A U-Shaped Spatio-Temporal Transformer as Solver for Motion Capture

Huabin Yang[1], Zhongjian Zhang[1], Yan Wang[1], Deyu Guan[2], Kangshuai Guo[1], Yu Chang[1], and Yanru Zhang[1,3](✉)

[1] University of Electronic Science and Technology of China, Chengdu, China
{huabinyang12,zhongjianzhang6972,guokangshuai}@gmail.com
yanbo1990@uestc.edu.cn
yuchang@std.uestc.edu.cn
[2] team randomersharp
zhguan_@outlook.com
[3] Shenzhen Institute for Advanced Study, UESTC
yanruzhang@uestc.edu.cn

Abstract. Motion capture (MoCap) suffers from inevitable noises. The raw markers can be mislabeled, occluded, or contain positional noise, which must be refined before being used for production. However, the clean-up of MoCap data is a costly and repetitive work requiring manual intervention of trained experts. To address this problem, this paper proposes a novel end-to-end Transformer-based framework called U-Solver for obtaining joint transformations directly from raw markers (called solving). Through the hierarchical framework composed of decoupled spatio-temporal (DeST) Transformer and the introduction of motion-aware network (MAN) in the temporal self-attention mechanism, U-Solver effectively learns the motion dynamics from both spatial and temporal dimensions. The raw markers can be automatically cleaned and solved through the U-Solver. The experimental results demonstrate that U-Solver outperforms previous state-of-the-art methods in terms of robustness, efficiency, and precision.

Keywords: Motion Capture · Neural Networks · Spatio-Temporal Prior · Motion Capture Solving.

1 Introduction

Motion capture (MoCap) refers to the process of recording and translating the movements into a digital model of an object or human. MoCap systems fall into three broad categories according to the sensors employed, namely mechanical MoCap, inertia MoCap, and optical MoCap. Mechanical MoCap systems rely on the rigid connecting rods and multiple joints fitted with angle sensors to track movements of each marker. Inertial MoCap systems mainly transmit the motion data wirelessly through the inertial sensor, and the movements of each marker are calculated according to the principle of inverse kinematics. Optical MoCap systems track the absolute position of each marker in 3D space by using calibrated multi-view infrared sensors. In contrast to mechanical and inertial MoCap systems, optical MoCap systems can attain high level of accuracy, flexibility, and convenience at the same time, which is able to widely satisfy the requirement of the game, film, and robot industries.

Despite the wide range of applications, optical MoCap systems require a lot of time to clean the incorrect MoCap data mainly including positional noise and tracking errors, which may take several hours per capture even with the aiding of commercial software [38]. As the development of deep learning, a few of denoising solutions of optical MoCap data are proposed based on deep neural network and become the state-of-the-art of this problem [19, 9]. Compared with traditional methods [2, 24, 6, 13], the most important contribution of these solutions is to facilitate the automatic and high-precision cleaning of incorrect markers. Nevertheless, the research of using deep neural network for optical MoCap data solving is just in its infancy. Several drawbacks of current approaches are still worth further enhancing. Recent works either lack sufficient consideration of using temporal information [19] if without the post-processing (temporal filtering), or subjects to errors accumulation because of its non-end-to-end architecture design [9]. Moreover, [19, 9, 20, 37, 29] are not robust enough, which require pre-processing including pose normalization and skeleton normalization. However, the temporal filter may cause the overly smoothed results, the pose normalization may fail when the reference markers (usually a set of chosen markers around torso) are noisy, and the skeleton normalization may result in deviation from reality.

In this work, we present U-Solver, a novel end-to-end Transformer-based framework for optical MoCap data solving. In our framework, the U-Net architecture [34] is adopted due to its unique hierarchical encoder-decoder structure and the skip connections. We propose the **De**coupled **S**patio-**T**emporal (DeST) Transformer to learn the inherent spatio-temporal correlation of human motion, and the **M**otion-**A**ware **N**etwork (MAN) to implicitly model motion dynamics from the perspective of temporal smoothness. The experimental results show that the precision of skeleton position, skeleton rotation and reconstructed marker position generated by the proposed framework are improved compared with the state-of-the-art. The contributions of this paper are briefly listed as:

- A novel end-to-end U-shaped Transformer-based framework is achieved for optical MoCap data solving task with superior performance and more efficient spatio-temporal information utilization in contrast to the state-of-the-art methods.
- A motion-aware network is proposed and combined with acceleration objective function to further enhance the temporal motion dynamics modeling. The good smoothness of the results allows redundant post-processing operations to be removed.
- A robust solution that achieves highest precision even without intervention of the pose normalization and the skeleton normalization procedures, hence removing the hindrance for practical use.

2 Related Work

2.1 MoCap Data Clean-up and Solving

For removing the noise from the MoCap collection and fitting the data captured to the skeletons, approaches based on different prior knowledge were proposed by researchers.

Previously, researches focused on dealing with the erroneous MoCap data. In some approaches, the skeleton structure is usually parameterized [22], or predetermined before production, assuming that the bone lengths are constant and rigid, and the joint rotation angles are within a reasonable range throughout production [18]. PCA-based methods and their variations [7, 26, 36] have been exploit to learn the correlations between markers from the embedding space. The low-rank property is also used to recover MoCap data [15, 16, 23, 27]. These two kinds of methods may become ineffective or inapplicable when too many markers are corrupted. Xiao et al. [41] proposed an algorithm for predicting the missing markers from the perspective of sparse representation of data. Aristidou et al. [2] proposed a method based on the self-similarity of human motions to automatically detect and fix erroneous MoCap data, which avoids the introduction of external databases by finding K-nearest neighbors of a given motion-word. Some researchers proposed to model MoCap data problem as a dynamical system [8, 24]. The Kalman filter as well as its variations [3, 6, 13] are applied to estimate the location of missing marker and reconstruct motion data. However, these methods built upon the assumption that linear models can be used to approximate the non-linear model.

With the recent advancement of deep learning technology, several approaches using neural networks for processing MoCap data have been proposed [19, 20, 9]. By introducing the concept of denoising from machine learning, Holden [19] proposed to produce joint transformations matrix directly from raw markers by training a six-layer ResNet in a simple and end-to-end way. In contrast, Chen et al. [9] proposed MoCap-solver to tackle optical MoCap problem, which requires three high-quality pre-trained MoCap-Encoders (relative to three intrinsic features in raw markers respectively: marker layouts, template skeleton and motion). However, the former's work is per-frame based and does not exploit temporal information, the latter's work utilizes both the temporal and spatial information, but its non-end-to-end structure is time-consuming and may cause the errors accumulation. Besides, optimization based methods such as MoSh++ [30] and DeepMurf [31], which require a pre-trained SMPL model [28] to regress, are also non-end-to-end, they tend to be slow in inference and sensitive to noise.

2.2 Smoothness

Smoothness as kinematic information exists even in noisy motion data. On the one hand, some works focused on the temporal prior knowledge of human motion. By using a temporal filtering operation in the post-processing, the temporal prior is introduced to ensure that the predicted results is smooth and natural

[19, 6, 13]. On the other hand, some researchers have embedded smoothness constraints (smoothness regularization term) into their works as temporal attributes of motion to help generate robust refinements from noisy MoCap data. Such smoothness term is often added to the original objective function [32, 16, 25].

2.3 Rotation Representations

In the realm of computer graphics, 3D rotation is often represented in multiple ways, such as Euler angles, quaternions, or rotation matrix. Euler angles is the most intuitive representation of rotations, but it will suffer from gimbal lock when two axes in a three-gimbal system align. Compared with other rotation representations, quaternions have a more compact representation and are suitable for interpolation. Recently, Zhou et al. [44] pointed out that the commonly used rotation representations in $\mathbb{R}^n$ with $n \leq 4$ have discontinuities and are an obstacle for neural networks to perform regression tasks. Therefore, unlike previous works, we use the 6D continuous rotation representations in our framework.

2.4 Attention Model

Recurrent Models are frequently used architecture for motion modeling tasks [1, 11, 25]. Cui et al. [11] proposed to incorporate a deep bi-directional attention to LSTM to encode and decode motion correlation from forward and backward directions. While the introduction of the attention mechanism does enhance performance, the inherent limitations of the RNN architecture hinder its ability to effectively capture structural correlations.

Transformer [37] has achieved significant performance in natural language processing [37, 12] and computer vision [14, 39]. Taking place one step ahead of the MoCap solving, Ghorbani et al. [17] exploited self-attention mechanism to capture local and global information for labeling unordered point cloud data on a per-frame basis, but which is conducted only on the spatial dimension of data and still lacks the use of temporal information. Luan et al. [29] used K-nearest neighbor information of markers, self-attention and cross-attention to recover human motion from MoCap data. Although it is an end-to-end solution, the number of parameters of the model is too large, which makes the network depth limited.

In contrast, given raw markers (markers may be mislabeled, occluded, or contain positional noise), our work focuses on processing these markers by exploiting both the spatial and temporal information to output joint transformations effectively and efficiently.

2.5 U-Net Architecture

The residual structure and encoder-decoder structure are frequently used structures of denoising neural networks [4, 40, 42]. The advantage of the residual structure is its skip connections, which help to alleviate the possible gradient problem.

Meanwhile, the hierarchical encoder-decoder structure can learn features at multiple scales. However, both structures lack active attention to the dimension of features and are unable to explicitly learn features of a specific dimension, which usually requires the intervention of means such as the dimension adjustment.

U-Net, as a special network with both skip connection structure and encoder-decoder structure, was first proposed by Ronneberger et al. [34] for biomedical image segmentation and has been quickly extended to many applications. Recently, U-Net architecture combined with attention mechanism has been widely used in computer vision such as image denoising [40, 42] and has established new state-of-the-arts due to its inherent scalable architecture as well as multi-scale feature extraction capability.

Motivated by these observations, we designed our framework called U-Solver, whose overall architecture is similar to U-Net and composed of DeST Transformer with a MAN added to the temporal self-attention mechanism. We leverage decoupled spatial and temporal self-attention mechanism to capture spatial and temporal nature of motion, respectively, then a weighted sum of attention is calculated to better model the motion dynamics.

3 Methodology

In this section, we first give the formulation of optical MoCap solving problem, then we describe the overall structure of U-Solver, followed by the details of each component and loss function.

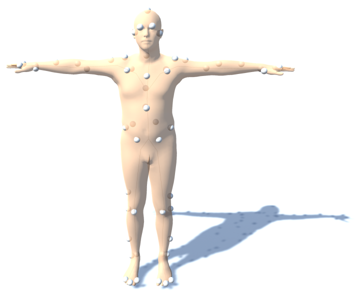


Fig. 1. Marker layout of T-pose. Layouts depend on the placements of markers.

3.1 Problem Formulation

The recorded markers in 3D space tends to be a temporal sequence of point cloud data, which implicitly contains information including motion, subject's body shape and marker layouts (Defined as local offset from each marker to each joint in a T-pose. Different marker placement corresponds to different marker layout. See Fig. 1 for an example).

Our goal is to transform the collected MoCap data in its raw form directly into the motion of the expected skeleton structure. We regard the MoCap solving problem as a regression task and formulate it as given a sequence of corrupted markers as input, producing corresponding smooth and accurate joint transformation matrix as output. Specifically, given a window size of T frames, and a character of J joints with M markers placed on its surface, a temporal sequence of raw markers can be represented as $X \in \mathbb{R}^{T\times M\times 3}$. Besides, the joint transformation matrix is generally expressed as a 3×4 matrix, which consists of a 3×3 rotation matrix and a 3×1 translation vector. However, taking into consideration the continuity of rotation representation contributes to regression task, as mentioned in [44], we transform the 3×3 rotation matrix to its 6D continuous rotation representation, so the output in our paper is represented as $Y \in \mathbb{R}^{T\times J\times(6+3)}$ instead of conventional representation $Y \in \mathbb{R}^{T\times J\times 3\times 4}$.

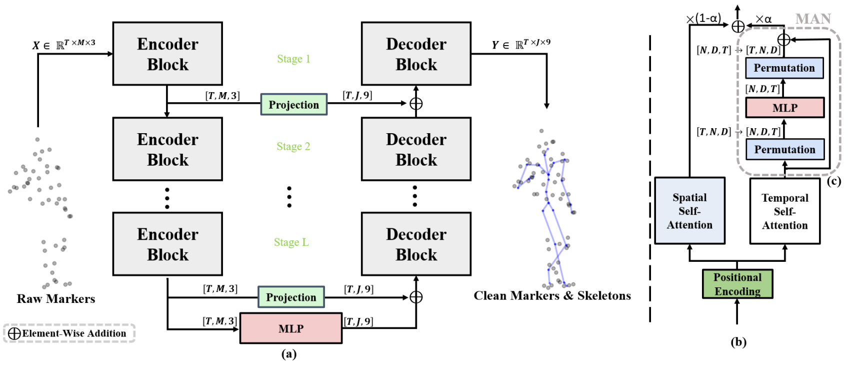


Fig. 2. (a) Overview of U-Solver. (b) Decoupled spatio-temporal(DeST) Transformer. The temporal self-attention network is followed by a motion-aware network. The weighted sum is passed to the next network. (c) Motion-aware network(MAN). The input is first permuted along temporal dimension, then processed through a MLP, and finally re-permuted back to its original dimensions.

3.2 Overall Structure

As is illustrated in Fig. 2, the backbone structure of the U-Solver is based on U-Net, which consists of L stages of encoder and decoder. Each encoder block or its corresponding decoder block consists of a DeST Transformer, with the only difference being the size of the data to be processed (i.e., $[T, M, 3]$ for encoder compared with $[T, J, 9]$ for decoder). In particular, because of the dimensions of the data are different sizes, we apply a projection to the output of encoder block before its skip connection with the parallel stage of decoder, i.e., the output of the l-th stage of encoder $X_l \in \mathbb{R}^{T\times M\times 3}$ is projected through a linear layer to fit the input size $\hat{X}_l \in \mathbb{R}^{T\times J\times 9}$ for the l-th stage of decoder, where $l \in \{1 : L\}$. At bottom, the output of the final stage of encoder X_L are passed through a simple multi-layer perceptron network (MLP) to produce initial input for decoder:

$$\hat{X}_L = \mathrm{MLP}(X_L), \tag{1}$$

where the hidden dimension in the MLP is two times wider than the output dimension. Once the joint positions are obtained, the marker positions can also be easily reconstructed by using a linear blend skinning function [19].

Decoupled Spatio-Temporal (DeST) Transformer Before proposing our methods, we first clarify the notation to be used next. As mentioned above, the encoder and decoder process data with different last two dimensions. In order to simplify the description, we use new notation to uniformly refer to them. From now on, the notation N refers to the number of tokens (i.e., $N = M$ for encoder and $N = J$ for decoder, token refers to marker for encoder and joint for decoder), and the notation D refers to the dimension of token (i.e., $D = 3$ for encoder and $D = 9$ for decoder). Therefore, the dimensions of the data to be processed in U-Solver is represented as $[T, N, D]$.

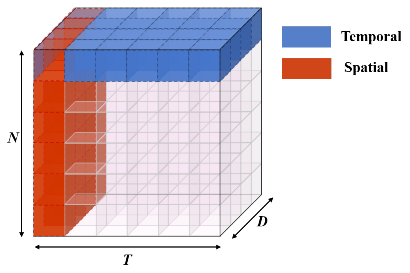


Fig. 3. Computational cost of the decoupled spatial and temporal self-attention mechanism.

We build DeST Transformer based on the nature of motion dynamics that the spatial and temporal information is contextually relevant. Since the computational cost of Transformer being quadratic with respect to the number of input tokens, as shown in Fig. 3, it is difficult to achieve a good balance between low memory requirements and high precision. Therefore, we introduce a decoupled spatio-temporal self-attention mechanism to reduce the computational cost from $\mathcal{O}(T \times N \times D)$ to $\mathcal{O}(T \times D + N \times D)$. Next, we present spatial self-attention and temporal self-attention, respectively.

Spatial Self-Attention The spatial self-attention only calculates self-attention of tokens within the same frame. At each timestamp $t \in \{1 : T\}$, given the full set of N tokens as input $x_t^- = \{x_t^1, ..., x_t^N\}$ with $x_t^n \in \mathbb{R}^D$, where the superscript indicates index of token and the subscript indicates the timestamp. Like vanilla Transformer [37], our spatial self-attention block first generates projection query (Q), key (K) and value (V) using a single linear layer, then the self-attention is computed as:

$$\mathrm{Attn}_{\mathrm{spatial}}(Q, K, V) = \mathrm{softmax}(\frac{QK^T}{\sqrt{d_{\mathrm{model}}}})V = WV, \tag{2}$$

where $Q, K, V \in \mathbb{R}^{T\times N\times d_{\mathrm{model}}}, W = \{W_1, ..., W_T\} \in \mathbb{R}^{T\times N\times N}$, d_{model} denotes the projection dimensions and $K^T \in \mathbb{R}^{T\times d_{\mathrm{model}}\times N}$ denotes the transposed K. For the t-th frame, the spatial self-attention matrix denoted by $W_t \in \mathbb{R}^{N\times N}$ shows how much attention the target token pays to the other tokens within this frame.

Temporal Self-Attention The calculation of temporal self-attention is similar to above-mentioned spatial self-attention. For simplicity, we assume that a token is only related to the same token in contextual frames. i.e., given a temporal sequence of T frames, for each token $n \in \{1 : N\}$, we take $x_-^n = \{x_1^n, ..., x_T^n\}$ as input, generating corresponding projection $\overline{Q}^{(n)}, \overline{K}^{(n)}, \overline{V}^{(n)}$, thus the temporal self-attention is formulated as follows:

$$\begin{aligned}\mathrm{Attn}_{\mathrm{temporal}}(\overline{Q}, \overline{K}, \overline{V}) = \mathrm{concat}(&\mathrm{softmax}(\frac{\overline{Q}^{(1)}\overline{K}^{(1)T}}{\sqrt{d_{\mathrm{model}}}})\overline{V}^{(1)},\\ &...,\\ &\mathrm{softmax}(\frac{\overline{Q}^{(N)}\overline{K}^{(N)T}}{\sqrt{d_{\mathrm{model}}}})\overline{V}^{(N)})\\ = \mathrm{concat}(&\overline{W}^{(1)}\overline{V}^{(1)}, ..., \overline{W}^{(N)}\overline{V}^{(N)})\\ = \overline{WV}&,\end{aligned} \tag{3}$$

where $\overline{Q}^{(n)}, \overline{K}^{(n)}, \overline{V}^{(n)} \in \mathbb{R}^{T\times d_{\mathrm{model}}}$ and $\overline{K}^{(n)T} \in \mathbb{R}^{d_{\mathrm{model}}\times T}$ denotes the transposed $\overline{K}^{(n)}$. The temporal self-attention matrix for token n is denoted as $\overline{W}^{(n)} \in \mathbb{R}^{T\times T}$, which represents how much attention the target token pays to its previous and future state.

Motion-Aware Network (MAN) Despite supporting long-range receptive fields, Transformer focuses primarily on extracting semantic correlations between any two tokens and tends to ignore the continuity of motion in the presence of noise. The Fig. 2 (c) presents the motion-aware network, which draws inspiration from the SmoothNet [43] and consists of a two-layer MLP with the hidden dimension two times wider than input dimension and a residual connection from the input. In challenging frames with long-term jitters, MAN is employed to model the continuity of motion from untrustworthy temporal features, as well

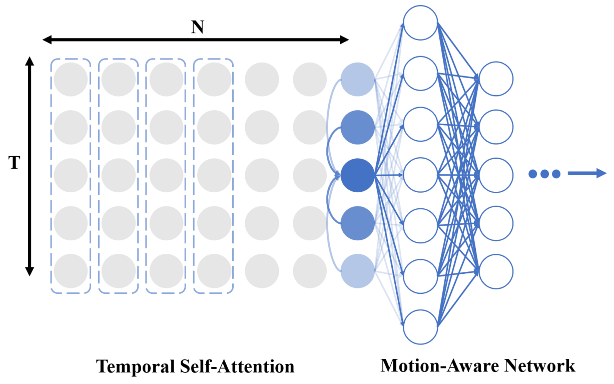


Fig. 4. Motion-aware network models the continuity of motion for each token. The weights and bias are shared among different tokens. For each token (column), T frames temporal self-attention (The weight value is represented by color intensity) are fed into MAN to learn temporal relations for smoothing without considering spatial correlation among tokens.

as to deal with noisy temporal information (See Fig. 4). In addition, in order to facilitate MAN to explicitly learn the temporal relations between consecutive frames, the dimensions of the input data are permuted along the temporal axis as follows:

$$[T, N, D] \rightarrow [N, D, T]. \tag{4}$$

After processing, the data is re-permuted back to its original dimensions:

$$[N, D, T] \rightarrow [T, N, D]. \tag{5}$$

Moreover, in our implementation, the multi-head attention mechanism is applied to both spatial self-attention and temporal self-attention to learn attention from different subspaces.

Weighted Sum as Decoupled Spatio-Temporal Attention The spatial self-attention and temporal self-attention are calculated in parallel, then aggregated together to get the weighted sum of self-attention as follows:

$$\mathrm{Attn}_{\mathrm{DeST}} = (1-\alpha) \times \mathrm{Attn}_{\mathrm{spatial}} + \alpha \times \mathrm{MAN}(\mathrm{Attn}_{\mathrm{temporal}}), \tag{6}$$

where $\alpha \in [0, 1]$ is a learnable trade-off parameter and MAN denotes the motion-aware network. The overall architecture of DeST Transformer is shown in Fig. 2 and a comparison between DeST self-attention and vanilla 2-D self-attention is shown in Fig. 5. Thanks to the designed decoupled spatio-temporal self-attention mechanism, DeST Transformer can better leverage its advantages in capturing long-range dependencies. This allows DeST Transformer to separately learn the temporal and structural characteristics of motion data, while reducing mutual interference between them.

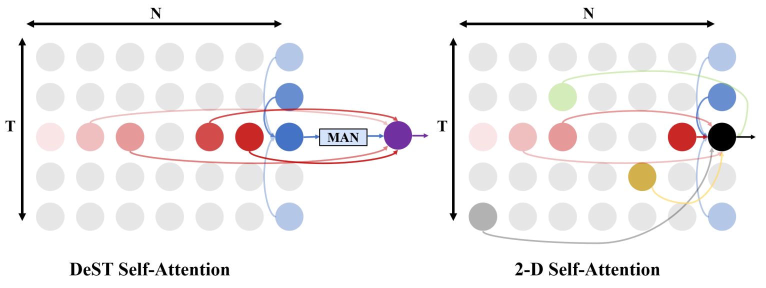


Fig. 5. DeST self-attention vs. 2-D self-attention. For DeST self-attention, the red circle represents spatial self-attention and the blue circle represents temporal self-attention. Compared to vanilla Transformer, the decoupled spatio-temporal design allows the model to explicitly learn the spatial and temporal correlations of every token while maintaining a low computational cost.

Loss Function The motion is often featured by its first-order and second-order information (i.e., velocity and acceleration), which are based on temporal dimension. At each timestamp t the velocity Y' and acceleration Y'' of motion can be defined as the difference between consecutive frames:

$$Y_t' = Y_t - Y_{t-1}, \tag{7}$$

$$Y_t'' = Y_t' - Y_{t-1}'. \tag{8}$$

In order to obtain results with precision and smoothness, we define objective functions as follows:

$$\mathcal{L}_{pose} = \|\hat{Y} - Y\|_1, \tag{9}$$

$$\mathcal{L}_{acc} = \|\hat{Y}'' - Y''\|_1. \tag{10}$$

So joint loss function is then formulated as:

$$\mathcal{L} = \lambda_{pose}\mathcal{L}_{pose} + \lambda_{acc}\mathcal{L}_{acc}, \tag{11}$$

where $\hat{Y}$ denotes the estimated results, $\lambda_{pose} = 0.95$ and $\lambda_{acc} = 0.05$ are the trade-off hyperparameters.

4 Experiments and Evaluation

In this section, quantitative and qualitative research for U-Solver are performed on publicly available datasets. We also compare our framework with previous state-of-the-art methods, and demonstrate that our framework consistently outperforms them. Due to memory limitations, we implement the U-Solver with two-stage encoder and decoder. All the experiments in this paper are conducted on a sever with an Inter Xeon Gold 6142M CPU and an NVIDIA GeForce RTX 3090 GPU.

4.1 Experimental Settings

Datasets We adopt the subsets CMU [10] and MPI-HDM05 [33] in AMASS dataset [30] for our experiments. The AMASS dataset unifies the existing different optical marker-based MoCap datasets, with more than 300 subjects and 11000 motions, making it the largest human motion database available. Specifically, we first split the CMU dataset into training set and test set, with a ratio of 2.5:1, then train our model on the training set of CMU and evaluate it on the test set of CMU and MPI-HDM05. Adopting the same method as in [9], the marker layouts as well as joint transformation are generated and sampled from different SMPL models [28], which ensures that our dataset has an adequate variety of human body shapes and marker distributions. Besides, for simulating raw markers, a simple but effective corruption function proposed by Holden [19] is used. As for the parameters of the corruption function, the probability of occlusion σ^o is set to 0.1, the probability of shifting σ^s is set to 0.1, and the scale of the random shifting β is set to 0.3m respectively. The detailed information of the datasets used in our experiments is shown in Table 1.

Datasets	Frames	Motions	Markers (M)	Joints (J)	Marker Layouts
CMU$_{train}$	5415k	1082			
CMU$_{test}$	2150k	1000	56	24	30
MPI-HDM05	1147k	1000			

Table 1. Details of the datasets.

Metrics We use mean per joint position error (MPJPE, mm), mean per joint rotation error (MPJRE, °), and mean per marker position error (MPMPE, mm) as our evaluation metrics. The MPJPE and MPMPE are both the mean Euclidean distance between predicted positions and ground truth positions. Considering a sequence with T frames, we use the geodesic distance to measure the joint rotational difference, which is computed as:

$$\text{MPJRE} = \frac{1}{T}\sum_{t=1}^{T}\left|\arccos\frac{\mathbf{Tr}((R_t)^{-1}\hat{R}_t)-1}{2}\right|, \tag{12}$$

where R denotes the ground truth rotation matrix, and $\hat{R}$ denotes the estimated results.

4.2 Quantitative and Qualitative Research

In this sub-section, we compare U-Solver with other state-of-the-art frameworks including deep learning based [9, 20, 19, 37, 29] and optimization based [30] methods in various scenarios.

In our experiments, we set batch size to 64, the optimizer is Adam [21], the learning rate is 0.001 with a decay factor of 0.9 for every 5 epochs, and the number of training epochs is 200. As for the other frameworks, for a fair comparison, we follow their origin parameter settings except that the batch size is set to 64. Note that all models presented next are trained only on the training set of CMU dataset and tested on the other two datasets.

Frameworks	Normalization		Evaluation Results		
	Pose	Skeleton	MPJPE	MPJRE	MPMPE
U-Solver	✗	✗	**5.3**	**1.9**	**5.1**
MoCap-solver [9]	✗	✗	18.6	7.2	20.5
	✓	✗	16.1	6.3	18.3
	✓	✓	13.8	5.7	15.8
ResNet [19]	✗	✗	38.7	19.4	35.5
	✓	✗	29.1	11.5	25.4
	✓	✓	24.7	9.4	19.6
MoSh++ [30]	✗	✗	66.7	28.0	61.2
	-	-	-	-	-
	-	-	-	-	-
Joint-Space CNN [20]	✗	✗	41.0	21.1	45.4
	✓	✗	34.7	14.4	34.8
	✓	✓	29.8	12.1	30.2
Transformer Encoder [37]	✗	✗	207.0	39.7	221.2
	✓	✗	199.0	35.1	212.6
	✓	✓	188.1	31.6	200.6
MEMformer [29]	✗	✗	31.1	21.8	28.8
	✓	✗	26.5	15.0	25.0
	✓	✓	24.4	12.0	21.3

Table 2. Performance of different frameworks on the test set of CMU dataset.

Precision To alleviate the difficulty of convergence, pose normalization (rigid body fitting) [5] is introduced to convert markers in global coordinates into local representations before processing [19, 9, 11]. The pose normalization procedure in MoCap solving task refers to robustly find a reference frame to represent contextual data in local space. Generally, a set of markers around torso are selected as reference markers, for reasons that the torso part of the human body is the least prone to deformation and these markers are the least prone to be blocked when capturing, and used to calculate the global rigid transformation matrix relative to the T-pose of the local reference frame. However, this method relies on trustworthy reference markers and may fail when suffers from noise. Although Chen et al. [9] has come up with an neural solution named reliability function, which can retrieve frames in a sequence that contain least noise in reference markers, it still fails to address the extreme cases where all reference markers in all candidate frames are noisy. Therefore, the pose normalization was removed from our framework pipeline. In addition, the skeleton normalization, which scales all

MoCap data to a uniform height by calculating an average skeleton length, was also removed from our framework pipeline as it is not applicable in production environments.

We evaluate all frameworks on the test set of CMU dataset and report the mean prediction error of these frameworks in Table 2. It is evident that the data normalization plays an important role in improving the precision of the results. It can be noted that our proposed U-Solver substantially outperforms other frameworks even without the intervention of above-mentioned normalization procedures and achieves the highest precision on all three metrics (For a visual comparison of different solving results please see Fig. 6).

To further evaluate the generalization ability of our proposed framework, we then evaluate these frameworks on the MPI-HDM05 dataset and present the prediction performance in Table 3. For fairness, the frameworks used in the following experiments do not use above two data normalization methods. It is observed that our framework still achieves the best results over all evaluation metrics.

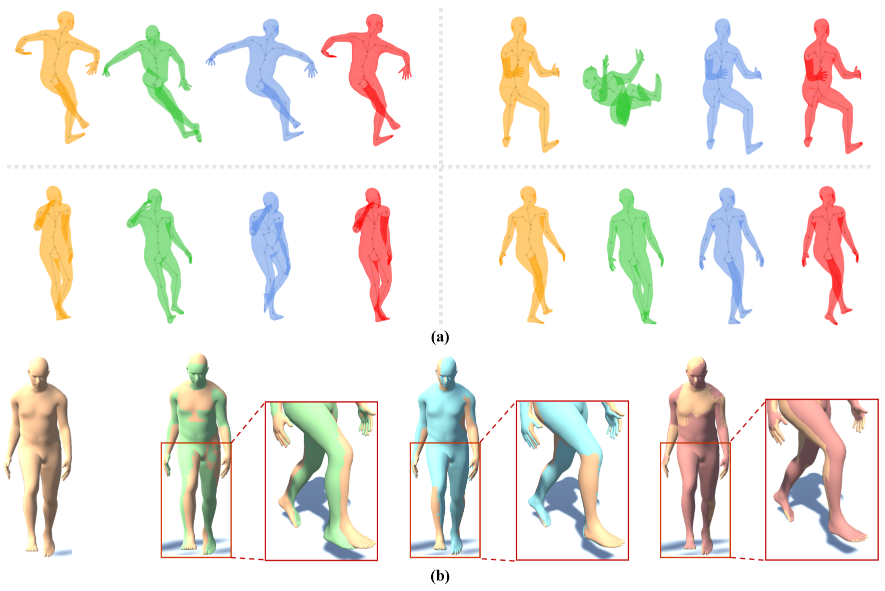


Fig. 6. Visualization of the solving results. From left to right: the ground truth (orange), ResNet (green), MoCap-solver (blue), ours(red). (a) Visualization through a SMPL model. (b) Visualization through a render engine. The ground truth is overlapped with the results of other three frameworks for a better comparison.

Frameworks	MPJPE	MPJRE	MPMPE
U-Solver	**6.6**	**2.7**	**6.1**
MoCap-solver [9]	24.7	15.9	29.6
ResNet [19]	51.3	34.5	49.1
MoSh++ [30]	54.7	19.3	48.2
Joint-Space CNN [20]	62.0	27.0	67.6
Transformer Encoder [37]	324.7	52.5	346.9
MEMformer [29]	48.4	36.7	44.5

Table 3. Performance of different frameworks on the MPI-HDM05 dataset.

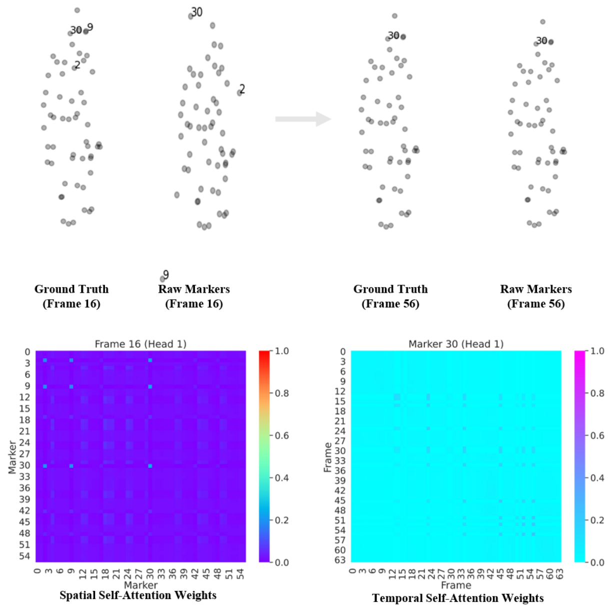
Frameworks	Training	Inference	Parameters
U-Solver	37	536	5793k
MoCap-solver [9]	67	3560	-
ResNet [19]	22	10863	22282k
MoSh++ [30]	-	100	-
Joint-Space CNN [20]	30	2543	5654k
Transformer Encoder [37]	38	2746	9008k
MEMformer [29]	45	108	185907k

Table 4. Average training time, inference speed and trainable parameters of framework.

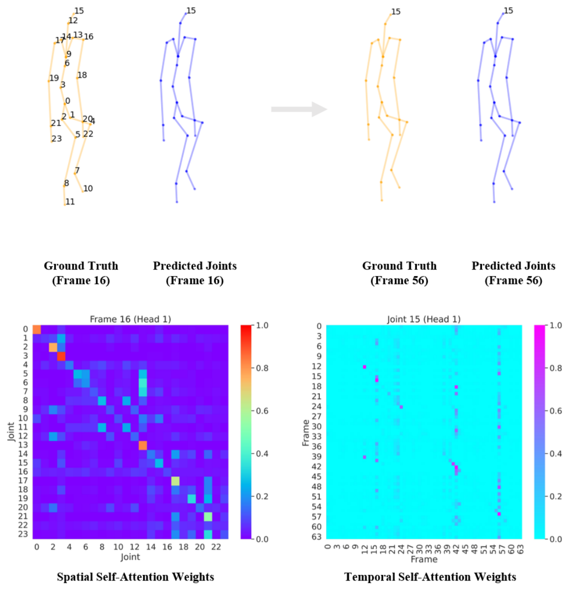
Time The Table 4 presents the average training time (hours) and inference speed (fps) of the evaluated frameworks. The ResNet takes the shortest training time because it has the simplest structure, while the MoCap-solver takes the longest training time due to its complex non-end-to-end structure. MoSh++ [30] as a pre-trained model dose not require training but has the slowest inference speed. The number of parameters of MEMformer [29] is the largest compared to other frameworks.

Visualization of Self-Attention We extract the self-attention weights from the first stage of U-Solver and visualize them in Fig. 7. We note that the spatial self-attention mechanism pays more attention to the noisy markers (See left part of (a)), especially for those who are obviously heterogeneous (i.e., marker 30, 2 and 9). Besides, joint 0, 2, 3 and 13 have higher weights in frame 16 (See left part of (b)). These, to some extent, prove that the spatial self-attention mechanism has learned structural characteristics of human body. As for the temporal self-attention mechanism, some correlations between frames are captured. We note that the temporal self-attention of markers (See right part of (a)) tends to focus on those abnormal frames (e.g., frame 16 and frame 56). Meanwhile, the heat map of temporal self-attention of joints shows some frames are highly correlated, indicating that the long-term global dependencies are captured by DeST Transformer (See right part of (b)).

Robustness to Noise In order to evaluate the robustness of our proposed framework in different noise environments, we first trained a framework on the training set of CMU dataset with a fixed noise intensity, and then conducted several experiments on the test set of CMU dataset with different noise intensity.



(a)



(b)

Fig. 7. Visualization of self-attention. (a) The scatter plots (top) and the heat maps of self-attention weight matrix (bottom) for markers. The spatial self-attention (left) and temporal self-attention (right) related to markers are extracted from the encoder of U-Solver. (b) The joint self-attentions from the corresponding decoder of U-Solver.

Specifically, we trained our framework on the training set with parameters of the corruption function fixed as $\sigma^o = 0.1$, $\sigma^s = 0.1$ and $\beta = 0.3$, then, by varying the parameters of the corruption function, we tested our framework on the test set of different noise intensity. The experimental results in Fig. 8 show that the prediction error of our framework increases with the increase of noise intensity, yet it remains within an acceptable range. Among all the noises, the occlusion noise plays an important role. Although the distribution of noise has changed, our framework still performs well compared to other frameworks.

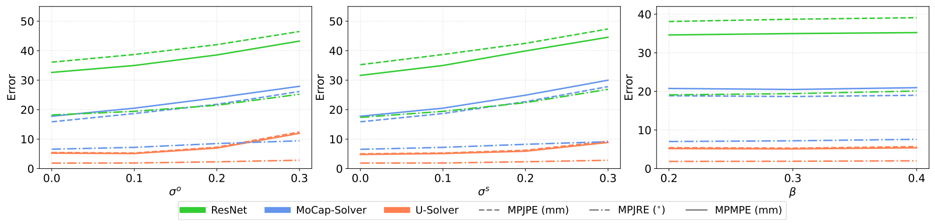


Fig. 8. Comparison of performance of frameworks on the test sets of CMU dataset. All frameworks are trained with fixed noise intensity ($\sigma^o = 0.1$, $\sigma^s = 0.1$ and $\beta = 0.3$) and test on test sets with different noise intensity.

Smoothness Although the Savitzky-Golay filtering [35] or other low-pass filtering is used as post-processing to ensure the temporal continuity of the predicted results, this kind of smoothness was achieved without considering that the high-frequency details about the motion are damaged, resulting in over-smoothed motion and significant errors under long-term jitters. To demonstrate the effectiveness of our framework in smoothing results, a visual comparison is shown in Fig. 9. It can be observed from the visualization results, our U-Solver can produce smoother results without the post-processing. Apart from the temporal continuity, the prediction results of our framework are also the closest to the ground truth. Furthermore, we show the prediction results of the two frameworks (ResNet and MoCap-solver) before and after the post-processing using the Savitzky-Golay filter. It can be seen that large instantaneous jitters and long-term jitters lead to biased errors.

4.3 Ablation Study

Several experiments were conducted to evaluate the impact on performance of the various components in our framework. The results are presented in Table 5, where $\rightarrow$ denotes the module replacement, w/o denotes that this component is removed from our framework, while $w/$ denotes the opposite.

In Table 5, the 1-D TCN represents that a 1-D convolution with a kernel size of 7 is performed along the temporal dimension of data in each layer, and

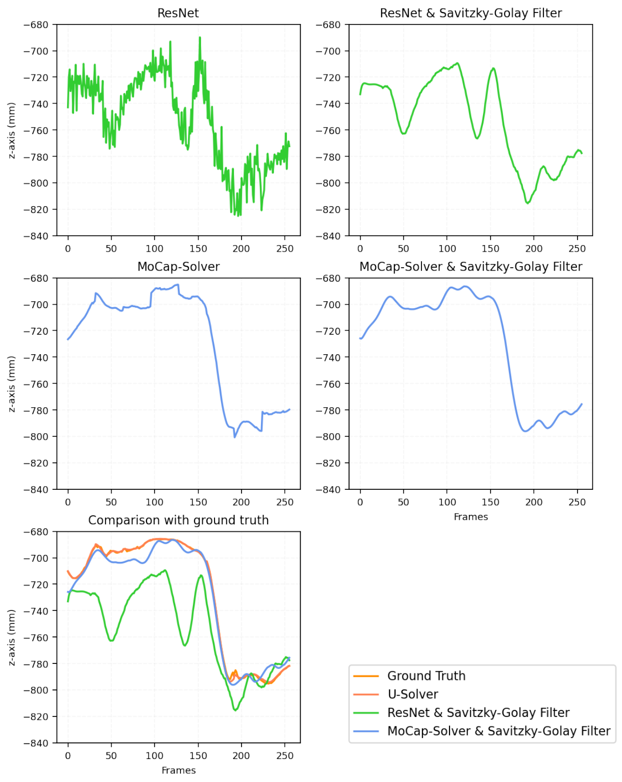


Fig. 9. Position of the joint at right wrist on the z-axis. Top-Left: The raw results of ResNet. The jitters on the trajectory may result from the neural network regarding the noise in markers as the introduction of a large acceleration on movements. Top-Right: The smoothed results of ResNet. The Savitzky-Golay filter is used to filter the raw results of ResNet. Middle-Left: The raw results of MoCap-solver. Middle-Right: The smoothed results of MoCap-solver. Bottom-Left: Comparison of results.

Methods	MPJPE	MPJRE	MPMPE
U-Solver (base framework)	**5.3**	**1.9**	**5.1**
MAN $\rightarrow$ 1-D TCN	7.1	4.1	6.9
DeST $\rightarrow$ 2-D self-attn	24.2	10.4	20.9
U-Net [34]	35.0	15.3	39.1
w/o 6d representation	7.3	6.5	7.4
w/o temporal self-attention	12.4	4.7	12.4
w/o spatial self-attention	16.9	4.3	17.2
w/o encoder	38.4	11.9	37.7
w/o decoder	12.0	6.5	13.5
w/o MAN	7.9	2.8	7.7
w/o $\mathcal{L}_{acc}$	6.1	2.1	6.4
w/ filtering (S-G filter)	5.5	-	5.3

Table 5. Performance of various methods applied to the U-Solver.

the 2-D self-attn represents a non-decoupled 2-D self-attention module. It is observed that the decoupled spatio-temporal Transformer performs better than non-decoupled counterpart. It is also observed that all components in our framework are indispensable. Both the DeST Transformer and the MAN contribute to the improvement of performance, and the combination of them will have better results. Moreover, the 6D continuous rotation representation also plays an important role in reducing the rotation error and also helps to reduce the position errors. In contrast, the use of Savitzky-Golay filter for data post-processing degrades the performance. The position errors were found to increase, indicating that the prediction results are overly smoothed.

5 Limitations and Future Work

Although the proposed framework shows better performance according to the experiment results, several limitations need to be discussed and further research in the future. Firstly, our framework is data-driven and may produce imprecise results when it comes to completely "unfamiliar" poses, which is an inherent flaw in the data-driven approach. Secondly, our framework is not capable of processing data that is inconsistent with the skeleton topology in the training set. The topological changes in body skeleton is not permitted, which requires a new trained framework. Therefore, a more general solution suitable for all skeletal topologies is worth further investigation. Besides, in the future we will extend to evaluate how our framework performs on datasets with more detailed human skeletons (e.g., finger joints). Finally, we will explore how to leverage temporal information for further facilitating automatic marker labeling.

6 Conclusion

In this paper, we have presented U-Solver, which attempts to deal with the optical MoCap data solving problem in a robust and accurate way. By combining the decoupled spatio-temporal Transformer and the motion-aware network in a hierarchical and end-to-end architecture, our U-Solver achieves better performance compared with existing state-of-the-art methods even without the intervention of pose normalization and skeleton normalization. Particularly, our framework pipeline can remove post-processing while preserving temporal continuity of the results, which not only reduces the processing time of the pipeline, but also eliminates biased errors. Experimental results show that the U-Solver outperforms other methods and is more suitable for production usage.

References

1. E. Aksan, M. Kaufmann, and O. Hilliges. Structured prediction helps 3d human motion modelling. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 7144–7153, 2019.

2. A. Aristidou, D. Cohen-Or, J. K. Hodgins, and A. Shamir. Self-similarity analysis for motion capture cleaning. *Computer Graphics Forum*, 37(2):297–309, 2018.
3. A. Aristidou and J. Lasenby. Real-time marker prediction and cor estimation in optical motion capture. *Visual Computer*, 29(1):7 – 26, 2013.
4. L. Bao, Z. Yang, S. Wang, D. Bai, and J. Lee. Real image denoising based on multi-scale residual dense block and cascaded u-net with block-connection. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 1823–1831, 2020.
5. P. Besl and N. D. McKay. A method for registration of 3-d shapes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(2):239–256, 1992.
6. M. Burke and J. Lasenby. Estimating missing marker positions using low dimensional kalman smoothing. *Journal of Biomechanics*, 49(9):1854–1858, 2016.
7. J. Chai and J. K. Hodgins. Performance animation from low-dimensional control signals. *ACM Trans. Graph.*, 24(3):686–696, jul 2005.
8. J. Chai and J. K. Hodgins. Constraint-based motion optimization using a statistical dynamic model. 26(3):8–es, jul 2007.
9. K. Chen, Y. Wang, S. Zhang, S. Xu, W. Zhang, and S. Hu. MoCap-solver: A neural solver for optical motion capture data. *ACM Trans. Graph.*, 40(4), jul 2021.
10. CMU. CMU MoCap Dataset, 2000.
11. Q. Cui, H. Sun, Y. Li, and Y. Kong. A deep bi-directional attention network for human motion recovery. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19*, pages 701–707. International Joint Conferences on Artificial Intelligence Organization, 7 2019.
12. J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.
13. K. Dorfmüller-Ulhaas. Robust optical user motion tracking using a kalman filter. 2007.
14. A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.
15. Y. Feng, M. Ji, J. Xiao, X. Yang, J. J. Zhang, Y. Zhuang, and X. Li. Mining spatial-temporal patterns and structural sparsity for human motion data denoising. *IEEE Transactions on Cybernetics*, 45(12):2693–2706, 2015.
16. Y. Feng, J. Xiao, Y. Zhuang, X. Yang, J. J. Zhang, and R. Song. Exploiting temporal stability and low-rank structure for motion capture data refinement. *Information Sciences*, 277:777–793, 2014.
17. N. Ghorbani and M. J. Black. SOMA: Solving optical marker-based mocap automatically. In *Proc. International Conference on Computer Vision (ICCV)*, pages 11117–11126, Oct. 2021.
18. L. Herda, P. Fua, R. Plänkers, R. Boulic, and D. Thalmann. Skeleton-based motion capture for robust reconstruction of human motion. CA '00, page 77, USA, 2000. IEEE Computer Society.
19. D. Holden. Robust solving of optical motion capture data by denoising. *ACM Trans. Graph.*, 37(4), jul 2018.
20. D. Holden, J. Saito, T. Komura, and T. Joyce. Learning motion manifolds with convolutional autoencoders. SA '15, New York, NY, USA, 2015. Association for Computing Machinery.
21. D. Kingma and J. Ba. Adam: A method for stochastic optimization. *International Conference on Learning Representations*, 12 2014.

22. A. Kirk, J. O'Brien, and D. Forsyth. Skeletal parameter estimation from optical motion capture data. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, volume 2, pages 782–788 vol. 2, 2005.
23. R. Y. Q. Lai, P. C. Yuen, and K. K. W. Lee. Motion Capture Data Completion and Denoising by Singular Value Thresholding. In N. Avis and S. Lefebvre, editors, *Eurographics 2011 - Short Papers*. The Eurographics Association, 2011.
24. L. Li, J. McCann, N. Pollard, and C. Faloutsos. Bolero: A principled technique for including bone length constraints in motion capture occlusion filling. In *Proceedings of the 2010 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, SCA '10, page 179–188, Goslar, DEU, 2010.
25. S. Li, Y. Zhou, H. Zhu, W. Xie, Y. Zhao, and X. Liu. Bidirectional recurrent autoencoder for 3d skeleton motion data refinement. *Computers & Graphics*, 81:92–103, 2019.
26. G. Liu and L. McMillan. Estimation of missing markers in human motion capture. *Vis. Comput.*, 22(9):721–728, sep 2006.
27. X. Liu, Y. ming Cheung, S.-J. Peng, Z. Cui, B. Zhong, and J.-X. Du. Automatic motion capture data denoising via filtered subspace clustering and low rank matrix approximation. *Signal Processing*, 105:350–362, 2014.
28. M. Loper, N. Mahmood, J. Romero, G. Pons-Moll, and M. J. Black. Smpl: A skinned multi-person linear model. *ACM Trans. Graph.*, 34(6), nov 2015.
29. J. Luan, H. Jiang, J. Diao, Y. Wang, and J. Xiao. Memformer: Transformer-based 3d human motion estimation from mocap markers. In *SIGGRAPH Asia 2022 Posters*, pages 1–2. 2022.
30. N. Mahmood, N. Ghorbani, N. F. Troje, G. Pons-Moll, and M. Black. Amass: Archive of motion capture as surface shapes. In *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 5441–5450, 2019.
31. W. Mao, M. Liu, M. Salzmann, and H. Li. Multi-level motion attention for human motion prediction. *International journal of computer vision*, 129(9):2513–2535, 2021.
32. J. Mei, X. Chen, C. Wang, A. Yuille, X. Lan, and W. Zeng. Learning to refine 3d human pose sequences. In *2019 International Conference on 3D Vision (3DV)*, pages 358–366. IEEE, 2019.
33. M. Müller, T. Röder, M. Clausen, B. Eberhardt, B. Krüger, and A. Weber. Documentation mocap database hdm05. Technical Report CG-2007-2, Universität Bonn, June 2007.
34. O. Ronneberger, P. Fischer, and T. Brox. U-net: Convolutional networks for biomedical image segmentation. In *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015*, pages 234–241, Cham, 2015. Springer International Publishing.
35. A. Savitzky and M. J. Golay. Smoothing and differentiation of data by simplified least squares procedures. *Analytical chemistry*, 36(8):1627–1639, 1964.
36. J. Tautges, A. Zinke, B. Krüger, J. Baumann, A. Weber, T. Helten, M. Müller, H.-P. Seidel, and B. Eberhardt. Motion reconstruction using sparse accelerometer data. *ACM Trans. Graph.*, 30(3), may 2011.
37. A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.
38. Vicon. Vicon software, 2023.
39. W. Wang, E. Xie, X. Li, D.-P. Fan, K. Song, D. Liang, T. Lu, P. Luo, and L. Shao. Pyramid vision transformer: A versatile backbone for dense prediction without con-

volutions. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 568–578, 2021.
40. Z. Wang, X. Cun, J. Bao, W. Zhou, J. Liu, and H. Li. Uformer: A general u-shaped transformer for image restoration. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 17683–17693, 2022.
41. J. Xiao, Y. Feng, and W. Hu. Predicting missing markers in human motion capture using l1-sparse representation. *Comput. Animat. Virtual Worlds*, 22(2–3):221–228, apr 2011.
42. S. W. Zamir, A. Arora, S. Khan, M. Hayat, F. S. Khan, and M.-H. Yang. Restormer: Efficient transformer for high-resolution image restoration. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5728–5739, 2022.
43. A. Zeng, L. Yang, X. Ju, J. Li, J. Wang, and Q. Xu. Smoothnet: A plug-and-play network for refining human poses in videos. In *Computer Vision – ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part V*, page 625–642, Berlin, Heidelberg, 2022. Springer-Verlag.
44. Y. Zhou, C. Barnes, J. Lu, J. Yang, and H. Li. On the continuity of rotation representations in neural networks. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5738–5746, 2019.