The final publication is available at IEEE Xplore via `http://doi.org/10.1109/VRW52623.2021.00172`.

Lightweight Quaternion Transition Generation with Neural Networks

Romi Geleijn * Adrian Radziszewski † Julia Beryl van Straaten ‡ Henrique Galvan Debarba§

IT University of Copenhagen, Copenhagen, Denmark

ABSTRACT

This paper introduces the Quaternion Transition Generator (QTG), a new network architecture tailored to animation transition generation for virtual characters. The QTG is simpler than the current state of the art, making it lightweight and easier to implement. It uses approximately 80% fewer arithmetic operations compared to other transition networks. Additionally, this architecture is capable of generating visually accurate rotation-based animations transitions and results in a lower Mean Absolute Error than transition generation techniques that are commonly used for animation blending.

Index Terms: Computing methodologies—Computer graphics—Animation—Procedural animation;

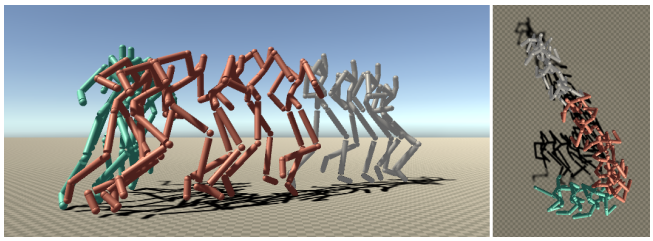
1 INTRODUCTION

Transitions are short animated sequences that connect two animations or keyframes in virtual characters. For instance, when a character transitions from a running state to a roll, the animation database may not contain the animation frames for this intermediate action connecting both states. Modern virtual characters require a large amount of animations, and each new state or animation added to the character increases the needed amount of transition animations exponentially [3]. These animation transitions are often generated using techniques such as linear blending and inertialisation [1]. However, these techniques frequently have unsatisfactory results, requiring the work of a professional animator to manually edit and fine tune the animation transition parameters. Therefore, the task of setting up such transitions and manually editing them becomes an immensely time-consuming and expensive task.

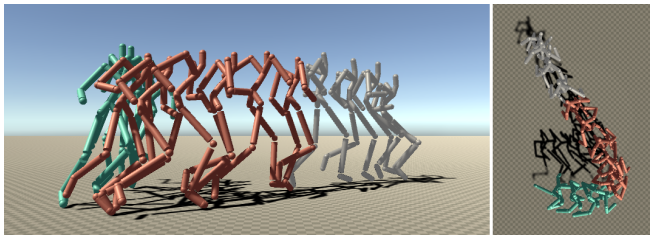
Here, we explore the use of neural networks in the task of creating animation transitions. Harvey & Pal [3] were the first to demonstrate their effectiveness, applying an adaptation of a Recurrent Neural Network (RNN), which they named Recurrent Transition Network (RTN), to the issue. Harvey & Pal built RTNs upon the ideas of Encoder-Recurrent-Decoder (ERD) networks, from Fragkiadaki et al [2], and residual temporal networks for modeling human body dynamics, from Martinez et al [6]. Moreover, they also extended ERD networks with future-context conditioning, by adding a *target vector* to indicate the frame that the network should attempt to predict by the end of a transition.

In summary, the RTN uses auto-encoders to encode the past context, the current context and the target state of the transition. Then, a recurrent generator consisting of a single Long Short-Term Memory (LSTM) layer is applied to the encoded input. Finally, the output of the recurrent Generator is decoded, resulting in an offset that is added to the previous frame to obtain the current frame. However, although the RTN is capable of generating natural looking motion, it lacks the capability to predict joint rotations, which are needed to completely represent the skeleton of a skinned character.

*e-mail: romi.geleijn@hotmail.com
†e-mail: adrad@outlook.dk
‡e-mail: juliavanstraaten@gmail.com
§e-mail: hend@itu.dk



(a) Generated transition



(b) Ground truth transition

Figure 1: In red, a transition generated with the proposed method (a) compared to the ground truth motion data (b). Images were sub-sampled to show a still every 5 frames.

This paper introduces a new network architecture tailored to transition generation. The network has a simpler architecture and a smaller footprint than the RTN, making it lightweight and easier to implement and, additionally, is capable of generating rotation-based animations transitions.

2 IMPLEMENTATION

The motion capture data set from Holden et al. [5] is used for training the network. It contains long animation sequences with transitions between various locomotion states and is re-targeted to a uniform skeleton. A subset of 22 joints from the skeleton is used, and the animation sequences are downsampled to 30 frames per second. Each animation is divided into smaller, overlapping 1 second sequences, where the root joint transformation of each pose is relative to the first pose in the sequence. Four additional frames are stored with each transition, the two poses immediately before the transition (the past context), and the two poses immediately after (the future context, or the target state). Joints are kept in local space, and each joint is represented by a local rotation quaternion, and additionally a translation in the case of the root joint. Thus, each sequence is represented by a vector of $T+4$ poses, with T being the transition length, and each pose consisting of $P+Q*22$ values representing the root joint translation ($P=3$) and the local rotation quaternion ($Q=4$) of the 22 joints. Additionally, the root translation values are standardised using z-normalisation on each dimension.

The network architecture, shown in figure 2, is based on the ERD architecture [2], but with additional operations performed in the encoded space. All dense layers in the auto-encoder use the Leaky Rectified Linear Unit activation function, except for the output layer with a linear activation. The network takes 2 past poses and 2 target poses as input and encodes them into a latent space representations,

The final publication is available at IEEE Xplore via `http://doi.org/10.1109/VRW52623.2021.00172`.

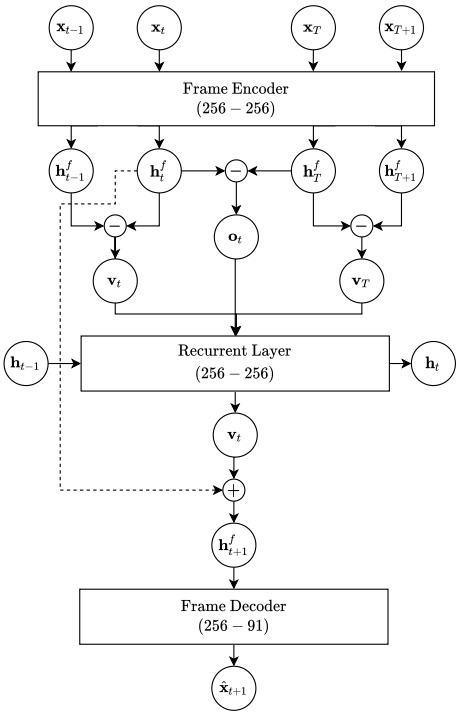


Figure 2: QTG architecture for a single time step, with layer sizes in parentheses.

which are used to derive the latent velocities and the offset from the target pose. The calculated latent information is then processed by a recurrent layer, composed of a GRU unit and a dense layer, into the current latent space velocities. The velocities are added to the last pose $\mathbf{h}^f_t$ through a residual connection, resulting in the next latent pose $\mathbf{h}^f_{t+1}$, which is decoded into the next pose, and fed back to the network to generate the next transition frame.

The network was tested on 1 second transitions (30 frames). It was trained using minibatches of size 32 for 1000 epochs, using the Mean Absolute Error loss, an Adam optimizer with beta values $\beta_1 = 0.5$ and $\beta_2 = 0.9$, and a learning rate set to 0.0001.

3 Results

In order to evaluate the QTG quantitatively, the Mean Absolute Error over time was calculated. The results, together with the results of two baseline methods, can be found in Fig. 3. When compared to the ground truth animation, the error of the QTG was several times lower than that of the two baselines, staying below 5 centimeters in the average per joint. Yet, it should be noted that the noisy error in the first few frames (0-100ms) of the transition can cause noticeable jitter to some transition animations.

We also present a visual comparison of the predicted transitions by comparing the individual frames of the predicted animation to the ground truth. Fig. 1b contains an example transition, in which the character turns right whilst running. The QTG predictions are displayed in Fig. 1a. The past context is shown in green, whilst the future context is shown in white; the images were sub-sampled and only show a still every 5 frames.When comparing the animations, there is a slight deviation in the movement of the left and right arms of the character at the middle of the transition animation. Both arms extend more in the QTG, in comparison to the ground truth animation. This is especially visible in the top-down view of the

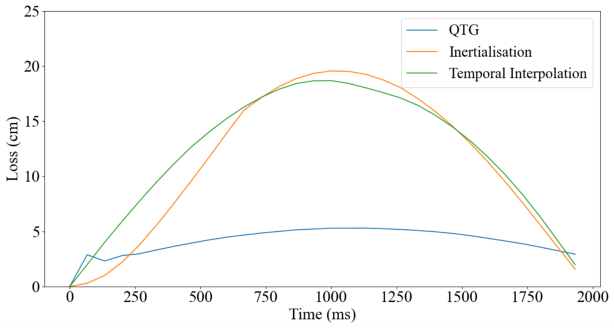


Figure 3: The Mean Absolute Error per joint per frame of the QTG network, Inertialisation and Temporal Interpolation.

movement. However, the generated frames are visually similar to the ground truth. The movement of the feet, as well as the root position and orientation of the character appear to be particularly similar.

Compared to the RTN, the proposed network performs $\approx 80\%$ fewer arithmetic operations due to its simpler structure. The QTG achieves a prediction time of 0.33ms to generate a 1 second transitions (30 frames), as tested on a Core i7-8700 CPU.

4 Conclusion

We presented the QTG, a lightweight neural network that can efficiently generate transition animations for virtual characters. The QTG requires approximately 80% fewer arithmetic operations than the RTN, but still generates visually accurate transitions. Additionally, the QTG generates rotation-based animation transitions, and results in a much lower error than transition generation techniques relying on pose blending.

In the future, we would like to compare our results with the RTN, and to investigate how QTG performs on a wider range of character animation transitions. Moreover, Harvey et al. [4] have recently expanded their research with an altered network architecture. However, their focus is no longer on generating real-time animation transitions, but on in-betweening for animation creation. Still, a comparison between [4] and the QTN would be interesting.

Acknowledgments

RG, AR and JBS contributed equally to this research.

References

[1] D. Bollo. Inertialization: high-performance animation transitions in Gears of War. `https://www.gdcvault.com/play/1025331/Inertialization-High-Performance-Animation-Transitions`, 2018. [Online; accessed 12-February-2021].

[2] K. Fragkiadaki, S. Levine, P. Felsen, and J. Malik. Recurrent network models for human dynamics. In *2015 IEEE International Conference on Computer Vision (ICCV)*, pp. 4346–4354, 2015. doi: 10.1109/ICCV.2015.494

[3] F. G. Harvey and C. Pal. Recurrent transition networks for character locomotion. In *SIGGRAPH Asia 2018 Technical Briefs*, 2018. doi: 10.1145/3283254.3283277

[4] F. G. Harvey, M. Yurick, D. Nowrouzezahrai, and C. Pal. Robust motion in-betweening. *ACM Trans. Graph.*, 39(4), July 2020. doi: 10.1145/3386569.3392480

[5] D. Holden, T. Komura, and J. Saito. Phase-functioned neural networks for character control. *ACM Trans. Graph.*, 36(4), July 2017. doi: 10.1145/3072959.3073663

[6] J. Martinez, M. J. Black, and J. Romero. On human motion prediction using recurrent neural networks. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 4674–4683, 2017. doi: 10.1109/CVPR.2017.497