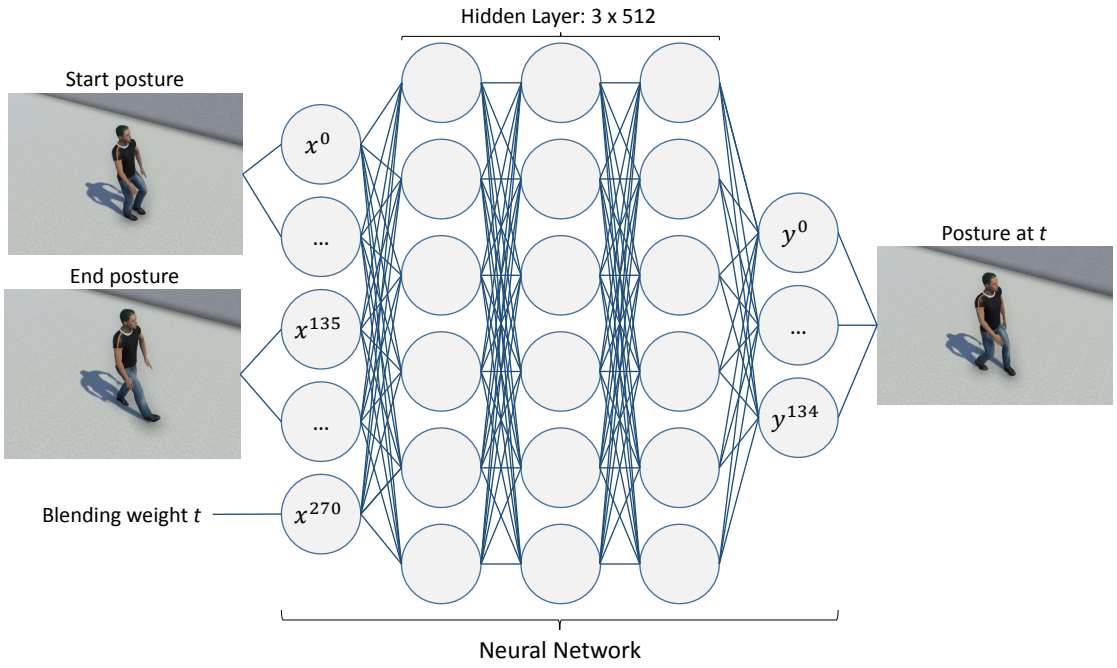

Figure 1: Visual representation of the neural network structure. Given a humanoid start and end posture, the network generates a blended posture according to the blending weight t.

ABSTRACT

Motion synthesis approaches are widely used throughout different domains such as gaming, virtual crowds or simulation within production industries. With ongoing digitization, these systems are becoming increasingly indispensable. In general, the utilized technologies can be subdivided in data-driven and model-based approaches, whereas each category has its advantages and disadvantages. In the field of data-driven motion synthesis, recent works present deep learning based approaches for full body motion synthesis, which offer great potential for modeling natural motions, while considering heterogeneous influence factors. In this paper, we propose a novel deep blending approach for blending collision-free and feasible postures between a humanoid start and target posture. The network has been trained utilizing the CMU database to generate feasible postures. The proposed approach can be utilized for posture-blending, motion synthesis with known start and end-posture or key-frame animation. A preliminary evaluation indicates the validity and the potential of the novel approach.

MIG '19, October 28–30, 2019, Newcastle upon Tyne, United Kingdom

ACM ISBN 978-1-4503-6994-7/19/10…$15.00
https://doi.org/10.1145/3359566.3360052

CCS CONCEPTS

• **Computing methodologies → Animation**; *Model development and analysis*; *Motion capture.*

KEYWORDS

Posture Blending, Neural Network, Motion Synthesis, Animation, Key-frame Animation, Deep Learning, In-Betweening

ACM Reference Format:
Felix Gaisbauer, Jannes Lehwald, Janis Sprenger, and Enrico Rukzio. 2019. Natural Posture Blending Using Deep Neural Networks. In *Motion, Interaction and Games (MIG '19), October 28–30, 2019, Newcastle upon Tyne, United Kingdom.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3359566.3360052

1 INTRODUCTION

Nowadays, motion synthesis systems are an essential aspect of many domains such as the entertainment or production industries. While motion blending is a commonly used technique for motion synthesis in the industry, there are many recent publications using data-driven, model-driven or physics-driven approaches. Due to increasing computational capabilities, deep learning approaches receive growing attention. Recent approaches for character animation, such as [Holden et al. 2017; Zhou et al. 2018b], are powerful alternatives to common motion blending systems and can consider heterogeneous constraints. These systems, however, fail to reach specific full body poses at the end of an animation. In addition, they require a large amount of training data for each action. In the case of highly specific and tailored movements, manual editing and motion generation using key-frame animations is still the dominant approach. In order to combine these traditional motion clips with novel data-driven approaches for character animation, motion blending is required. Simple linear blending, however, is not sufficient for all movements and can generate many artifacts (e.g. pose-averaging, foot-sliding, etc.). Optimizing the blending functions is time-intensive and not always feasible.

In this paper, we present a deep neural network (DNN) addressing motion blending. It synthesizes human postures based on a given start and end posture as well as an interpolation weight. The network was trained using the CMU motion capture database [cmu 2019] comprising heterogeneous motions of different subjects. The proposed approach can be used for different application scenarios. Beside the blending of feasible postures, the novel approach can be utilized to improve key-frame animation and reduce the amount of manually specified postures. Moreover, the novel approach can be applied for motion synthesis in scenarios in which the desired end posture is already known. Overall, the proposed approach can be considered as a powerful and flexible alternative to common posture blending approaches based on interpolation.

2 RELATED WORK

The proposed approach in this paper relies on a base of different technologies. Therefore, the state of the art regarding motion-, posture blending, deep learning-based motion synthesis and in-betweening is revisited.

2.1 Motion Blending & Posture Interpolation

Motion blending techniques are used throughout different domains and can produce natural-looking results with low computational efforts. In general, these interpolation approaches can be subdivided into Barycentric-, K-Nearest-Neighbor- and Radial Basis Function-interpolation [Feng et al. 2012]. Moreover, inverse blending [Huang and Kallmann 2010] and inverse kinematics are frequently utilized for solving various constraints. There is, however, no single blending techniques suited for every task [Feng et al. 2012]. In general, it can be denoted that with an increasing amount of constraints these systems tend to fail or require a vast amount of time to satisfy all desired constraints.

In contrast to motion blending, posture blending approaches do not interpolate entire motions, instead only individual postures are generated based on given input postures. The simplest way to perform a posture blend is to interpolate between the joint transformations of different postures. However, linear or spline-based interpolation only produces feasible results if the temporal and spatial distance between the two postures is small. For instance, if the input of the posture blend are two walk postures being in different phases of the walk-cycle, foot sliding might occur, whereas the walk-cycle might be entirely neglected in the blended posture. Moreover, depending on the given input postures, self-collisions might occur in the resulting posture. The authors of [Badler et al. 1994] presented a posture blending approach with collision avoidance utilizing a finite state machine, where the states define the overall primitives of postures like standing or squatting. Given the approach, however, these states as well as the transitions in between must be manually defined. More sophisticated approaches such as [Wang et al. 2014] utilize motion planning algorithms to generate feasible and collision free postures. These approaches allow covering large distances between consecutive postures, however at the expense of computational time. With our novel approach, we want to reduce the manual efforts, whereas an interpolation between heterogeneous postures is provided in a lightweight manner.

2.2 Deep Learning based Motion Synthesis

Due to the growing computational capabilities and ongoing development of software for machine learning, recently, deep learning approaches received significant attention in various domains, including character animation. The authors of [Holden et al. 2017] proposed an approach to synthesize human walk motions based on a phase-functioned neural network which has been trained using unstructured motion capture data. Each walk phase is represented by a distinct network, whereas each network models the transition to the next phase utilizing given control input and the current posture. In [Gaisbauer et al. 2018] the approach has been extended to synthesize natural reach motions given positional and rotational constraints. Moreover, other approaches such as [Zhou et al. 2018b] synthesize natural humanoid motions based on recurrent neural networks or utilize convolutional auto-encoders [Holden et al. 2015]. The presented approach within the paper builds upon the recent publications in this field and utilizes a feed-forward neural network to synthesize body postures.

2.3 In-Betweening

Despite the previously addressed domains, posture interpolation is also related to in-betweening. Similar to the data-driven motion synthesis, recent publication focus on the use of artificial intelligence for addressing the problem of generating data in between frames. Recent works present an auto-completion approach based on autoregressive two-layer recurrent neural network [Zhang and van de Panne 2018], whereas the validity is approved based on a hopping lamp animation stemming from a physics-based model. [Yagi 2017] present a filter based approach for in-betweening in which a convolutional neural network is used to generate intermediate frames and smooth animation of line drawing. Moreover, [Li et al. 2019] propose a novel approach for video in-betweening based on 3D convolutions. The recent progress in this domain using artificial neural networks unveils the possibilities of applying these techniques for other problems as well. Therewith, we consider DNNs as a promising approach for the interpolation of humanoid postures.

3 POSTURE BLENDING NETWORK

The basic idea of the proposed approach is to generate natural human postures blended between a start and end posture based on a deep neural network. Whereas traditional posture blending might fail to produce feasible transitions between two postures without further knowledge, the developed DNN has learned various factors of motions such as velocity and motion phase. Therefore, the network is capable of reproducing plausible postures which do not contain self-collision and unnatural postures.

3.1 Input & Output Variables

The input and output variables are crucial parameters for neural networks. As stated before, only the start h_0 and end posture h_1 together with the blending weight t are required to blend between two human postures. Postures are defined by the configuration of a kinematic skeleton of $j = 22$ joints in this work. The root position p_0 and rotation r_0 are defined in the global Cartesian space, whereas the joint rotations r_i are defined in the local space w.r.t. the root joint. The root position is defined as a vector in Cartesian space and rotations $r_i \in \mathbb{R}^6$ are described by the forward and up direction vector of the joint i after rotation. This specific representation has been selected to define rotations in the same space as positions and proved to produce better results compared to Euler angles or quaternions in our experiments. Our initial experiments are thus confirming a recent finding by [Zhou et al. 2018a], who analyzed different rotation representations for neural networks in a quantitative evaluation. One posture is the combination of the global position and rotation of the root joint and the local rotations of all 21 subsequent joints in the skeleton of the utilized avatar. In total, all rotations and positions sum up to one posture vector $h_i \in \mathbb{R}^{135}$. Therefore, the full input vector consists of $X = \{h_0, h_1, t\} \in \mathbb{R}^{271}$ with $t \in \mathbb{R}$ being the blending weight. The output vector y of the system is the human posture $Y = h_t \in \mathbb{R}^{135}$. Figure 1 schematically describes these parameters.

3.2 Data Acquisition & Preprocessing

In order to cover a high variety of motions, the complete CMU motion capture database was used for the generation of input and output data. Overall, this database contains 2605 motion files in 6 categories.

Before processing, the animations were re-targeted to the Unity Mecanim skeleton (finger joints excluded). Hereafter, the individual motions were randomly separated to 80% training, 10% validation and 10% test sets. In the next step, the motions were cut into clips with random length between 0.1 and 1.0 seconds. Longer motions sequences were considered as to inconsistent and diverse for the network to learn. Afterwards the segmented motions were normalized w.r.t. the initial frame. The clip was first translated such that the projection of the root joint on the ground plane started at zero. Afterwards the whole clip was rotated such that the forward direction of the root joint pointed towards the global Z-axis.

Afterwards, the pre-processed clips were split into input- and output-pairs for each frame. The start (h_0) and end posture (h_1) were defined by the start and end posture of the clip. For each frame, the blending weight t was defined by elapsed frame time proportional to the whole duration of the clip. The output posture h_t was then defined by the specific posture at this frame. Hence, for a motion clip with a length of 30 frames, 30 pairs are generated. At the end, the input and output vectors were rescaled to a mean of 0.0 and a standard deviation of 1.0.

3.3 Neural Network Structure

We propose a fully-connected feed-forward neural network for posture blending. The input vector x and output vector y are described in 3.1. Given an input vector, the network generates an output vector as follows:

$$\Phi(x;\alpha) = W_3\ SELU(W_2\ SELU(W_1\ SELU(W_0 x + b_0) + b_1) + b_2) + b_3 \tag{1}$$

The network α is defined by its weights and biases and are defined by

$$\begin{aligned}\alpha = \{&W_0 \in \mathbb{R}^{h\times n}; W_1 \in \mathbb{R}^{h\times h}; W_2 \in \mathbb{R}^{h\times h}; W_3 \in \mathbb{R}^{m\times h};\\ &b_0 \in \mathbb{R}^h; b_1 \in \mathbb{R}^h; b_2 \in \mathbb{R}^h; b_3 \in \mathbb{R}^m\}\end{aligned} \tag{2}$$

Here $n = 271$ and $m = 135$ describe the dimensionality of the input and output vector as described in section 3.1. The numbers of units used in the hidden layer is described by $h = 512$. Scaled exponential linear units (SELU) [Klambauer et al. 2017] were used as an activation function. For network training, we utilized the Adam algorithm [Kingma and Ba 2014] with a learning rate of 0.001.

The hyperparameters were found by an optimization explained in section 3.4. A visual representation of the utilized feed-forward network structure is shown in figure 1. Since the network is a compact representation of learned motions, all training motions fit into the total disk space of the weights which is around 3 MB.

3.4 Hyperparameter optimization

A genetic algorithm as described in [Man et al. 1996] was used to find the optimal hyperparameter set $z = \{n_h, n_n, a_h, a_o, o, l\}$

for the network, where $n_h = \{0, 1, 2, 3, 4, 5\}$ is the number of hidden layers, $n_n = \{32, 64, 128, 256, 512, 1024\}$ the number of neurons in the hidden layers, $a_h, a_o = \{Linear, Sigmoid, tanh, ELU, LeakyReLU, SELU\}$ the activation function of the hidden and output layer respectively, $o = \{Adam, SGD, RMSProp, Adagrad, Adadelta\}$ is the optimizer and $l = \{0.01, 0.001, 0.0001\}$ its learning rate. Both, the genetic algorithm as well as the neural networks, were implemented in python whereas keras 2.2.4 with tensorflow backend (gpu) was utilized for the neural network.

The population size of 12 was optimized over 8 generations. The fitness of the trained networks were defined by their performance against the generated validation data described in section 3.2. The training process for each neural network was composed of two subsequent phases with different batch sizes. In the initial training step a mini-batch size of 32 was utilized whereas in the second phase the size was raised to 30,000 to fine-tune the network parameters. To ensure optimal trained networks the early-stopping strategy was used in both cases. Additionally, a checkpoint of each model was created once and overwritten after each epoch, if the validation loss had decreased. Therefore, the models with the best validation losses were utilized for the performance evaluation. Furthermore, batch normalization between each hidden layer was utilized to improve the overall training speed. As loss function the mean squared error was used. The training of one network took around five hours on a NVIDIA Quadro M6000, which resulted in a total training time of around 960 hours or 20 days for all generations.

3.5 Post Processing

The postures generated by the model are approximations of the real motion. Moreover, the training data might contain multiple motions with similar start and end postures, whereas the specific motion in between varies. Those ambiguities result in over-smoothed and averaged results produced by the network. Therefore, the output near to the start and end postures are not calculated exact and small jumps between the blended and real pose are inevitable. To remove this inconsistency, further post processing steps are performed.

First, the posture h_t is calculated by the neural network with the weight t. Additionally, a posture h_i through linear blending with the same weight t between the start and end posture is determined. Afterwards, an additional linear blending from h_t to h_i with the weight $f = (2t - 1)^4$ is performed and the generated posture h_f is used as the final posture. As a function for determining the blending weight different polynomials have been considered. However, the selected polynomial ensures that the start/end postures are matched, whereas the output of the neural network is not manipulated too much (natural poses are expected to be similar if getting close to start /end). The function of f ensures that the start and end posture are matched perfectly, since its value is 1 and any sudden jumps near it is removed with the linear blending. Without the applied post-processing, we encountered occasionally sudden jumps between start/end postures.

Since no environment information is added during blending, the collision of the feet in uneven terrain is a common problem, especially if the input postures require a walking motion in between. This problem is avoided, by applying foot grounding to the produced posture h_f.

Figure 2: Illustration of the postures generated by the novel approach at different blending weights t. The start and end postures stem from a forward walking (1st row), backward ball dribbling (2nd row) and a knee-to-elbow motion (3rd row).

4 EVALUATION

The validity of the network was evaluated based on the test data, as described in 3.2. The used data comprises in total around 600,000 entries. The total mean squared error over all test motions was $5.4 * 10^{-3}$ and the generation of one blended posture needed $5.4 * 10^{-4}$ seconds on average. However, since the quantified values give only limited impression of the quality of the generated results, a further evaluation was carried out.

4.1 Validation of Example Motions

To examine the generated results of the novel approach, three different motions from the test set have been utilized for visualization. Random motion clips from the whole motions with the duration of 1.0s were chosen since these represent the outer border of the trained data sets. Figure 2 visualizes the resulting postures at weights $t = \{0.0, 0.25, 0.5, 0.75, 1.0\}$ after post processing was applied.

The first postures (Figure 2, 1st row) originate from a forward walking motion clip during a right gait cycle. From a start posture which shows the initial contact of the right foot with the ground, the motion ends with the toe off posture of the right foot. The blending model produces postures in between, where the left foot is raised according to its walking phase. The swing motions of both arms are modeled and fit to the walking state. In the second case, a backwards ball dribbling clip was used to extract the start and end posture (Figure 2, 2nd row). Similar to the first motion, the

backward walking cycle is well represented by the network. The ball dribbling left hand moves slightly up and down like in a real dribbling movement. The last postures represent a knee-to-elbow motion (Figure 2, 3rd row). It can be seen that the network has learned to first set the right foot to the ground before the left foot can be lifted into the air to perform the movement.

In comparison, generated motions through linear blending include unnatural looking foot sliding for the first two and a hovering human in the third motion. Therefore, for the examined scenarios, it can be concluded that the novel network produces more natural results compared to linear blending. For the comparison, further blending approaches were not considered since the only information that is available are the start and end posture. For spline based interpolation approaches (e.g., Catmull Rom, Bézier) further keyframes such as provided by preceding or subsequent frames would be required.

4.2 Self-Collisions

A common problem of applying linear blending to interpolate between two postures are self-collisions. For instance, if a start posture comprises an arm in front of a person and the arm is positioned side-wards to the body in the target posture, a straight interpolation would result in a trajectory crossing the own body and induce a self-collision. Given the novel deep learning based blending approach, it is expected that the occurrence of self-collisions can be reduced, since the network has learned the characteristics of the underlying datasets. Therefore, besides the previously examined motions, the generated postures by the novel approach are further compared to linear blending in terms of self-collisions. In this context, 921 motions of the test data set were used to validate whether the amount of self-collisions can be reduced by applying the neural network, compared to linear posture blending. A humanoid rig consisting of 15 box-colliders was set up to test for self-collisions. As illustrated in Figure 3, the arms and legs were each modeled by three box-colliders. Moreover, the torso is represented by two box colliders, whereas the head is additionally approximated by a box-collider. A self-collision is registered if two box-colliders intersect with each other. The overlapping is determined using the Physics.OverlapBox functionality of the Unity Engine.

In particular, collisions between colliders in the same segment (e.g. arm, legs or torso) were ignored. The amount of self-collisions was examined for different blending durations, to determine the overall capability of the novel approach. In this context, ten different blending times ranging from 0.1 s to 1.0s were examined. Figure 4

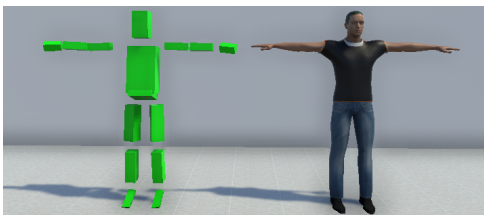

Figure 3: Illustration of the utilized approximation for determining self-collisions.

visualize the gathered results of the evaluation. Each bar depicts the relative amount of self-collisions r_s compared to linear blending. Values below 100% denote less self-collisions for the novel approach compared to linear blending. The ground-truth data stemming from the CMU library contains occasionally self-collisions. Therewith, test-data containing self-collisions was excluded from the analysis.

Examining the analyzed motions of the given test data, it can be encountered that the novel approach produces less self-collisions in every tested scenario. In trend, the amount of self-collisions is reduced with growing blending times, leading to a minimum of $r_s = 51.75\%$ at a blending duration of 1.0s. Overall, aggregating all ten results, a mean value of 76.13%, median of 81.80% and a standard deviation of 10.94% can be denoted. For each tested scenario, in average 249.70 self-collisions occurred using linear blending, whereas the amount can be reduced to 190.70 using the novel posture blending approach. Summarizing the results, it can be concluded that the novel approach outperforms the linear blending in terms of self-collision reduction. The network has learned the characteristics of the natural training-data and thus produces less self-collisions.

4.3 End-Effector Accuracy

In some cases, the joint positions of end-effectors are more important than the actual posture. We compared the positional accuracy of the hand and feet joints between our method and linear blending on a per-frame basis for different interpolation steps. Using the test dataset, we performed interpolation in 0.1 steps with both methods and compared the positions of end-effectors in Cartesian space with the ground truth. The positional error of our approach was subtracted from the positional error of the linear blending. Thus positive values denote a higher positional error for the linear blending, negative values a higher positional error for our method. The differential error was summed over all end-effectors for the interpolated frame. We computed the mean and standard deviation of these differential end-effector errors for all frames and present them in figure 5. As all differential errors are within two standard deviations, our approach does not perform significantly different from the linear posture blending, but with comparable accuracy.

5 LIMITATIONS

Even though the functionality of the novel approach could be illustrated, there are still several limitations. The whole CMU database contains heterogeneous actions with different coverage. As all of these actions have been used in an unstructured way, our method produces undesired behavior (e.g. fast hand-movements between two close postures) and does not solve foot sliding in some cases. We expect the usage of prior knowledge of actions to further improve the performance of our method, as well as with respect on the positional accuracy of end-effectors. In particular, the suitability of the network for specific motion types (e.g. grasping, walking) must be examined in more detail. Ignoring further parameters like motion type or style, the training data can contain ambiguous motions being represented by the same start/end posture. As a result, the neural network might average the motions, which results in over-smoothed results that can be encountered in the test scenarios.

Furthermore, the duration of motions is not explicitly annotated. Instead, only the static start and end posture are used as input.

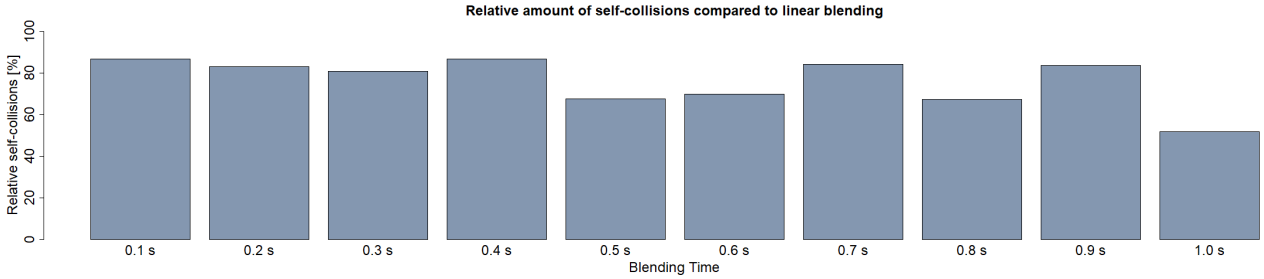

Figure 4: Overview of the relative amount of self-collisions of the novel approach compared to linear blending.

Hence, temporal parameters with impact to the acceleration and velocity of the motion can not be externally specified. On the other hand, no velocity information are contained in the output. For instance if two identical key-frames with different duration are inserted, in every case the identical results are generated. Considering the temporal relation of motion clips including acceleration and velocity will most likely improve the performance of our model and its applicability.

Currently, the network generates a full posture as output. A further measure which might improve the overall results is to utilize the residual of the blended postures. In this context, it is expected that the positional error compared to linear blending can be reduced.

Finally, the evaluation indicated that the identification of a suitable metric for a comparison of the generated motions to the ground truth data is difficult. To overcome these difficulties, Generative Adversarial Networks (GANs) could be utilized in future work. The network (generator) could create postures, whereas the discriminator intrinsically learns the characteristics of feasible postures.

6 CONCLUSION

In this paper, we presented a novel approach to generate feasible humanoid postures from given input postures and a blending weight. The proposed approach incorporates unstructured data of the CMU library within a neural network and can generate intermediate postures which preserve the characteristics of the trained input motions. The evaluation indicates that our approach can generate postures with fewer self-collisions while retaining the positional accuracy of linear interpolation.

Figure 5: Mean differential error of end-effectors between our approach and linear posture blending for different blending steps. Error bars denote one standard deviation.

ACKNOWLEDGMENTS

The authors acknowledge the financial support by the Federal Ministry of Education and Research of Germany within the MOSIM research project (grant number 01IS18060A-H).

REFERENCES

Norman I Badler, Ramamani Bindiganavale, John P Granieri, Susanna Wei, and Xinmin Zhao. 1994. Posture interpolation with collision avoidance. *Center for Human Modeling and Simulation* (1994), 58.

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

Andrew Feng, Yazhou Huang, Marcelo Kallmann, and Ari Shapiro. 2012. An analysis of motion blending techniques. In *International Conference on Motion in Games*. Springer, 232–243.

Felix Gaisbauer, Philipp Froehlich, Jannes Lehwald, Philipp Agethen, and Enrico Rukzio. 2018. Presenting a Deep Motion Blending Approach for Simulating Natural Reach Motions. In *EG 2018 - Posters*, Eakta Jain and Jirí Kosinka (Eds.). The Eurographics Association.

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

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

Yazhou Huang and Marcelo Kallmann. 2010. Motion parameterization with inverse blending. In *International Conference on Motion in Games*. Springer, 242–253.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014). https://arxiv.org/abs/1412.6980

Günter Klambauer, Thomas Unterthiner, Andreas Mayr, and Sepp Hochreiter. 2017. Self-Normalizing Neural Networks. *CoRR* abs/1706.02515 (2017). arXiv:1706.02515

Yunpeng Li, Dominik Roblek, and Marco Tagliasacchi. 2019. From Here to There: Video Inbetweening Using Direct 3D Convolutions. *arXiv preprint arXiv:1905.10240* (2019).

K. F. Man, K. S. Tang, and S. Kwong. 1996. Genetic algorithms: concepts and applications [in engineering design]. *IEEE Transactions on Industrial Electronics* 43, 5 (Oct 1996), 519–534.

He Wang, Edmond Ho, and Taku Komura. 2014. An Energy-Driven Motion Planning Method for Two Distant Postures. *IEEE Transactions on Visualization and Computer Graphics* (06 2014).

Yuichi Yagi. 2017. A filter based approach for inbetweening. *ArXiv* abs/1706.03497 (2017).

Xinyi Zhang and Michiel van de Panne. 2018. Data-driven Autocompletion for Keyframe Animation. In *Proceedings of the 11th Annual International Conference on Motion, Interaction, and Games (MIG '18)*. ACM, New York, NY, USA, Article 10, 11 pages. https://doi.org/10.1145/3274247.3274502

Yi Zhou, Connelly Barnes, Jingwan Lu, Jimei Yang, and Hao Li. 2018a. On the Continuity of Rotation Representations in Neural Networks. *arXiv preprint arXiv:1812.07035* (2018).

Yi Zhou, Zimo Li, Shuangjiu Xiao, Chong He, Zeng Huang, and Hao Li. 2018b. Auto-Conditioned Recurrent Networks for Extended Complex Human Motion Synthesis. In *International Conference on Learning Representations*. https://openreview.net/forum?id=r11Q2SlRW