

SHUFFLED AUTOREGRESSION FOR MOTION INTERPOLATION

Shuo Huang¹, Jia Jia^{1,3}, Zongxin Yang², Wei Wang, Haozhe Wu¹, Yi Yang², Junliang Xing^{1*}

¹Department of Computer Science and Technology, Tsinghua University, Beijing 100084, China

²CCAI, College of Computer Science and Technology, Zhejiang University

³Beijing National Research Center for Information Science and Technology

ABSTRACT

This work aims to provide a deep-learning solution for the motion interpolation task. Previous studies solve it with geometric weight functions. Some other works propose neural networks for different problem settings with consecutive pose sequences as input. However, motion interpolation is a more complex problem that takes isolated poses (e.g., only one start pose and one end pose) as input. When applied to motion interpolation, these deep learning methods have limited performance since they do not leverage the flexible dependencies between interpolation frames as the original geometric formulas do. To realize this interpolation characteristic, we propose a novel framework, referred to as *Shuffled AutoRegression*, which expands the autoregression to generate in arbitrary (shuffled) order and models any inter-frame dependencies as a directed acyclic graph. We further propose an approach to constructing a particular kind of dependency graph, with three stages assembled into an end-to-end spatial-temporal motion Transformer. Experimental results on one of the current largest datasets show that our model generates vivid and coherent motions from only one start frame to one end frame and outperforms competing methods by a large margin. The proposed model is also extensible to multiple keyframes' motion interpolation tasks and other areas' interpolation.

Index Terms— neural networks, motion interpolation, transformer, human motion, animation

1. INTRODUCTION

When creating character animations, animators use human motion interpolation methods to fill blank frames between hand-made keyframes. The general methods provided by animation software, such as spline curves [1], fail to depict vivid human motions. To achieve better performance, previous researchers attempt to improve the interpolation kernels of the weight functions from two perspectives. They either use general interpolation functions in higher order space, like QLERP [2], SpFus [3], or establish a human joint model and use statistical methods to study it [4]. However, in many cases, these mathematical methods still need manually editing of curve parameters or more keyframes to obtain better visual quality.

Recently several works [5, 6, 7, 8] have contributed to similar tasks. Including motion interpolation, these tasks are collectively referred to as motion completion. The difference between their inputs is shown in Fig. 1. For the in-betweening task, [6] propose time-to-arrival embedding and scheduled target noise, forming a robust RNN-based autoregressive model. For the infilling task, [7, 8] view human motion infilling as spatial-temporal image inpainting and apply autoencoders to reconstruct motions. For both tasks and

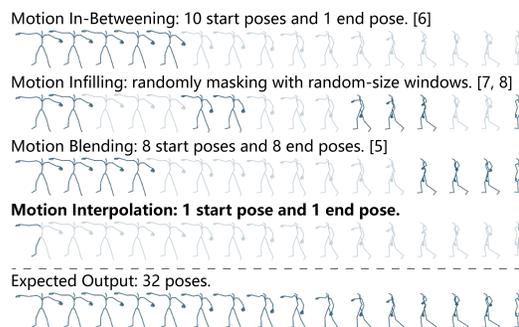


Fig. 1. The differences between input proportions (the unmasked proportions) of motion completion scenarios (motion in-betweening, infilling, blending, interpolation). Please note that the example motion consists of 32 poses, and one pose drawn in this figure stands for two poses in the motion sequence.

the blending task, [5] use a Transformer [9, 10] encoder to modify the linear interpolation, which is a more versatile method.

Though previous deep learning methods have achieved promising results on these similar tasks, there are still some challenges to be solved. On the one hand, the above methods show poor performance when applied to motion interpolation, which is a more complex problem with the smallest input size. Due to the substantial reduction of input information, autoregressive models suffer from severe error accumulation, which hinders them from finishing the transition to the end pose. In contrast, non-autoregressive models tend to predict over-smoothed motions. On the other hand, these methods don't leverage some essence of the interpolation task. The weight function method [1, 2, 3, 4] shows that it is reasonable to generate an interpolation frame with any frame as the condition. Given the linear interpolation, the impact of each keyframe on the resulting frame depends only on their temporal distance. So it is acceptable that the keyframe is in the past or the future, away from or near the resulting frame. This indicates that the frames have flexible dependencies in an interpolation task.

To address the problems above, we propose a Shuffled AutoRegression (SAR) method. It is an extension of the original autoregressive (AR) framework. The AR framework generates future frames from past frames in chronological order. In contrast, as shown in Fig. 2, the SAR framework generates frames in a custom order, and the dependencies between frames are also freely selected, which enables it to catch the flexible inter-frame dependencies. In fact, the SAR framework can express any dependency as long as these dependencies form a directed acyclic graph. For the first problem, SAR al-

*Corresponding author

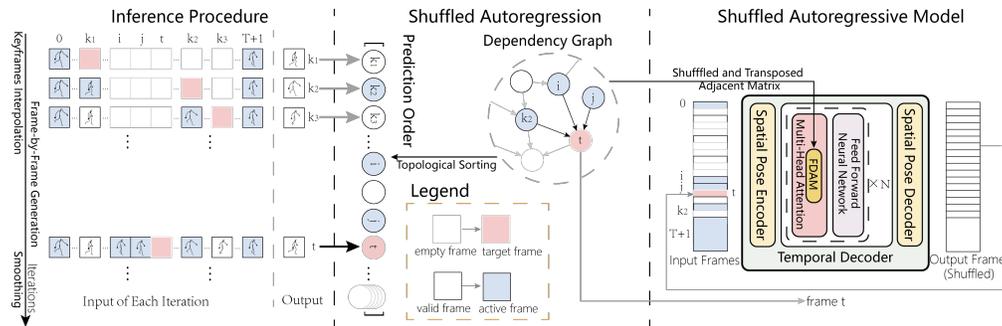


Fig. 2. From left to right are the inference procedure of the SAR, the SAR framework, and the SAR model. The Legend is in the lower middle. An empty frame means that we haven't generated this frame yet. When the current iteration is generating it, it becomes a target frame. After its generation, it becomes valid. If the ongoing iteration depends on a valid frame, then the frame becomes active. Firstly, we design a dependency graph whose topology can be explained as the pipeline of the inference procedure. (k_n in the first stage means the n -th keyframe.) Secondly, we construct the SAR model, deriving the FDAM from the graph. Thirdly, the model works in iterations of the inference procedure. The middle and right parts of the figure shows the iteration when predicting frame t conditioned on frame $0, k_2, i, j, T + 1$. We can find the target frame for every iteration by querying the dependency graph. Then we put it back at the input for the next iteration.

leviates the error accumulation of autoregressive methods and takes advantage of parallel generation. The error accumulates along the dependency graph, so we can manipulate its topology to constrain it.

In particular, we design a DAG topology representing a motion interpolation pipeline. The proposed pipeline contains three stages: 1) keyframe interpolation, 2) frame-by-frame generation, and 3) motion smoothing. The three stages follow under a divide-and-conquer strategy to constraint error accumulation. We incorporate the three stages in an end-to-end Transformer [9] architecture, the attention mechanism of which describes the direct relationships between any two frames. We prove that the architecture is fit to carry out the SAR by controlling the attention mask. Specifically, we propose a Flexible Dependency Attention Mask (FDAM) module, which enables a Transformer Decoder (GPT-like) to carry out SAR.

To evaluate the effectiveness of our model, we evaluate our model for motion interpolation between two frames on the massive AMASS dataset [11]. To enforce our model's generality, we embrace motions of various amplitudes and design different sliding windows for different framerates. Experiments show the effectiveness of our proposed model compared with state-of-the-art deep-learning methods. Although the interpolation between two frames is discussed later in this paper, the flexibility of the SAR makes it easy to extend the model to multiple frames' interpolation.

The contributions of the paper can be summarized as follows:

- We propose a deep learning method to solve motion interpolation, a more complex setting than other works. And it outperforms state-of-the-art methods for motion completion.
- We propose to address the motion interpolation problem with Shuffled AutoRegression architecture and an end-to-end model realizing the SAR. The approach can be extended to interpolation problems in other fields.
- We propose a Flexible Dependency Attention Mask (FDAM) module, which enables a Transformer to carry out the SAR.

2. PROBLEM FORMULATION

Our human body's skeletal structure consists of rigid bone segments linked by J joints, where each joint consists of a relative rotation angle of 3 rotational Degrees of Freedom (DoF). We shall represent a

pose by a tensor $\mathbf{p} \in \mathbb{R}^{J \times 3}$. Given the pose at the start moment $\mathbf{p}^{(0)}$ and the end moment $\mathbf{p}^{(T+1)}$, motion interpolation is to generate a sequence of poses $\mathbf{P} = [\mathbf{p}^{(1)}, \mathbf{p}^{(2)}, \dots, \mathbf{p}^{(T)}] \in \mathbb{R}^{T \times J \times 3}$. The generated motion should accord with human exercise habits and directly connect the given frames. As this problem is non-deterministic, in practice, the results are often optimized to fit into the existing motion data, indicated by the distance of each generated pose from the ground truth. As the bone length of our body model is simplified and inherently encoded, here we optimize the L2 distance of joint angles between generation and the ground truth to a minimum.

3. PROPOSED METHOD

Aiming to reduce the error accumulation of autoregressive models, we propose Shuffled AutoRegression (SAR), whose dependencies form a Directed Acyclic Graph (DAG). (If frame i conditions on the prediction of frame j , then there exists **dependency** from frame i to frame j .) The error accumulates along the edges of the dependency graph. Based on this observation, we design a specific DAG for motion interpolation in which error accumulates much slower than the original order, as shown in Fig. 2. The designed order consists of three stages, which form a pipeline for the task. Note that the SAR can handle any kind of DAG. Here we only state a particular kind of DAG that converges quickly and works well in the attempt. And we compare it with a DAG representing binary search in subsection 4.4.

To carry out the SAR, we leverage the Transformer [9] decoder used in GPT-2 [12] as the backbone of the regressor. Here, we design a Flexible Dependency Attention Mask (FDAM) module, which enables the architecture to fit SAR. We further prove that FDAM is a shuffled and transposed adjacent matrix of the dependency graph.

3.1. Shuffled AutoRegression

3.1.1. Prediction Order and Dependency Graph

Before introducing the concept of SAR, we first elaborate on why the error accumulates so fast in the original autoregressive order. To generate a sequence $\mathbf{X} = [\mathbf{x}_0, \mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_{T-1}]$ (\mathbf{x}_{inp} as default input), the autoregressive prediction order is a sequential order:

$$\mathbf{O}_{AR} = [0, 1, 2, \dots, T - 1], \quad (1)$$

$$\mathbf{x}_i = f(\mathbf{x}_{\text{inp}}, \mathbf{x}_0, \mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_{i-1}; \Theta), \quad (2)$$

where f is the function of an autoregressive model with parameters Θ . However, in motion interpolation, most of the input frames to the model are biased predictions, while only two of them are the ground truth. This leads to an unacceptable error accumulation speed.

We propose Shuffled AutoRegression (SAR) to alleviate the problem. We only choose those elements that are highly related and have fewer errors as references for the following generation to reduce the proportion of deviation. And to make those highly correlated elements happen to be the ones with the minor error, SAR breaks down the left-to-right order along with the half-full dependencies. Our formulas are shown below:

$$\mathbf{O}_{SAR} = [t_0, t_1, t_2, \dots, t_{T-1}], \quad (3)$$

$$\mathbf{x}_{t_i} = f(\mathbf{x}_{\text{inp}}, \text{choice}(\mathbf{x}_{t_0}, \mathbf{x}_{t_1}, \mathbf{x}_{t_2}, \dots, \mathbf{x}_{t_{i-1}}); \Theta'), \quad (4)$$

where \mathbf{O}_{SAR} represents the prediction order of SAR, in which we can decide each t_i freely. And choice function chooses partial input by masking the rest. If nodes of a directed graph represent elements in the sequence and edges represent dependencies in SAR, then they form a **dependency graph**. It is easy to prove that the dependency graph is a DAG, and the prediction order is one of its topological sorting arrays. Note that a DAG may have different sorting arrays. They are all valid prediction orders because some nodes share the same dependencies and can be generated simultaneously.

Error accumulates through edges of the dependency graph, so we can constraint error accumulation by deliberately designing the \mathbf{O}_{SAR} and the choice function, which decides the graph topology.

3.1.2. Pipeline

We instantiate a dependency graph for motion interpolation, representing a pipeline of 3 stages: keyframe interpolation, frame-by-frame generation, and smoothing.

Keyframe Interpolation Frames generated earlier generally have a minor error, making them the best reference for future generations. We select several keyframes, which equally split the entire sequence. Then generating the frames between adjacent keyframes is a smaller-size motion interpolation problem. It is a divide-and-conquer strategy, which can be done recursively by selecting multiple levels of keyframes. For simplicity, we only select one level in the paper.

Frame-by-Frame Generation After sub-problems become simple enough, we solve them by a left-to-right autoregression called frame-by-frame generation. This stage ensures the continuity of the sub-problem because of adjacent dependencies. Keyframe interpolation and frame-by-frame generation can dilute error accumulation in two directions. First, error accumulates through levels of keyframes. Then, it flows through intervals of minimal sub-problems.

Smoothing We apply a global smoothing process by re-generating the sequence depending on all the previous predictions, which is a parallel generation. This stage can also be integrated into the dependency graph by duplicating the previous nodes.

3.2. Backbone of the Regressor

As shown in Fig. 2, the regressor consists of (1) Spatial Pose Encoder, (2) Temporal Decoder, (3) Spatial Pose Decoder, and it work under the guidance of the dependency graph that controls the FDAM.

3.2.1. Spatial Pose Encoder and Motion Embedding

The model receives the motion sequence $\mathbf{P}_{\text{inp}} \in \mathbb{R}^{N \times J \times 3}$ as input. To handle 3D motion sequence, we feed each pose $\mathbf{p}_{\text{inp}}^{(i)} \in \mathbb{R}^{J \times 3}$

into a Spatial Transformer [9] Encoder, which conducts attention between every two joints, receiving an embedding matrix $\mathbf{E}_{\text{inp}} = [\mathbf{e}_{\text{inp}}^{(0)}, \mathbf{e}_{\text{inp}}^{(1)}, \mathbf{e}_{\text{inp}}^{(2)}, \dots, \mathbf{e}_{\text{inp}}^{(N-1)}]^T$, where each embedding $\mathbf{e}_{\text{inp}}^{(i)}$ is a $J \cdot D$ dimensional vector. Then we add sinusoidal position embedding and feed it to the Temporal Decoder.

3.2.2. Temporal Decoder

The Temporal Decoder contains several blocks of GPT-2 [12] with the FDAM Multi-Head Attention. The Temporal Decoder takes \mathbf{E}_{inp} as input and outputs a sequence of embedding \mathbf{E}_{gen} of the same size.

In the FDAM Multi-Head Attention layer, the FDAM inside it can control which input frames are used or masked for generating each $\mathbf{e}_{\text{gen}}^{(i)}$, which realizes the choice function of Equation (4).

Following is the algorithm for generating FDAM. For a specific dependency graph, find one of its topological sort arrays \mathbf{O}_{SAR} . For every element O_i in the \mathbf{O}_{SAR} , find all nodes from which have edges (x_j, O_i) in the graph. Then for each x_j , set $\text{FDAM}[O_{i-1}, x_j] = 1$ (Let $O_0 = 0$). So the FDAM is the transposed adjacent matrix of the dependency graph which is shuffled by a function:

$$\text{FDAM}_{O_{i-1}} = \text{AdjacentMatrix}_{O_i}^T.$$

3.2.3. Spatial Pose Decoder and Shuffled Output

We use an MLP as the Spatial Pose Decoder to decode the embedding \mathbf{E}_{gen} to an output pose sequence \mathbf{P}_{gen} of the same size as \mathbf{P}_{inp} . Because of the SAR, the output is in shuffled order as described in subsection 3.2.2. For every generation, we receive one valid element $\mathbf{p}^{(O_i)}$ from the active path $\mathbf{p}_{\text{gen}}^{(O_{i-1})}$. Then we put it back and run for another iteration.

3.3. Training and Inference

The training method contains two steps. In the first step, we apply a teacher-forcing strategy that facilitates convergence. We optimize MSE loss between the result and ground truth. In the second step, the model's output is entirely generated from the start and end frame, improving test-time performance. We block the gradients of the model and use the trained model to operate the first two stages. Then we feed the trajectory to the network, generating the whole sequence in parallel for the second time. Then We optimize MSE loss between the generated frames and the ground truth. The inference method works in the same way as the second training step.

4. EXPERIMENT

4.1. Dataset

We adopt SMPL+H [13, 14] skeleton and conduct experiments on a dataset constructed from the AMASS dataset [11]. The raw data contains various actions and motion sequences at different frame rates. We follow previous work [6] to cut a long motion sequence into small pieces by sliding a window. We set the sequence length to 31. However, to enforce our model's generality, we embrace motions of various amplitudes and design different sliding windows for different framerates. For a high framerate, we set the window lengths assigned to both 31 and 62 frames, then downsample the latter one. Totally, we have a 166,696 sliced sequence. The constructed dataset is split into training, validation, and test splits consisting of roughly 70%, 10%, and 20% of the samples, respectively.

Implementation Details For pose embedding, we use a Transformer encoder which consists of 4 blocks with embedding dimensions of

Table 1. Quantitative results against several baseline methods and ablation study results. For all metrics, the lower, the better. The Neighbor L2 Distance is shown as the value minus the ground truth value (full model value) for qualitative (ablation) results.

Model	MPJAE	MPJPE	Neigh Dist	NPSS
QN [17]	0.3922	0.6759	105.9001	2.868
BERT [5]	0.0153	0.0466	-1.9475	0.095
CAE [8]	0.0247	0.0852	-1.9628	0.121
SLERP [18]	0.0103	0.0406	-5.2662	0.093
Ours	0.0102	0.0365	-1.0368	0.089
Full Model	0.0102	0.0365	0	0.089
Original AR	0.1847	0.3461	89.395	6.594
Binary Search	0.0152	0.0584	1.3489	0.107
w/o smoothing	0.0121	0.0387	2.0001	0.096

size 1248 (24 for each joint) and 12 heads attention layers. For Temporal Decoder, we use 6 blocks with 8 heads of attention. For prediction order, we select keyframes set [1, 9, 19, 29].

4.2. Evaluate Metrics

Reconstruction Loss We leverage the standard metrics Mean Per Joint Angle Error (MPJAE) and Mean Per Joint Position Error (MPJPE) to evaluate motion interpolation.

Neighbour L2 Distance [15] An indicator of both naturalness and smoothness, which can be viewed as the speed of motion. The closer the measured value is to the ground truth value, the better.

NPSS We adopt Normalized Power Spectrum Similarity (NPSS) [16], which is proved to be highly correlated to the human assessment of motion quality.

4.3. Results

We compare our model with four methods, including neural networks and mathematical functions. For neural networks, We choose Quaternet (QN) proposed by [17], a unified BERT-like model (BERT) proposed by [5] and a convolutional autoencoder (CAE) proposed by [8] as baselines. The selection covers the main types of motion modeling solutions and contains each type’s state-of-the-art methods. Additionally, we use spherical linear interpolation (SLERP) as our baseline of mathematical methods.

Quantitative Results The performance of different methods tested on the test dataset is summarized in Table 1.

Quaternet[17], transferred from the motion prediction task, fails to work since it is autoregressive. Our model outperforms other deep learning models by a large margin on reconstruction losses and NPSS. And our model is the closest to the ground truth value when evaluating the Neighbor L2 Distance. The other models have a lower value than the ground truth value, which indicates that their results are more monotonous. SLERP predicts an average trajectory of the motion space, resulting in advantages in metrics values. Therefore, its good metric values can not reflect its visual quality.

Qualitative Results We randomly select 20 samples from the test dataset. We demonstrate the most representative sample where the motion is taking a step forward in Fig. 3.

The SLERP interpolation is weaker in naturalness than other models. And the uniform rotation of the knees in SLERP’s result leads to penetration with the ground. The autoencoder model [8] generates a trembling upper body because it is a local-oriented model based on CNN. The jitter can be seen more clearly in the video. Our

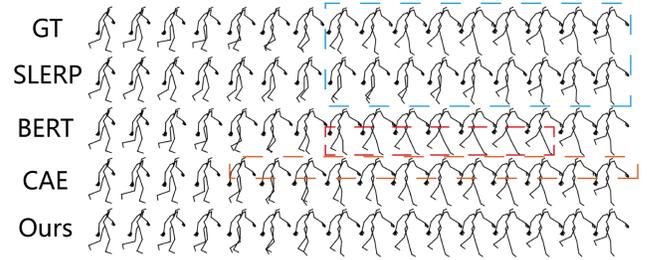


Fig. 3. Qualitative Comparison. The expected motion is taking a step forward. GT stands for the ground truth. Dotted rectangles in the figure mark the defects of methods.

model and the BERT-like model [5] have better results in motion interpolation since both can generate coherent actions. The over-smoothed problem of the BERT-like model from its linear interpolation input has not been fully solved. It can be seen in the figure that the actor of the BERT-like model seems to be throwing his legs powerlessly. Compared with the Bert-like model, our results show the strength of the human body and are closer to the ground truth.

4.4. Ablation Study

We conduct an ablation study on AMASS dataset to test each pipeline function, presented in Table 1. The results show that each component of our model contributes to the performance.

Firstly we train the model with the original autoregression to evaluate the importance of the SAR. The model fails to work as expected. As the frame-by-frame generation progresses, the motion speed gradually decreases due to error accumulation. Finally, the model tends to predict no movement visually and fails to connect the end pose. Secondly, the SAR with a dependency graph that does not contain the frame-by-frame generation is tested. We follow the order of binary search, which recursively generates the middle frame of an interval. The output becomes incoherent because of the scarcity of local dependencies. Thirdly, we train the model without the smoothing step. The growth of Neighbour L2 Distance value shows that the smoothing step significantly improves smoothness.

5. CONCLUSIONS

In this work, we propose the novel Shuffled AutoRegression (SAR) for motion interpolation. Based on the observation that the dependency graph of SAR is a directed acyclic graph, we further propose an idea for constructing a specific kind of dependency graph. The topology of the graph represents a pipeline containing three stages: (1) keyframes interpolation, (2) frame-by-frame generation, and (3) smoothing. Additionally, we devise the Flexible Dependency Attention Mask (FDAM) and plug this module into our backbone regressor. Our framework alleviates the error accumulation problem and generates consecutive and natural motions. Experiments on the AMASS dataset show that our model outperforms other methods for similar tasks. Our framework can also be extended to multiple keyframes’ interpolation tasks and interpolation in other fields.

Acknowledgments: This work is supported by the National Key R&D Program of China under Grant No.2021QY1500, the state key program of the National Natural Science Foundation of China (NSFC) (No.61831022).

6. REFERENCES

- [1] James Ferguson, “Multivariable Curve Interpolation,” *Journal of the ACM (JACM)*, vol. 11, no. 2, pp. 221–228, 1964.
- [2] Ladislav Kavan and Jiří Žára, “Spherical blend skinning: A real-time deformation of articulated models,” *Proceedings of the Symposium on Interactive 3D Graphics*, vol. 1, no. 212, pp. 9–16, 2005.
- [3] Alonso Patron-Perez, Steven Lovegrove, and Gabe Sibley, “A spline-based trajectory representation for sensor fusion and rolling shutter cameras,” *International Journal of Computer Vision*, vol. 113, no. 3, pp. 208–219, 2015.
- [4] Tomohiko Mukai and Shigeru Kuriyama, “Geostatistical motion interpolation,” Tech. Rep. 3, 2005.
- [5] Yinglin Duan, Yue Lin, Zhengxia Zou, Yi Yuan, Zhehui Qian, and Bohan Zhang, “A Unified Framework for Real Time Motion Completion,” *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 36, no. 4, pp. 4459–4467, 2022.
- [6] Félix G. Harvey, Mike Yurick, Derek Nowrouzezahrai, and Christopher Pal, “Robust motion in-betweening,” *ACM Transactions on Graphics*, vol. 39, no. 4, 2020.
- [7] Daniel Holden, Jun Saito, and Taku Komura, “A deep learning framework for character motion synthesis and editing,” in *ACM Transactions on Graphics*, 2016, vol. 35.
- [8] Manuel Kaufmann, Emre Aksan, Jie Song, Fabrizio Pece, Remo Ziegler, and Otmar Hilliges, “Convolutional Autoencoders for Human Motion Infilling,” in *Proceedings - 2020 International Conference on 3D Vision, 3DV 2020*, 2020, pp. 918–927.
- [9] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin, “Attention is all you need,” in *Advances in Neural Information Processing Systems*, 2017, vol. 2017-December.
- [10] Jacob Devlin, Ming Wei Chang, Kenton Lee, and Kristina Toutanova, “BERT: Pre-training of deep bidirectional transformers for language understanding,” in *NAACL HLT 2019 - 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies - Proceedings of the Conference*, 2019, vol. 1, pp. 4171–4186.
- [11] Naureen Mahmood, Nima Ghorbani, Nikolaus F. Troje, Gerard Pons-Moll, and Michael Black, “AMASS: Archive of motion capture as surface shapes,” in *Proceedings of the IEEE International Conference on Computer Vision*, 2019, vol. 2019-October, pp. 5441–5450.
- [12] Radford Alec, Wu Jeffrey, Child Rewon, Luan David, Amodei Dario, and Sutskever Ilya, “Language Models are Unsupervised Multitask Learners — Enhanced Reader,” *OpenAI Blog*, vol. 1, no. 8, pp. 9, 2019.
- [13] Matthew Loper, Naureen Mahmood, Javier Romero, Gerard Pons-Moll, and Michael J. Black, “SMPL: A skinned multi-person linear model,” in *ACM Transactions on Graphics*, 2015, vol. 34.
- [14] Javier Romero, Dimitrios Tzionas, and Michael J. Black, “Embodied hands: Modeling and capturing hands and bodies together,” in *ACM Transactions on Graphics*, 2017, vol. 36.
- [15] Jiashun Wang, Huazhe Xu, Jingwei Xu, Sifei Liu, and Xiaolong Wang, “Synthesizing Long-Term 3D Human Motion and Interaction in 3D Scenes,” in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2021, pp. 9396–9406.
- [16] Anand Gopalakrishnan, Ankur Mali, Dan Kifer, Lee Giles, and Alexander G. Ororbia, “A neural temporal model for human motion prediction,” in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2019, vol. 2019-June, pp. 12108–12117.
- [17] Dario Pavlo, David Grangier, and Michael Auli, “QuaterNet: A quaternion-based recurrent model for human motion,” in *British Machine Vision Conference 2018, BMVC 2018*, 2019.
- [18] Ken Shoemake, “Animating Rotation With Quaternion Curves.,” *Computer Graphics (ACM)*, vol. 19, no. 3, pp. 245–254, 1985.