



Transformer Based Motion In-Betweening

Pavithra Sridhar^(✉), V. Aananth, Madhav Aggarwal, and R. Leela Velusamy

National Institute of Technology - Tiruchirappalli, Tiruchirappalli 620015, TN, India
pavisri99@gmail.com, leela@nitt.edu

Abstract. In-betweening is the process of drawing transition frames between temporally-sparse keyframes to create a smooth animation sequence. This work presents a novel transformer-based in-betweening technique that serves as a tool for 3D animators. We first show that this problem can be represented as a sequence-to-sequence problem and introduce Tween Transformers - a model that synthesizes high-quality animations using temporally-sparse keyframes as input constraints.

We evaluate the model's performance via two complementary methods - quantitative and qualitative evaluation. The model is compared quantitatively with the state-of-the-art models using LaFAN1, a high-quality animation dataset. Mean-squared metrics like L2P, L2Q, and NPSS are used for evaluation. Qualitatively, we provide two straightforward methods to assess the model's output. First, we implement a custom ThreeJs-based motion visualizer to render the ground truth, input, and output sequences side by side for comparison. The visualizer renders custom sequences by specifying skeletal positions at temporally-sparse keyframes in JSON format. Second, we build a motion generator to generate custom motion sequences using the model.

Keywords: Motion in-betweening · Kinematics · Transformer · LAFAN1

1 Introduction

Realistic and accurate animation generation is an important but challenging problem with many applications, including animating 3D characters in films, real-time character motion synthesis in Video Games, and Educational applications. One widely used method to generate animations is motion in-betweening, commonly known as tweening. It generates intermediate frames called in-betweens between two temporally sparse keyframes to deliver an illusion of movement by smoothly transitioning from one position to another.

In traditional animation pipelines, animators manually draw motion frames between a set of still keyframes indicative of the most critical positions the body must be at during its motion sequence. Recent improvements include Motion Capture (MOCAP) technologies [9] and query-based methods [15, 19] to generate animations. However, MOCAP technology is expensive, and human-drawn animations are preferred. With the rise of computer-aided animation, deep learning-based algorithms have enabled the smooth generation of keyframes from sparse

frames by learning from large-scale motion capture data. Existing models currently use Recurrent Neural Networks (RNNs) [7, 10], Long Short Term Memory Networks (LSTMs) [8], and BERT-based models [3, 4].

The complexity of generating character animations includes

1. Replicating complex human behavior to create realistic characters
2. Predominantly used transition generation methods are either expensive or inefficient
3. RNNs/LSTMs, though they can capture long-term dependencies, cannot be parallelized due to the sequential processing of input, resulting in longer training times
4. RNNs/LSTMs do not support transfer learning making it hard to use pre-trained models

Inspired by the concept of self-attention to capture long-term dependencies, this paper proposes a transformer-based model to generate realistic animation sequences. Model generalization constitutes the main effort this framework puts into improving the performance of machine learning predictions. This would be analogous to large text transformer models like GPT-3 [2]. This work not only eases the effort put in by the animators but also helps researchers by unblocking transfer learning for the task of in-betweening, thus introducing a level of generalization into the model.

Overall, the contributions in this paper can be summarized as follows:¹

1. Represent motion in-betweening as a sequence to sequence problem where the input sequence consists of keyframes and the output sequence represents the complete and smoothed motion sequence.
2. Set a baseline for the input sequence by filling the frames between the keyframes with interpolated values.
3. Experiment with the efficiency and viability of using transformers to achieve sequence to sequence translation for human motion and compare them with the existing results.
4. Evaluate the model against other state-of-the-art models [4, 8, 16] for the same task using L2P, L2Q, and NPSS metrics.
5. Build a visualizer and a motion generator that qualitatively evaluates the output of the model in comparison to the ground truth and input sequences.

2 Related Work

The problem is analogous to machine translation, where sequence-to-sequence (seq2seq) architectures are prevalent [1, 18, 21]. “Encoder-only” models like BERT [3] are designed to learn the context of a word based on all its surroundings (left and right of the word), making them suitable for feature extraction, sentiment classification, or span prediction tasks but not for generative tasks like

¹ Code can be found in <https://github.com/Pavi114/motion-completion-using-transformers>.

translation or sequence completion. The pre-training objectives used by encoder-decoder transformers like T5 [17] include a fill-in-the-blank task where the model predicts missing words within a corrupted piece of text that is analogous to in-betweening when motion sequences replace sentences.

Early works in human motion prediction include using Conditional Restricted Boltzmann Machines (RBMs) [20] to encode the sequence information in latent variables and predict using decoders. More recently, many RNN-based approaches like Encoder-Recurrent-Decoder (ERD) networks [5] propose separating spatial encoding and decoding from the temporal dependencies. Other recent approaches investigate new architectures like transformers [13] and loss functions to improve human motion prediction further [6, 12].

Initial approaches in motion in-betweening focused on generating missing frames by integrating keyframe information with spacetime models [23]. The following widely successful method for in-betweening adopted a probabilistic approach, framing it as a Maximum A posteriori Optimization problem (MAP) [14], dynamical Gaussian process model [22], or Markov models with dynamic auto-regressive forests [11]. The latest deep learning approaches include works by Holden et al. [10], and Harvey et al. [7] and helped RNNs dominate this field. The latest work using RNN focuses on augmenting a Long Short Term Memory (LSTM) based architecture with time-to-arrival embeddings and a scheduled target noise vector, allowing the system to be robust to target distortions [8]. Some recent work includes BERT-based encoder-only models [3, 4] that predict the entire sequence in one pass and deep learning approaches for interpolation [16]. However, BERT-based models will be less effective than encoder-decoder models for generative tasks.

3 Methodology

The following sections detail the model architecture, **Tween Transformers**, to perform motion frame completion similar to sentence completion.

3.1 Tween Transformers (TWTR)

The architecture of Tween Transformers (TWTR) consists of four main components:

1. Input masking module
2. Input encoding neural network that encodes each motion sequence and converts the input to a set of sequential tokens
3. Transition generation network that includes a standard transformer comprising encoder and decoder modules with feed-forward and multi-head attention networks.
4. Output decoding neural network that computes a sequence of character motion.

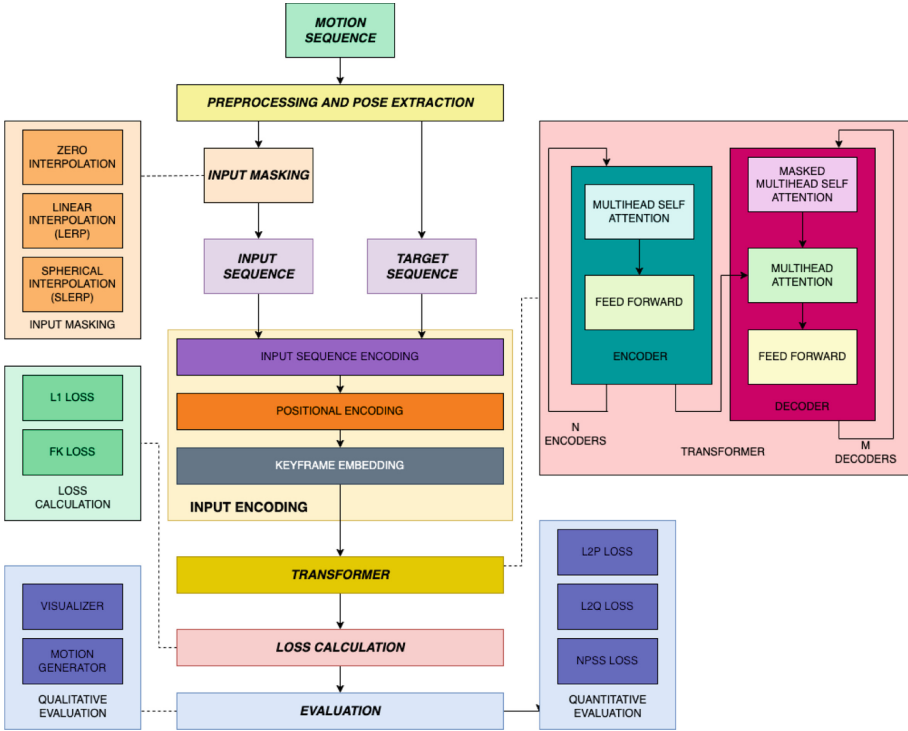


Fig. 1. Model architecture of TWTR

While the transition generation module learns the temporal dependencies, the input and output encoding networks aim to learn spatial dependencies between the different body joints for encoding and decoding motion sequences. Finally, the model also uses multiple losses, including forward kinematics loss, to improve the realism of the generated sequences. It is assumed that the input has both position (x, y, z) and orientation (q_0, q_1, q_2, q_3) variables. Therefore, a single pose can be defined with a root position coordinate $P \in R^3$ and a quaternion matrix $Q \in R^{J \times 4}$, where J represents the joint number of the input pose (here, 22). The following sections discuss the model’s architecture in detail, as indicated in Fig. 1.

Input Masking. There are multiple keyframe gaps k specified in the model configuration. The frames belonging to the keyframe gap are filled with interpolated values derived from the frames constituting the two ends of the keyframe gap. Two kinds of interpolations are carried out and compared. They are implemented in the following ways:

- positions and rotations are linearly interpolated
- positions are linearly interpolated while rotations are spherically interpolated

Input Encoding. As seen in Fig. 1, model encoding has three modules - Input Sequence Encoding, Positional Encoding, and Keyframe Embedding.

1. Input Sequence Encoding:

The input sequence encoder network is a set of three Linear encoders fully connected to two-layer Feed-Forward Networks (FFN) with ReLU activations. The input sequence encoder takes in the global root position $root_p$, local quaternions q , and global root velocity $root_v$ and outputs a set of “sequential tokens”. The hidden sizes of the FFNs are 16, 8, and 8 for q , $root_p$, and $root_v$, respectively. The *embedding* hyperparameter defines the output sizes of the FFNs. The outputs from the FFNs are concatenated to form the output of the input sequence encoding network. Equation (1) describes the Linear Encoder, and Eq. (2) describes the Input Sequence Encoder.

$$L(x) = \text{Linear}(\text{ReLU}(\text{Linear}(x))) \quad (1)$$

$$I(root_p, root_v, q) = L_p(root_p) \parallel L_v(root_v) \parallel L_q(q_1) \parallel \dots \parallel L_q(q_j) \quad (2)$$

where $root_p \in R^3$, $root_v \in R^3$, $q_i \in R^4$, I denotes the Input Sequence Encoder, and L denotes the Linear Encoder.

2. Positional Encoding: Positional encoding, a popular method introduced by Vaswani et al. [21], involves adding a set of predefined sinusoidal and cosine signals to introduce temporal knowledge to the transformer model. The positional encoding for source $\mathbf{Z}_s = [z_{tta,2i}]$ and target $\mathbf{Z}_t = [z_{tta,2i}]$ is computed using Eq. (3)

$$\begin{aligned} z_{tta,2i} &= \sin\left(\frac{tta}{basis^{2i/d}}\right) \\ z_{tta,2i+1} &= \cos\left(\frac{tta}{basis^{2i/d}}\right) \end{aligned} \quad (3)$$

where tta is the number of timesteps until arrival and the *basis* component influences the rate of change in frequencies along the embedding dimension d . A basis of 10,000 is used.

3. Keyframe Embedding: Following previous works [4], the model incorporates additive keyframe embeddings. The keyframe embeddings E_{kf} classify the frames in the sequence into keyframes, unknown frames, and ignored frames. They’re represented by learnable embedding vectors $\{\hat{e}_0, \hat{e}_1, \hat{e}_2\}$ respectively. The keyframe embeddings are represented by Eq. (4), where $e_{kf}^t \in \{\hat{e}_0, \hat{e}_1, \hat{e}_2\}$ and T is the sequence length. The embeddings are added to the input sequence, similar to positional encodings.

$$E_{kf} = [e_{kf}^1, e_{kf}^2, \dots, e_{kf}^T] \quad (4)$$

Transformer. A transformer consists of multiple encoder and decoder layers. Each encoder includes a multi-head self-attention layer (MHSA) and a feed-forward network (FFN), and each decoder consists of a masked multi-head self-attention layer (MMHSA), multi-head attention layer (MHA) and a feed-forward network. The attention function leveraged in the transformer maps a query and a set of key-value pairs - all vectors - to an output. The processing of a single attention head can be represented as follows:

$$Attention(Q, K, V) = Softmax\left(\frac{QK^T}{\sqrt{d_k}}\right)V \quad (5)$$

where $Q = W_q A$ represents a query matrix, $K = W_k A$ represents a key matrix, and $V = W_v A$ represents a value matrix. W_q , W_k , and W_v are the corresponding weight matrices, and d_k represents the dimension of the key matrix. The Query matrix can be interpreted as the keyframe for which Attention is calculated. The Key and Value matrices represent the keyframes that are “attended to”, i.e., how relevant that keyframe is to the query keyframe. In MMHSA, the target is masked before applying the attention mechanism. All the attention outputs are concatenated and sent to the FFN.

Output Decoding. The decoder takes in the concatenated “sequential tokens” outputted by the Input Sequence Encoder and outputs the global root position $root_p$, local quaternions q , and global root velocity $root_v$. To reverse engineer the spatial dependencies, each of the three FFNs, one for each output, comprises two linear layers with ReLU activation. The hidden sizes of the FFNs are the same as in the Input Sequence Encoder, and the output sizes are defined by the original dimensions of the three parameters. Equation (6) describes the Output Decoder.

$$O(x) = (L_p(x[:d_p]), L_v(x[d_p:d_p+d_v]), Q) \quad (6)$$

$$Q = \begin{bmatrix} L_q(x[d_p+d_v:d_p+d_v+d_q]) \\ L_q(x[d_p+d_v+d_q:d_p+d_v+2 \times d_q]) \\ \dots \\ L_q(x[d_p+d_v+(J-1) \times d_q:d_p+d_v+J \times d_q]) \end{bmatrix}$$

where d_p , d_v , and d_q are embedding dimensions for p, v, and q. $x[i:j]$ represents a tensor containing the values in x from the i^{th} index to the $(j-1)^{th}$ index. J denotes the number of joints in the skeleton, $Q \in R^{J \times 4}$ denotes the tensor of stacked quaternions, O denotes the Output Decoder, and L denotes the Linear Encoder.

3.2 Loss Computation

Given a collection of predicted motion sequences and the ground truth, in-betweening loss is computed as the scaled sum of two individual losses - Reconstruction loss and Forward Kinematics (FK) loss.

$$L = \alpha_r L_R + \alpha_{fk} L_{FK} \quad (7)$$

where α_r and α_{FK} are constants to balance the disparity of individual losses. For training we use $\alpha_r = 100$ and $\alpha_{FK} = 1$.

Reconstruction Loss L_R . Reconstruction loss evaluates the ability of the model to “reconstruct” the target sequence from the input sequence. Reconstruction loss accounts for the difference in output and target quaternions values and is computed using an L1 norm. While Harvey et al. [8] compute and sum reconstruction losses for q , x , and *contacts*, they acknowledge that the most important component is q . Reconstruction loss is computed using Eq. (8).

$$L_R = \frac{1}{NT} \sum_{n=0}^{N-1} \sum_{t=0}^{T-1} \hat{q}_n^t - q_n^t \quad (8)$$

where \hat{q}_n^t is the rotational quaternion of the predicted motion sequence n at time t . q refers to the ground truth quaternion. N refers to the number of sequences, and T refers to the length of each motion sequence.

Forward Kinematics Loss L_{FK} . Forward Kinematics loss compares the difference in the global positions of joints between the ground truth and the model’s output. Forward Kinematics loss evaluates the ability of the model to “understand” the relationships between relative angles and global positions. Although the offsets of various joints in the skeleton are not provided to the model, it learns to respect human geometry and maintain correct posture by minimizing the Forward Kinematics loss. The Forward Kinematics loss is computed using Eq. (9).

$$L_{FK} = \|\hat{p}_{global} - p_{global}\|_1 + \|\hat{q}_{global} - q_{global}\|_1 \quad (9)$$

where \hat{p}_{global} and \hat{q}_{global} can be derived from the local coordinates using Forward Kinematics $FK(\hat{p}_{local}, \hat{q}_{local})$ and, similarly p_{global} and q_{global} can be derived from the local coordinates using Forward Kinematics $FK(p_{local}, q_{local})$.

3.3 Training

Following previous works [8, 16], the entire dataset was split into windows of maximum length $T_{max} = 65$. To construct each batch, the number of start keyframes are set to 10 and the number of end keyframes to 1. The number of in-between frames is sampled from the range [5, 44] without replacement.

The weight associated with the number of in-between frames n_{in} is set to be inversely proportional to it, $w_{n_{in}} = \frac{1}{n_{in}}$. This prevents overfitting on the windows with a large number of in-between frames. Shorter windows are sampled more often as they are more abundant and hence harder to overfit. Therefore, the number of unique non-overlapping sequences of a given total length $10 + 1 + n_{in}$ is approximately inversely proportional to n_{in} . Finally, given the total sampled sequence length, the sequence start index is sampled uniformly at random in the range $[0, T_{max} - (1 + 10 + n_{in})]$.

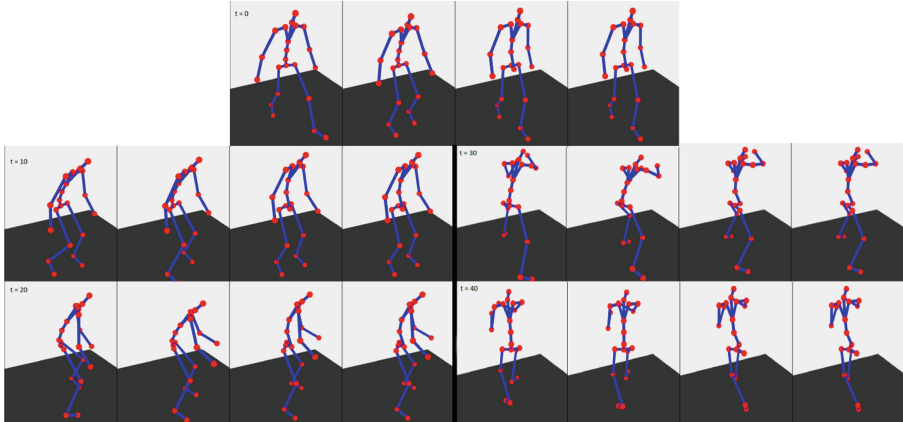


Fig. 2. Stills from the Ground Truth, LERP, Model Output, and Smoothed Output sequences at different timestamps for the action “Aiming2” performed by subject “Subject5”. Considering the frames at $t = 20$, it is clear that the output produced by our model resembles the ground truth more than the interpolated sequence.

4 Setup and Experimental Results

4.1 Dataset

The publicly available Ubisoft La Forge Animation (LaFAN1) Dataset was used for all the experiments. Introduced by Harvey et al. [8] in Ubisoft, LaFAN1 consists of general motion capture clips in high definition. The motion sequences are in BVH format. The LaFAN1 dataset comprises five subjects, 77 sequences, and 496,672 motion frames at 30 fps for a total of 4.6 h. There are around 15 themes, from everyday actions like walking, sprinting, and falling to uncommon actions like crawling, aiming, and a few sports movements. Similar to other works [4, 8, 16], all sequences of subject five were used for testing and benchmarking, with the remaining used for training.

4.2 Evaluation Metrics

The model is evaluated against the L2P, L2Q, and NPSS metrics used in previous studies on the subject five sequences of the LAFAN1 dataset. The L2P defines the average L2 distances of the positions between the predicted motion sequence and the ground truth sequence. Equation 10 shows the L2P calculation. Similarly, the L2Q defines the average L2 distances of the global quaternions. A combination of local quaternions, positions, and motion sequence properties is used to compute these metrics. Equation 11 shows the L2Q calculation.

$$L2P = \frac{1}{NT} \sum_{n=0}^{N-1} \sum_{t=0}^{T-1} \hat{p}_n^t - p_n^t \quad (10)$$

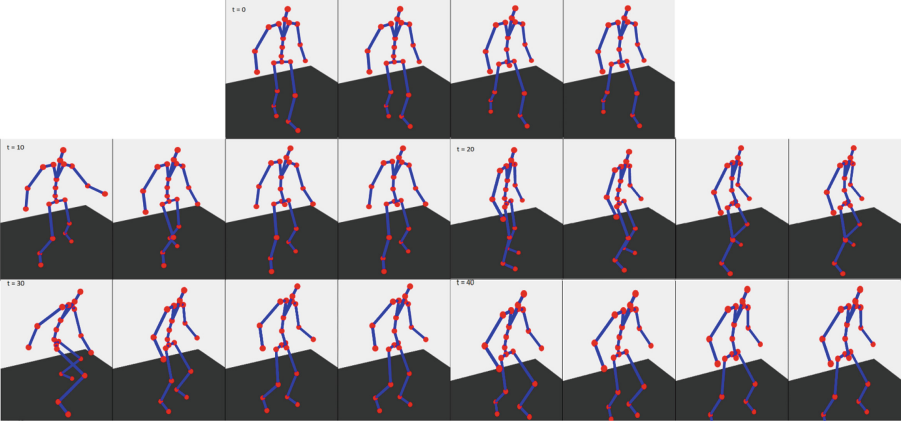


Fig. 3. Stills from the Ground Truth, LERP, Model Output, and Smoothed Output sequences at different timestamps for the action “Dance2” performed by subject “Subject5”. The dance action is unconventional and full of seemingly random movements. Considering the frames at $t = 10$, $t = 20$, and $t = 30$, the output produced by the model is better at $t = 10$, the output produced by interpolation is better at $t = 20$, and neither come close at $t = 30$.

$$L2Q = \frac{1}{NT} \sum_{n=0}^{N-1} \sum_{t=0}^{T-1} \hat{q}_n^t - q_n^t \quad (11)$$

where \hat{q} is the rotational quaternion of the predicted motion sequence n at time t . q refers to the ground truth quaternion. Similarly, \hat{p} refers to the position of the predicted motion sequence p refers to the ground truth position. N refers to the number of sequences, and T refers to the length of each motion sequence.

Normalized Power Spectrum Similarity (NPSS) is an approach comparing angular frequencies with the ground truth. It is an Earth Mover Distance (EMD) based metric over the power spectrum, which uses the squared magnitude spectrum values of the Discrete Fourier Transform coefficients. Equation (12) computes the NPSS metric.

$$NPSS = \frac{\sum_{i=0}^{N-1} \sum_{j=0}^{T-1} w_{i,j} * emd_{i,j}}{\sum_{i=0}^{N-1} \sum_{j=0}^{T-1} w_{i,j}} \quad (12)$$

where $emd_{i,j}$ refers to the EMD distance, and $w_{i,j}$ refers to the weights.

Harvey et al. [8] state that the L2P metric is a better metric than any angular loss for assessing the visual quality of transitions with global displacements as it helps us weigh the positions of the bones and joints. Hence, they argue that L2P is a much more critical metric than L2Q and NPSS.

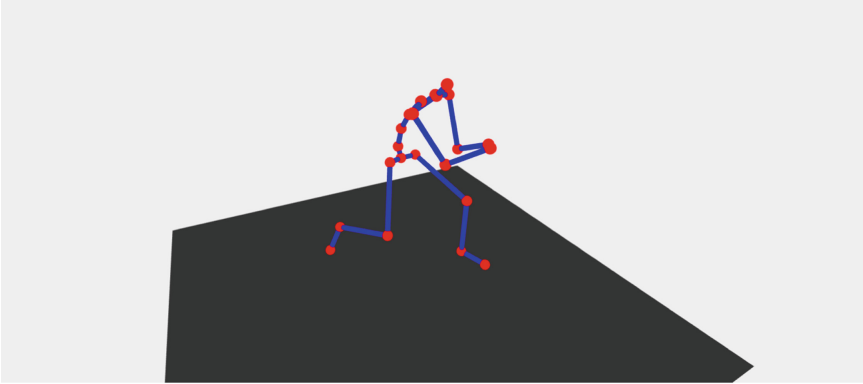


Fig. 4. Still from the motion generator

4.3 Data Preprocessing

First, the local position and orientation values from the BVH files provided in the LaFAN1 dataset [7] are extracted. Twenty-two joints are considered for the skeleton model. Forward Kinematics was used to compute the absolute positions of each joint from the relative positions (relative to hip) given in the dataset. Positions are modeled as standard matrices, and orientations are modeled using quaternions. Further, global position and root velocity are computed from local positions using Forward kinematics.

4.4 Hyperparameters

Most hyperparameters from previous baselines are retained to show the relative improvement in performance using Transformers. This study presents a novel hyperparameter comparison using different interpolation techniques - Linear and Spherical, to compare the performance of several baseline studies. A batch size of 64 for 100 epochs was used. Adam optimizer with a learning rate of 10^{-4} along with a constant dropout of 0.2 was utilized. Keyframe gaps of 5, 15, and 30 were tested to compare the performance of the transformer over higher frame gaps.

4.5 Visualizer and Motion Generator

To qualitatively evaluate the model, a visualizer was built using Node and ThreeJs that juxtaposed the ground truth, interpolated sequence, output sequence, and a smoothed output sequence of the transformer model. The model's output is stored in JSON format and rendered using a custom web-based visualizer. The visualizer was built from scratch using Typescript, NodeJs, Express, and ThreeJs. Figures 2 and 3 show a sample output of the model generated using the visualizer. Further, the motion generator was built using Python,

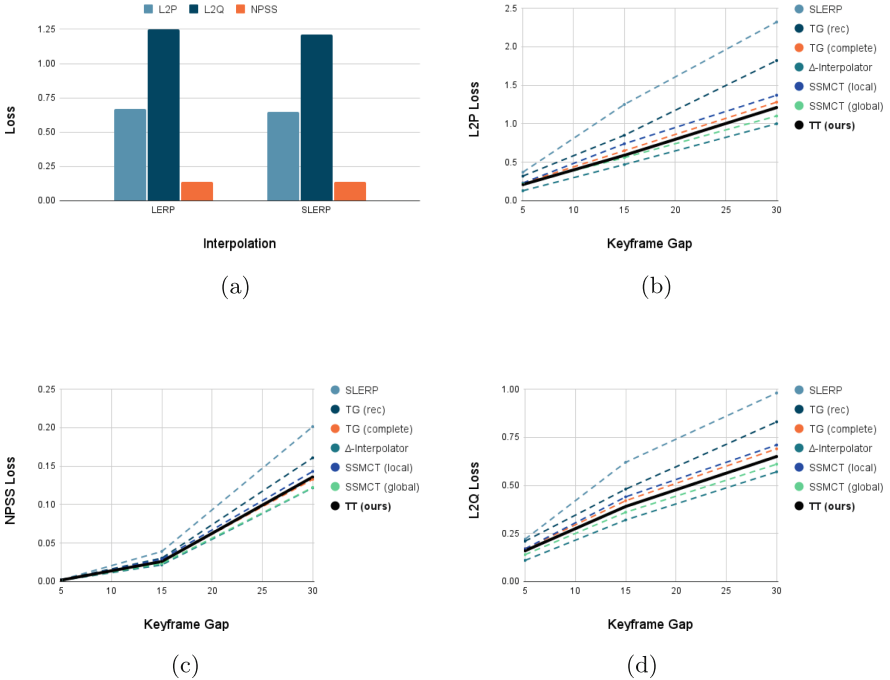


Fig. 5. (a) Comparison of model performance at keyframe gap = 30 with three commonly used metrics - L2P, L2Q, and NPSS, (b) Comparison of L2P losses at various keyframe gaps of the motion in-betweening methods included in this study, (c) Comparison of NPSS losses at various keyframe gaps of the motion in-betweening methods included in this study, (d) Comparison of L2Q losses at various keyframe gaps of the motion in-betweening methods included in this study.

Flask, Node, and Three.js using the visualizer module as a base. The motion generator allows a user to modify keyframes in a given motion sequence and generate in-between frames for the same. The plugin consists of a backend Flask server that uses an instance of our model to generate the in-between frames. Figure 4 shows a still from the motion generator where the stick model is animating a generated custom motion sequence.

4.6 Inferences

As expected, SLERP performs better than LERP. However, it is observed that the performance at 30 fps is almost comparable, as seen in Fig. 5a. This is because the spherical motion becomes almost linear for very short timescales. As seen in Table 1, it is inferred that the Tween Transformer model outperforms the interpolation model and performs closely with the baseline models. Figures 5b, 5d, and 5c confirm that Tween Transformers follow a similar trend to that of

Table 1. The Tween Transformer model is compared with baseline Motion In-betweening methods using L2P, L2Q, and NPSS metrics for various sequence lengths. The Interpolation based methods are included as part of the study. TT (Ours) refers to the Tween Transformer model.

Length	L2Q			L2P			NPSS		
	5	15	30	5	15	30	5	15	30
Zero Velocity	0.56	1.10	1.51	1.52	3.69	6.60	0.0053	0.0522	0.2318
SLERP	0.22	0.62	0.98	0.37	1.25	2.32	0.0023	0.0391	0.2013
TG _{rec}	0.21	0.48	0.83	0.32	0.85	1.82	0.0025	0.0304	0.1608
TG _{complete}	0.17	0.42	0.69	0.23	0.65	1.28	0.0020	0.0258	0.1328
SSMCT _{local}	0.17	0.44	0.71	0.23	0.74	1.37	0.0019	0.0291	0.143
SSMCT _{Global}	0.14	0.36	0.61	0.22	0.56	1.1	0.0016	0.0234	0.1222
Δ -Interpolator	0.11	0.32	0.57	0.13	0.47	1.00	0.0014	0.0217	0.1217
TT (Ours)	0.16	0.39	0.65	0.21	0.59	1.21	0.0019	0.0261	0.1358

other models. Experiments show that training is crucial to obtain a visually smooth output. Moving Average Smoothing was observed to have minimal effect on the output sequence as the model trains.

5 Conclusion

This work presents the Tween Transformer, a novel, robust, transformer-based motion in-betweening technique that serves as a tool for 3D animators and overcomes the challenges faced by existing RNN-based models [8, 16], including sequential training, capturing long-term dependencies, and transfer learning. The generic model treats the application of in-betweening as a sequence-to-sequence problem and solves it using a transformer-based encoder-decoder architecture. It unboxes the potential of robust Transformer-based models for motion in-betweening applications. To conclude, the results encourage the application of low-resource cost-efficient models and enable further developments with the scope of transfer learning on the generalized implementation.

References

1. Bahdanau, D., Cho, K., Bengio, Y.: Neural machine translation by jointly learning to align and translate. In: Bengio, Y., LeCun, Y. (eds.) 3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, 7–9 May 2015, Conference Track Proceedings (2015). <https://arxiv.org/abs/1409.0473>
2. Brown, T., et al.: Language models are few-shot learners. In: Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M.F., Lin, H. (eds.) Advances in Neural Information Processing Systems, vol. 33, pp. 1877–1901. Curran Associates, Inc. (2020). <https://proceedings.neurips.cc/paper/2020/file/1457c0d6bfc4967418bfb8ac142f64a-Paper.pdf>

3. Devlin, J., Chang, M.W., Lee, K., Toutanova, K.: BERT: pre-training of deep bidirectional transformers for language understanding. In: Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers), pp. 4171–4186. Association for Computational Linguistics, Minneapolis, Minnesota (2019). <https://doi.org/10.18653/v1/N19-1423>, <https://aclanthology.org/N19-1423>
4. Duan, Y., et al.: Single-shot motion completion with transformer. arXiv preprint [arXiv:2103.00776](https://arxiv.org/abs/2103.00776) (2021)
5. Fragkiadaki, K., Levine, S., Malik, J.: Recurrent network models for kinematic tracking. CoRR abs/1508.00271 (2015). <https://arxiv.org/abs/1508.00271>
6. Gopalakrishnan, A., Mali, A.A., Kifer, D., Giles, C.L., II, A.G.O.: A neural temporal model for human motion prediction. CoRR abs/1809.03036 (2018). <https://arxiv.org/abs/1809.03036>
7. Harvey, F.G., Pal, C.: Recurrent transition networks for character locomotion. In: SIGGRAPH Asia 2018 Technical Briefs. SA 2018, Association for Computing Machinery, New York, NY, USA (2018). <https://doi.org/10.1145/3283254.3283277>
8. Harvey, F.G., Yurick, M., Nowrouzezahrai, D., Pal, C.: Robust motion in-betweening. ACM Trans. Graph. **39**(4), 1–12 (2020). <https://doi.org/10.1145/3386569.3392480>
9. Holden, D.: Robust solving of optical motion capture data by denoising. ACM Trans. Graph. **37**(4), 1–12 (2018). <https://doi.org/10.1145/3197517.3201302>
10. Holden, D., Saito, J., Komura, T.: A deep learning framework for character motion synthesis and editing. ACM Trans. Graph. **35**(4), 1–11 (2016). <https://doi.org/10.1145/2897824.2925975>
11. Lehrmann, A.M., Gehler, P.V., Nowozin, S.: Efficient nonlinear Markov models for human motion. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2014)
12. Liu, Z., et al.: Towards natural and accurate future motion prediction of humans and animals. In: 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pp. 9996–10004 (2019). <https://doi.org/10.1109/CVPR.2019.01024>
13. Martínez-González, Á., Villamizar, M., Odobez, J.: Pose transformers (POTR): human motion prediction with non-autoregressive transformers. CoRR abs/2109.07531 (2021). <https://arxiv.org/abs/2109.07531>
14. Min, J., Chen, Y.L., Chai, J.: Interactive generation of human animation with deformable motion models. ACM Trans. Graph. **29**(1), 1–12 (2009). <https://doi.org/10.1145/1640443.1640452>
15. Müller, M., Röder, T., Clausen, M.: Efficient content-based retrieval of motion capture data. ACM Trans. Graph. **24**(3), 677–685 (2005). <https://doi.org/10.1145/1073204.1073247>
16. Oreshkin, B.N., Valkanas, A., Harvey, F.G., Ménard, L.S., Bocquet, F., Coates, M.J.: Motion Inbetweening via Deep Δ -Interpolator. arXiv e-prints [arXiv:2201.06701](https://arxiv.org/abs/2201.06701) (2022)
17. Dhariwal, P., Sastry, G., McCandlish, S.: Enct5: Fine-tuning t5 encoder for discriminative tasks (2021)
18. Ren, M., Kiros, R., Zemel, R.S.: Exploring models and data for image question answering. In: Proceedings of the 28th International Conference on Neural Information Processing Systems, vol. 2, pp. 2953–2961. NIPS 2015, MIT Press, Cambridge, MA, USA (2015)

19. Tanuwijaya, S., Ohno, Y.: TF-DF indexing for mocap data segments in measuring relevance based on textual search queries. *Vis. Comput.* **26**(6–8), 1091–1100 (2010). <https://doi.org/10.1007/s00371-010-0463-9>
20. Taylor, G.W., Hinton, G.E.: Factored conditional restricted Boltzmann machines for modeling motion style. In: Proceedings of the 26th Annual International Conference on Machine Learning ICML 2009, pp. 1025–1032. Association for Computing Machinery, New York, NY, USA (2009). <https://doi.org/10.1145/1553374.1553505>
21. Vaswani, A., et al.: Attention is all you need. In: Proceedings of the 31st International Conference on Neural Information Processing Systems NIPS 2017, pp. 6000–6010. Curran Associates Inc., Red Hook, NY, USA (2017)
22. Wang, J.M., Fleet, D.J., Hertzmann, A.: Gaussian process dynamical models for human motion. *IEEE Trans. Pattern Anal. Mach. Intell.* **30**(2), 283–298 (2008). <https://doi.org/10.1109/TPAMI.2007.1167>
23. Witkin, A., Kass, M.: Spacetime constraints. In: Proceedings of the 15th Annual Conference on Computer Graphics and Interactive Techniques SIGGRAPH 1988, pp. 159–168. Association for Computing Machinery, New York, NY, USA (1988). <https://doi.org/10.1145/54852.378507>