SynSP++: General Pose Sequences Refinement via Synergy of Smoothness and Precision

Lei Jin*[1], Tao Wang*[1], Dunbo Cai[2], Ling Qian[2], Junliang Xing[3], Jian Zhao[4,5], Xuelong Li[5,6]
[1]Beijing University of Posts and Telecommunications,
[2]Center for Technology Research & Innovation, China Mobile (Suzhou) Software Technology Co., Ltd.,
[3]Tsinghua University, [4]EVOL Lab, Institute of AI (TeleAI), China Telecom,
[5]China School of Artificial Intelligence, Optics and Electronics (iOPEN), Northwestern Polytechnical University,
[6]Institute of AI (TeleAI), China Telecom.
jinlei@bupt.edu.cn, wangtao@bupt.edu.cn, caidunbo@cmss.chinamobile.com,
qianling@cmss.chinamobile.com, jlxing@tsinghua.edu.cn, zhaoj90@chinatelecom.cn,
xuelong_li@chinatelecom.cn

***Abstract*—Predicting human pose sequences via existing pose estimators often encounters various estimation errors. Motion refinement methods aim to optimize the predicted human pose sequences from pose estimators while ensuring minimal computational overhead and latency. Previous investigations have primarily focused on balancing the two objectives, i.e., smoothness and precision, while optimizing the predicted pose sequences. However, we have observed that the inherent trade-off between these objectives can provide additional quality cues related to the predicted pose sequences. These cues, in turn, facilitate network optimization for lower-quality poses. To leverage this quality information, we propose a motion refinement network, termed SynSP++, to achieve a Synergy of Smoothness and Precision in the sequence refinement task. Furthermore, our findings suggest that recalling similar high-quality poses can serve as a reference to correct inaccuracies in predicted poses by paying heightened attention to similar poses. Specifically, we introduce a pose vector database, which can search for similar pose sequences to refine predicted poses from existing pose estimators. Compared with previous methods, SynSP++ benefits from pose quality cues and reference information from similar poses with a much shorter input sequence length, achieving state-of-the-art results among four challenging datasets. Github code: https://github.com/InvertedForest/SynSP2 .**

***Index Terms*—Human Pose Estimation; Human Motion Refinement; Vector Database.**

I. Introduction

Estimating high-quality human pose sequences from RGB videos is essential for automatic driving, human-computer interaction, motion analysis, *etc*. However, human pose estimation [1]–[4] still sometimes exhibits noticeable errors in predicting pose sequences. The pose sequences generated by image-based pose estimators often show significant jitters due to a lack of inter-frame correlation. In contrast, video-based pose estimators alleviate jitters and improve accuracy by leveraging temporal information of input videos. However, training video-based pose estimators requires substantial video data and computational resources [5]–[7].

Consequently, researchers propose motion refinement methods [9], [10] as post-processing techniques to optimize the predicted sequences from pose estimators to approximate the ground-truth sequences from human annotation, with two primary objectives: precision and smoothness. Precision objective aims to improve the position accuracy of the predicted joints through the temporal and spatial information, evaluated by

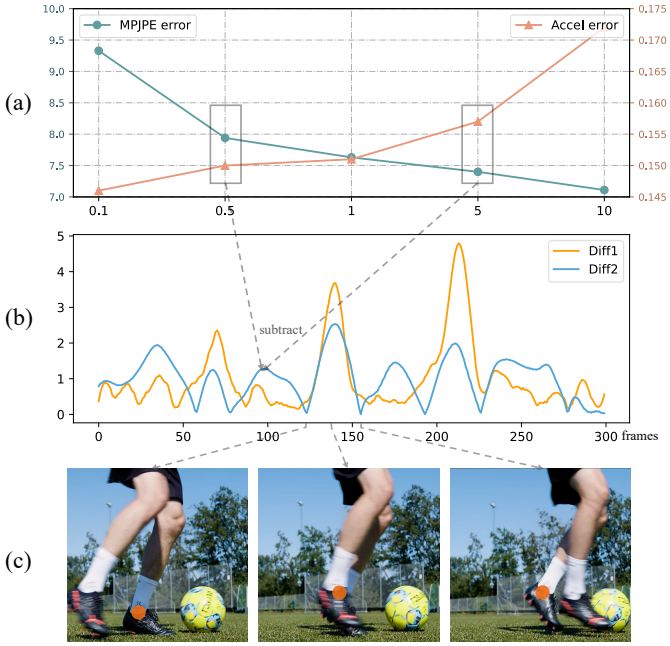

Fig. 1: Illustration of our motivation. (a): The x-axis represents the ratio of weights for Precision and Smoothness loss in the training process of SynSP [8], while the y-axis represents the value of MPJPE (Precision) and Acceleration error (Smoothness). These two errors exhibit a conflicting relationship. (b): Diff1 denotes the difference between the predicted sequence (from the pose estimator) and the gound-truth sequence, and Diff2 denotes the difference between two output sequences from training our model with a loss ratio of 0.5 and 5, respectively. We notice that the trends of Diff1 and Diff2 are roughly the same in (b). (c): A larger value of Diff1 indicates lower prediction quality for the predicted sequences. For example, the left ankle of the person in the middle picture of (c) is occluded, making it difficult for the pose estimator to infer.

*Both authors contributed equally to this research.

Mean Per Joint Position Error (MPJPE) [11], [12]. Smoothness objective does not strive to make the output sequences as smooth as possible but rather to align the output acceleration with that of the ground-truth sequences. Both objectives aim to make the output sequences close to the ground-truth sequences. However, focusing on one of them will greatly increase errors in the other, as shown in Fig. 1. At first glance, the two objectives appear to optimize the predicted sequences harmoniously without conflict. Nevertheless, due to their lack of knowledge about the exact position of the ground-truth data, both objectives attempt to adjust joint positions according to their criteria, leading to a tense relationship. The greater the tension, the more uncertain the network is about the predicted sequences from the pose estimator.

Due to the inherent tension between Precision and Smoothness objectives, previous methods attempt to achieve a satisfactory balance between them. As shown in Fig. 1(a), it can be observed that with an increase in the ratio of weights for precision and smoothness loss during the training process, there is a noticeable rise in Acceleration error (Smoothness) while MPJPE (Precision) decreases, which is a common phenomenon in previous methods [9]. However, we note that the quality of the predicted sequences from the estimator is related to the difference of outputs from SynSP [8] biased towards the Precision and Smoothness objectives. Specifically, as illustrated in Fig. 1(b), the difference (Diff2) between the two output sequences, which focus on precision and smoothness respectively, is correlated with the quality of the predicted sequences, which is generated by the difference (Diff1) between predicted sequences from pose estimators and ground-truth sequences from human annotation. To further verify this correlation, we calculate the Pearson Correlation Coefficient between Diff1 and Diff2 on the Human3.6M and 3DPW datasets. Based on our experiment, the mean Pearson correlation coefficient [13] is 0.48 (bigger than 0.3 is considered as relevant), indicating that the dissimilarity (Diff2) between the two output sequences can serve as an **indicator** for the quality of predicted sequences without knowing the ground-truth. Overall, this tension relationship hinders previous works from achieving optimal performance, but we ingeniously translate this disagreement between smoothness and precision into the quality cue of predicted sequences, enabling our model to surpass previous works with significantly shorter input sequences and lower time delay.

In this work, we propose SynSP [8] and SynSP++ to utilize this quality information. Specifically, we employ a transformer network and encode the quality information by Pose Quality Encoding (PQE) to guide the network's attention toward low-quality poses in sequences. Additionally, in SynSP [8], with the support of the multi-view dataset Human3.6M, we further find that multi-view information is effective in enhancing the reliability of PQE. Therefore, we introduce the Pose Similarity Encoding (PSE) module to exploit the multi-view information by guiding the network's attention towards similar poses from multi-view. In the latest version, since the PSE module is limited to multi-view support from datasets, the goal of SynSP++ is to enhance the multi-view design in a more general manner. Based on the insights presented in

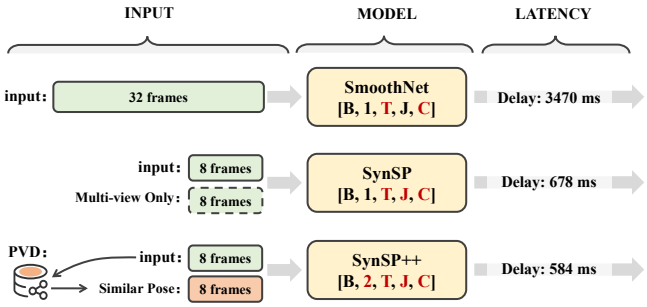

Fig. 2: Illustration of the distinctions among SmoothNet, SynSP, and SynSP++. The model's notation "[B, N, T, J, C]" represents the shape of the input tensor, where B denotes the batch size, N signifies the number of interacting pose sequences, T indicates the frame length of the input sequence, J represents joint number per instance, and C denotes joint spatial dimension. The red characters indicate that these dimensions are interacted with the network, as opposed to the batch dimension. In cases without input from other views, SynSP has an N value of 1. Additionally, SynSP++ consistently retrieves similar poses from PVD and effectively utilizes its network structure. In short, SynSP can only utilize reference information from other views, which is limited due to the demand for multi-view input. But SynSP++ is more general and could get this information from PVD.

Tab. VI, we hypothesize that utilizing similar pose sequences as input instead of pose sequences from multiple views can also benefit the multi-view pipeline of the model. In this way, we introduce a Pose Vector Database (PVD) containing numerous accurate pose sequences, which aids model training and inference by providing similar pose sequences. Implementing this enhancement leads to significant improvements in our model's performance, as demonstrated in Tab. VII, and the distinction between related works is illustrated in Fig. 2.

A preliminary version of this work was presented as a conference paper (termed SynSP [8]). For the human motion refinement task, the state-of-the-art performance of SynSP is achieved by effectively balancing the trade-off between smoothness and precision and leveraging multi-view information. However, the datasets and application scenarios supporting multi-view are limited, so we propose SynSP++ to improve the generality of our work, and there are four main different aspects as shown in 1) Input: we further introduce Pose Vector Database (PVD), which is achieved by pose sequence downsampling, dimensions simplifying, similarity transformation and normalizing, to search similar pose sequence for model training and inference; 2) Model: we make some modifications to the model to deal with the searched pose dimension; 3) Application: with the help of PVD module and the refined model, SynSP++ achieves a wider range of application scenarios and higher performance, while SynSP cannot fully exploit its network structure in the single-view scenes; 4) Format: we restructure the abstract, introduction, and methods sections, conducted substantially more experiments to demonstrate the effectiveness of the proposed SynSP++, and validated the contributions of each component. Thus, SynSP is extended to be SynSP++.

The main contributions are summarized as follows:

- We translate the inherent tension between smoothness and precision into the quality cue of predicted sequences with the Pose Quality Encoding module in SynSP++. Then, we utilize this cue as the weight to help regulate the network's attention on each sequence frame.
- We further observe that similar poses can help rectify the predicted poses from pose estimators. Consequently, we propose the Pose Vector Database as a plug-in pose knowledge base for small models to search out similar poses in order to obtain reference information with very low latency and high accuracy.
- We thoroughly investigate the fusion approach for integrating the input poses and the searched poses, conduct extensive experiments on model architecture, and propose an efficient pose fusion method, which makes the small model obtain an example learning ability.

With the above technical contributions, we have developed a high-performance pose sequence refinement system. We conduct sufficient experiments on four challenging datasets among 2D, 3D, and SMPL representations with image-based and video-based pose estimators. Notably, compared with previous methods SmoothNet and SynSP, MPJPE of SynSP++ is reduced by **25.0%** and **5.1%** on AIST++ dataset with 3D pose estimator VIBE, respectively.

II. Related Work

In this section, we mainly review three streams of related works: human pose estimation algorithms, motion refinement methods, and other pose refinement methods.

A. Human Pose Estimation

Human pose estimation task [3], [14]–[16] is to extract the location of human joints from the collected data. There are 2D, 3D, and SMPL representations for the human pose. 2D representation only contains 2D joint coordinates parallel to the camera plane. 3D representation has one more dimension of depth information. SMPL representation [17] is a skinned vertex-based model that represents a wide variety of body shapes in natural human poses while it needing more parameters to represent.

Human pose estimators can be divided into image-based methods [14] and video-based methods [18]. Pose sequences from image-based methods often show a lot of jitters when applied to videos, while video-based methods require more data and computing resources for training. 2D pose estimator, Hourglass [14] found a outstanding stacked hourglass networks for human pose estimation; SPIN [19] can learn to reconstruct 3D human pose and SMPL pose via model-fitting in the loop; video-based pose estimators, TCMR [20] and VIBE [18], can generate smooth pose sequences with temporal information. A few multi-view methods [21], [22] that combine multi-view information to improve accuracy. We conduct the experiments based on the pose estimators mentioned above.

There are multiple reasons for joint jitters in a video, but they are all caused by errors generated by the human pose estimators. According to our knowledge, the errors leading to jitters can be categorized as follows: 1). Miss error: Failure in joint localization. 2). Swap error: Confusion between the joints of different individuals, resulting in incorrect joint identification. 3). Inversion error: Confusion between joints within a person due to their similarity, such as left and right hands. 4). Jitter error: Small localization error of a joint, which is the main cause of joint jitters. Image-based pose estimators are more likely to have these errors due to a lack of temporal information than video-based pose estimators. However, SynSP++ can improve the performance of image-based estimators and make them comparable to the video-based pose estimators, as shown in Tab. II of the manuscript.

B. Motion Refinement Methods

The predicted human pose sequences can be further refined by motion refinement methods, making the position and acceleration of each joint in these sequences closer to those of the ground-truth sequences. The motion refinement methods can be divided into low-pass filters and deep learning-based methods. The traditional low-pass filters, *e.g.*, Savitzky-Golay [23], Gaussian1d [24], and One-Euro [10], are generally adapted to smooth the predicted pose sequences, and perform well for those abrupt jitters, such as confusion in left and right hips. Although the processed sequences from the low-pass filter looks smoother, it is difficult to ensure precision simultaneously. Deep learning-based method, *i.e.*, SmoothNet [9], DeciWatch [25], and HANet [26], takes the two objectives into account and aims to improve both precision and smoothness. In particular, they propose integrating a dedicated temporal-only refinement network with existing pose estimators to mitigate jitter effectively. SynSP [8] is also based on deep learning and can achieve better results by considering the pose sequence quality and multi-view correlation information. However, multi-view scenes may be limited, and the related multi-view datasets are also scarce. Therefore, we propose SynSP++ to leverage reference information from another pose sequence while maintaining the original design of utilizing tension between smoothness and precision.

Additionally, several human pose prior methods [27]–[30], which try to learn a prior distribution of valid human poses. Although time-consuming for their sampling steps, these methods can be applied to the pose generation/completion/refinement task. HuMoR [31] introduces a novel conditional VAE which enables expressive and general motion reconstruction and generation; GAN-based method [32] proposes a simple yet effective prior for SMPL model to bound it to realistic human poses; ACTOR [33] learns an action-conditioned variational prior using Transformer VAE; GFPose [34] employs a novel score-based generative framework to model plausible 3D human poses. However, they refine poses primarily based on a prior pattern rather than temporal context. In this paper, we also conduct a comparison experiment with the state-of-the-art pose prior method GFPose.

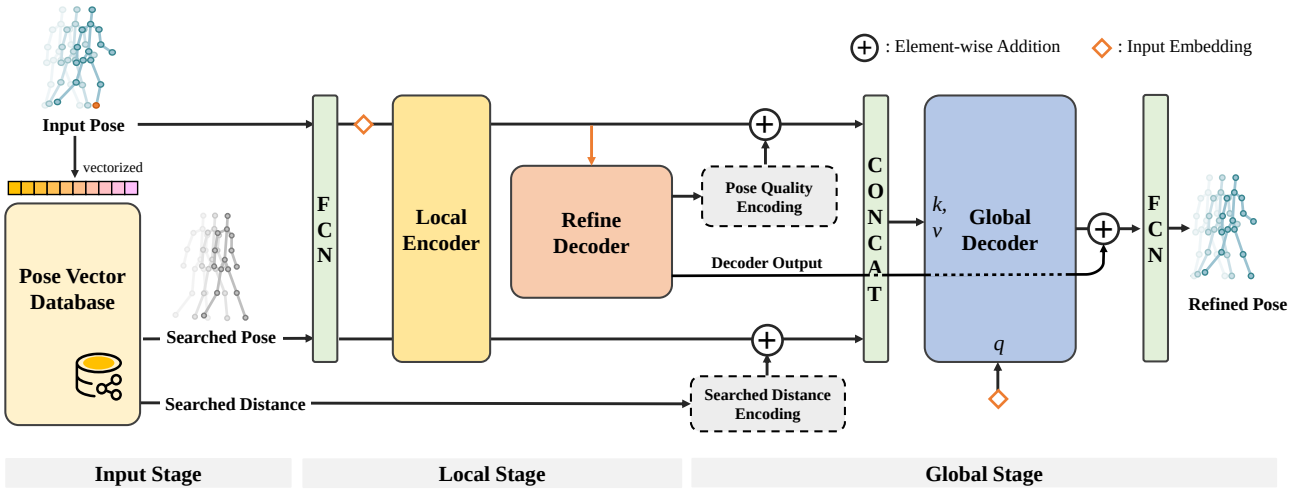

Fig. 3: The architecture of SynSP. Here is the pipeline: 1) In the **Input Stage**, the input pose sequence is vectorized and sent to the Pose Vector Database (PVD) for searching similar pose sequences and calculating their L2 distance. 2) In the **Local Stage**, both pose sequences interact locally with themselves. The input pose sequence undergoes thorough analysis in the Refine Decoder module, generating Pose Quality Encoding (PQE) and Decoder Output. 3) In the **Global Stage**, PQE is added to local information of the input pose to adjust its confidence level. In contrast, Searched Distance Encoding (SDE), generated by PVD, is added to provide distance information for local information of the the searched pose. Subsequently, these components are concatenated and fed into Global Decoder for information fusion. The output result from Global Decoder is combined with Decoder Output from Refine Decoder before passing through a fully connected network. Finally, the refined output result is obtained. Detailed design motivation is given in Sec III.

C. Other Pose Refinement Methods

There are other forms of pose refinement works. However, some pose refinement modules are not transferable to other methods, *e.g.*, Graph-PCNN [35] is a two-stage human pose estimation model with graph pose refinement module, and the pose refinement module is coupled within the model, and needs its unique embedding. Other pose refinement methods are not designed to work with videos but images, *e.g.*, PoseFix [36] takes images and poses as inputs to improve the accuracy, albeit at the cost of increased computational resources and inference time.

III. Method

In this section, we first introduce the task of motion refinement and overall architecture of SynSP++ in overview. Then we describe our model with three stage, *i.e.*, input stage, local stage, and global stage. Finally, we present our training and inference process in details.

A. Overview

Task Description. Motion refinement methods aim to enhance the precision and smoothness of predicted pose sequences generated by human pose estimators [3] by aligning them with ground-truth sequences. In this study, the predicted pose sequence $\mathbf{P}$ from a pose estimator is represented as a tensor in $\mathbb{R}^{T\times J\times D}$, where *Time* denotes the number of frames in the predicted pose sequences, *Joint* represents the number of input pose's [1] joints associated with datasets, and *Dimension* indicates the spatial dimensions of these joints. Throughout this paper, we use $P_{(t,j,d)}$ to denote a scalar value representing the position of joint j, at frame t, in dimension d. We use $\overline{P}_{(t,j,d)}$, $\hat{P}_{(t,j,d)}$, and $\tilde{P}_{(t,j,d)}$ to represent the input pose, ground-truth pose, and searched pose, respectively.

Overall Architecture. We propose SynSP++ to leverage two types of information: 1) quality information derived from the tension relationship between the two objectives, *i.e.*, smoothness and precision; 2) reference information obtained from similar high-quality pose sequences that can be utilized for the error pose repair. As illustrated in Fig. 3, SynSP++ is mainly based on the Transformer structure [37] and consists of three stages. In the Input Stage, the input pose is matched with its similar poses from a pose vector database, and their L2 distances are computed before being fed into the model. In the Local Stage, the input pose and similar searched pose focus solely on self-analysis, preparing for subsequent global optimization. In the Global Stage, all information is aggregated into the Global Decoder to generate the final result. Further details about this pipeline can be found in the caption of Fig. 3.

B. Input Stage

In contrast to large models, small models with limited parameter capacity cannot retain all reasonable human pose sequences in memory, resulting in a behavior that resembles an effective filter for the motion refinement task. Consequently, we propose to search for reasonable reference pose sequences for our small model to learn in the Input Stage. This stage involves searching for similar poses within the training set and incorporating them into the network as supplementary knowledge to optimize the input pose sequences.

[1] In this paper, we will use both 'pose' and 'pose sequence' interchangeably for simplicity.

Pose Vector Database. We propose the Pose Vector Database (PVD) as a plug-in pose knowledge base for small models that store high-quality pose sequences. This enables the search for similar pose sequences to assist in motion refinement tasks during training and inference. Leveraging the vector similarity search function provided by the Milvus vector database [38], PVD offers real-time processing with high throughput and low latency, providing appropriate poses and corresponding similarity scores.

The key aspect in constructing PVD lies in effectively transforming input pose sequences into appropriate vector representations that remain invariant to variations in viewing angle and distance of the pose. Here, we present the pose alignment process employed within the PVD module: 1) A template pose is designed to represent human poses with essential joints by removing unimportant joint points such as fingers and eyes. 2) The time dimension of input pose sequences is down-sampled by keeping only the pose data at the start and end frames to reduce computation. 3) The joint dimension of input pose sequences with varying joint formats is reduced to align with the minimal joint dimension of the template pose. 4) Similarity transformation is applied to align input poses with the template pose. Finally, the aligned input poses are integrated into PVD as vectors while preserving pose information with minimal data usage for low latency. It is crucial to emphasize that only training set data is imported into PVD, and pose sequences from the same individual can solely search for poses from other individuals to prevent network overfitting during training.

Searched Distance Encoding. We further introduce similarity distance between input pose and searched pose as a factor to guide the our model's attention. Popular similarity metrics are used to measure similarities between pose vectors. Currently, Milvus supports Euclidean Distance (L2), Inner Product (IP), Cosine Similarity (COSINE), and binary metric. Given that we have aligned pose sequences with the template pose using matrix similarity transformation, we opt for the L2 distance as the return type. The utilization of L2 distance is crucial in adjusting the attention of our model, thus we refer to it as Searched Distance Encoding (SDE) and denote it as $SDE_{(1)}$.

Application of PVD. During the training process, if we only recall poses that are most similar to the input poses, it may result in network overfitting due to the inherent interdependence between the input and searched poses. Therefore, we search the three most similar poses for each pose sequence and randomly select either the top-ranked instance or one of the remaining two comparable poses with a 50% probability. This approach ensures that our network possesses generalization capabilities across varying levels of similarity. However, we exclusively choose the most similar poses during the inference process.

C. Local Stage

The local stage first processes the information of the input pose and searched pose independently in the Local Encoder and then produces the quality information of input pose in the Refine Decoder. In our experiment, we observe that the small model SynSP++ has a decline in performance when early interaction occurs between input pose and searched pose, even with unique encodings for each pose. We hypothesize that this could be attributed to the similarity between these two poses, which presents challenges for the network to differentiate and process those similar pose information within limited parameters. To address this issue, we propose handling these two types of poses in the Local Encoder, respectively.

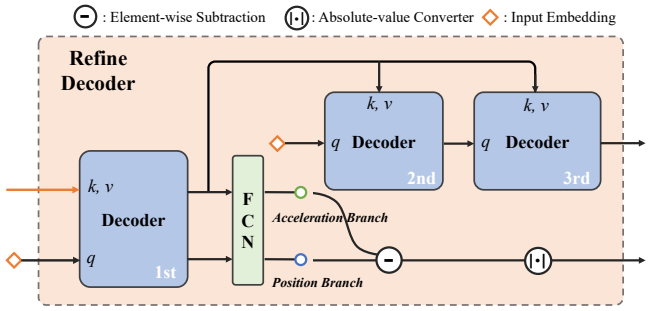

Fig. 4: Illustration of Refine Decoder. The blue decoders in the figure have the same decoder layer structure and number of decoder layers, except the first decoder has two parallel layers at the end as shown in Equ. 2.

Local Encoder. The input pose and searched pose in the Local Encoder are initially processed separately by fully connected layer, aiming to encode the joint dimension J and spatial dimension D to perceive human body posture. The encoding formula is presented as follows.

$$\begin{aligned} E^{0_{th}}_{(t,e)} &= \textbf{FCN}(P_{(t,j\times d)}), \\ E^{d_{th}}_{(t,e)} &= \textbf{ELayer}^{d_{th}}(E^{(d-1)_{th}}), d \in [1,5]. \end{aligned} \tag{1}$$

This formula provides an encoding template, where the input pose and searched pose instantiate this template to encode their respective information. In this formula, E represents the embedding containing pose information, and **ELayer** is the encoder layer, the basic component of Transformer [37]. $\textbf{ELayer}^{d_{th}}$ denotes the d_{th} decoder layer, $\mathrm{FCN}(\cdot)$ is a fully connected network. Through experiments, we have found that 5 layers of the encoder layer can provide better performance. In this way, we can obtain local encoding information $\overline{E}$ and $\tilde{E}$ of input and searched pose, respectively.

Refine Decoder. As previously mentioned, an early interaction between input pose and searched pose can lead to a decrease in performance, because the model need more parameters to distinguish the two poses. To address this issue, we propose Refine Decoder, which focuses solely on processing the input pose. Our experiments showed that parallel operations on the searched pose resulted in little performance gain compared to their computational cost and latency.

The structure of the Refine Decoder is shown in Fig. 4. At first, we stack 5 standard decoder layers of Transformer [37] into the first decoder. The Base Decoder takes the local information $\overline{E}^{0_{th}}_{(t,e)}$ from the fully-connected network as the query, to interact with the local information $\overline{E}^{5_{th}}_{(t,e)}$ from the Local Decoder, which serves as keys and values for the first decoder (You can find what $\overline{E}^{0_{th}}_{(t,e)}$ and $\overline{E}^{5_{th}}_{(t,e)}$ stand for in Equ. 1). At the last layer of the first decoder, we use

two decoder layers to generate two branches biased towards smoothness and precision simultaneously. The formula of the first decoder is as follows.

$$\begin{aligned} K, V &= E^{5_{th}}_{(t,e)},\ Q^0 = E^{0_{th}}_{(t,e)}, \\ Q^{d_{th}}_{(t,e)} &= \textbf{DLayer}^{d_{th}}(Q^{(d-1)_{th}}, K, V), d \in [1, 3], \\ Q^{pos}_{(t,e)} &= \textbf{DLayer}^{4_{th}}(Q^{3_{th}}, K, V), \\ Q^{acc}_{(t,e)} &= \textbf{DLayer}^{5_{th}}(Q^{3_{th}}, K, V), \\ P^{pos}_{(t,j)} &= \textbf{FCN}(Q^{pos}_{(t,e)}), \\ P^{acc}_{(t,j)} &= \textbf{FCN}(Q^{acc}_{(t,e)}), \end{aligned} \tag{2}$$

where Q, K, and V denote the query, key, and value for the first decoder, respectively. $\textbf{DLayer}^{d_{th}}$ denotes the d_{th} decoder layer, $\mathrm{FCN}(\cdot)$ is fully connected network. The input $E^{0_{th}}, E^{5_{th}}$ canb found in Equ. 1. In this way, SynSP++ can generate two pose sequences P^{pos} and P^{acc} from its Acceleration Branch and Position Branch, which are biased towards the precision and smoothness objectives in the final loss function Equ. (7), respectively.

The first decoder serves three purposes: 1) Enhancing the model's performance by supervising the output in the middle of the model [4]. 2) Providing Pose Quality Encoding of input pose for the subsequent Global Decoder. 3) The output of the first decoder can be used as input for the next two decoders, providing additional information Q^{local} in the following manner.

$$\begin{aligned} K, V &= Q^{acc}_{(t,e)},\ \overline{Q}^0 = E^{0_{th}}_{(t,e)}, \\ \overline{Q}^{d_{th}}_{(t,e)} &= \textbf{DLayer}^{d_{th}}(\overline{Q}^{(d-1)_{th}}, K, V), d \in [1, 10], \\ Q^{local}_{(t,e)} &= \overline{Q}^{10_{th}}_{(t,e)}. \end{aligned} \tag{3}$$

Pose Quality Encoding. As shown in Fig. 1, we have empirically observed that when outputs from the two branches show a greater discrepancy, the quality of the input pose is poorer. Specifically, we compute the Pearson Correlation Coefficient between the prediction quality and the difference between the two outputs is about 0.48, which shows that the two have a strong correlation. Hence, we propose the Pose Quality Encoding (PQE) to utilize this correlation to improve the final prediction and assign confidence to the input pose sequence on the time dimension T. PQE is calculated as follows:

$$\begin{aligned} P^{dis}_{(t,j)} &= |P^{acc}_{(t,j)} - P^{pos}_{(t,j)}\ |, \\ PQE_{(t)} &= \frac{1}{J}\sum_{j=1}^{J} exp(-\frac{P^{dis}_{(t,j)}}{\max\limits_{j \in J} P^{dis}_{(t,j)} + \sigma}). \end{aligned} \tag{4}$$

Here, we calculate PQE by the distance P^{dis} between the middle outputs P^{acc} and P^{pos}, which are generated by Acceleration Branch and Position Branch as shown in Fig. 4, respectively. To ensure that extreme values do not interfere with PQE, P^{dis} is normalized by dividing the maximum value in the individual's sequence. We introduce a small constant σ (set to 1×10^{-6}) to avoid division by zero. We do not use the softmax function for normalization because softmax needs to manually set the temperature coefficient to avoid extreme output distribution without knowing the mean value of P^{dis}.

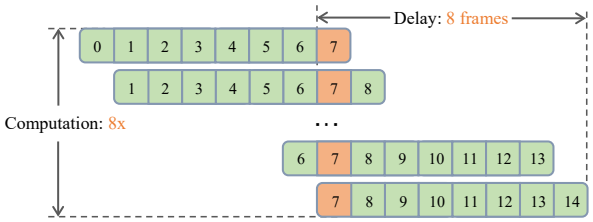

Fig. 5: Illustration of Sliding Window Average Algorithm (SWAA), widely used by motion refinement works. This figure proves that the amount of computation and latency is proportional to the window length.

D. Global Stage

Now, we have obtained all the supplementary information, including Pose Quality Encoding, searched poses, and corresponding Searched Distance Encoding. Specifically, PQE is a pose sequence quality indicator to achieve synergy of smoothness and precision. Besides, the searched pose and its corresponding SDE provide additional reference information to refine the final prediction. We use the embedding $\overline{E}^{0_{th}}_{(t,e)}$ of the input pose as a query, with the outputs and their respective encodings of Local Encoder as k and v. Similar to the Refine Decoder, we employ 5 layers of standard decoders to generate the encoding of the output pose as shown in Equ. 5.

$$\begin{aligned} E^{input}_{(t,e)} &= \overline{E}^{5_{th}}_{(t,e)} + PQE_{(t,1)}, \\ E^{search}_{(t,e)} &= \tilde{E}^{5_{th}}_{(t,e)} + SDE_{(1,1)}, \\ G_{t,e} &= concat(E^{input}_{(t,e)}, E^{search}_{(t,e)}, dim = 1), \\ K, V &= G_{t,e},\ D^{0_{th}}_{(t,e)} = \overline{E}^{0_{th}}_{(t,e)}, \\ D^{d_{th}}_{(t,e)} &= \textbf{DLayer}^{d_{th}}(D^{(d-1)_{th}}, K, V), d \in [1, 5], \end{aligned} \tag{5}$$

where $\overline{E}^{d_{th}}_{(t,e)}$ and $\tilde{E}^{d_{th}}_{(t,e)}$ denotes the embedding of input pose and searched pose from Local Encoder, respectively. Then they are combined with their encodings and passed through 5 decoding layers. Moreover, all tensor addition operations need to consider the broadcasting mechanism in the python package NumPy [39].

Finally, by adding the output of Global Decoder to that of Refine Decoder and then passing them through a fully connected network, we obtain the final result $P^{output}_{(t,j,d)}$. Here, $Q^{local}_{(t,e)}$ contains information of input pose and can be found in Equ. 3.

$$\begin{aligned} P^{output}_{(t,j\times d)} &= \textbf{FCN}(D^{5_{th}}_{(t,e)} + Q^{local}_{(t,e)}), \\ P^{output}_{(t,j,d)} &= P^{output}_{(t,j\times d)}. \end{aligned} \tag{6}$$

E. Training and Inference

In contrast to previous studies, our approach employs a window length of T=8 for both training and inference. This choice significantly reduces the time delay by $(32-8)/12 = 2$s when the pose estimator operates at an inference speed of 12fps, as compared to the window length of 32 [9]. Moreover, this reduction in window length also leads to a decrease in computational requirements to $8/32 = 25\%$, as illustrated in Fig. 5.

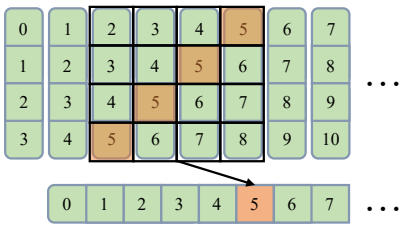

Fig. 6: Illustration of Sliding Window Average Algorithm (SWAA) with parallel computing. Take the output window length of 4 as an example. The numbers in the figure represent the frame number of the output, and each column represents the output sequence of the network. It can be observed that the calculation of SWAA can be implemented in convolution operation (the black wire frame represents the convolution kernel).

Training. The losses of the three outputs, *i.e.*, Acceleration Branch output and Position Branch output, and final output, share the same calculation method. It is noted that the first two outputs are generated in Refine Decoder, as shown in Fig. 4. We have two losses, $\mathcal{L}_{acc}$ and $\mathcal{L}_{pos}$, for each one of the three outputs:

$$\begin{aligned} A_{(t,j)} &= (P_{(t+2,j)} - P_{(t+1,j)}) - (P_{(t+1,j)} - P_{(t,j)}), \\ \mathcal{L}_{\text{pos}} &= \frac{1}{V \cdot T \cdot J} \sum_{v=1}^{V} \sum_{t=1}^{T} \sum_{j=1}^{J} |P_{(t,j)} - \hat{P}_{(t,j)}|, \\ \mathcal{L}_{\text{acc}} &= \frac{1}{V \cdot T \cdot J} \sum_{v=1}^{V} \sum_{t=1}^{T} \sum_{j=1}^{J} |A_{(t,j)} - \hat{A}_{(t,j)}|, \\ \mathcal{L}_{\text{total}} &= \mathcal{L}_{\text{pos}} + \lambda \mathcal{L}_{\text{acc}}. \end{aligned} \tag{7}$$

In our experiments, λ is set as 0, 1, and 0.5 for the Position Branch, Acceleration Branch, and the final output, respectively, as a trade-off between smoothness and precision.

Inference. The outputs during inference go through a post-processing called Sliding Window Average Algorithm adopted by most works as shown in Fig. 5. It can be seen that the delay and computation needs for model and post-processing becomes smaller with shorter window length.

Sliding Window Average Algorithm. The Sliding Window Average Algorithm (SWAA) is essential and common for motion refinement methods to average poses of the same frame from multiple predictions, thereby optimizing one pose at multiple time points. SWAA can reduce the MPJPE error by 27.1% and the Accel error by 45.4% for SmoothNet and SynSP in the Human3.6M dataset with FCN estimators. As shown in Fig. 5, the delay mainly comes from the waiting time for real-time input video. For videos that have already been processed, the calculation speed of SWAA is particularly important. As shown in Fig. 6, we accelerate the SWAA through convolution operation and reduce the time of SWAA calculation to 0.374% of the original version, further enhancing the efficiency of motion refinement methods.

IV. Experiments

A. Dataset.

For the fairness of the comparison, we use these datasets made by SmoothNet [9] with kinds of pose estimators in the experiments. To compare with pose prior models, we also follow GFPose [34] to add the same degree of Gaussian or uniformly distributed noise to Human3.6M and CMU-Mocap datasets as input for the pose denoise task. We conduct experiments of SynSP++ in three human pose representations: 2D, 3D, and SMPL. Their visualization is shown in Fig. 7.

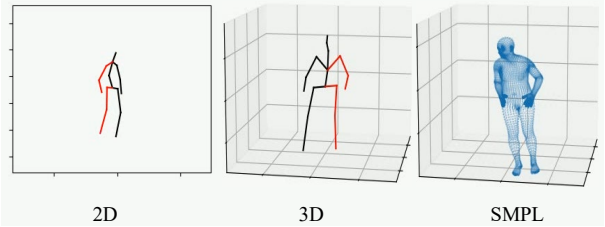

Fig. 7: Illustration of 2D, 3D, and SMPL representations.

Human3.6M dataset [41] is a widely used benchmark for multi-view based 3D human posture research, recording the performance of 5 female and 6 male subjects, under 4 different viewpoints, for training realistic human sensing systems and for evaluating the next generation of human pose estimation models and algorithms. This dataset has 1,559,752 frames for the training and 543,344 for the testing.

3DPW dataset [42] is an important outdoor dataset to evaluate the effectiveness of video-based methods with occlusion and crowded scenes, having 22,463 frames for the training and 35,219 frames for the testing with accurate 3D pose in challenging sequences, including walking in the city, going up-stairs, *etc*.

AIST++ dataset [43] is a new multi-modal dataset of rapid 3D dance motion and music, covering 10 dance genres with multi-view videos. This dataset has 5,858,138 frames for the training and 2,851,927 for the testing.

CMU-Mocap dataset [44] contains most daily behaviors like walking, swimming, and climbing, except for flying and rock climbing, as mentioned on the official website. We split this dataset into 2,241,016 frames for the training and 755,857 frames for the testing.

B. Metrics.

Following the related works [9], we use Accel error, mean per joint position error (MPJPE), and PA-MPJPE (Procrustes-aligned MPJPE) [45], [46] to evaluate the acceleration and position proximity between the predicted sequence and the ground-truth sequence, and the lower metrics the smaller error of the predicted sequence. In addition to MPJPE, Procrustes Analysis MPJPE (PA-MPJPE) is commonly employed to evaluate frame-level precision by mitigating scale, rotation, and translation effects. For 3D poses, the unit of measurement is mm; whereas for 2D poses, accurate localization precision is assessed using pixel within an image.

C. Implementation Details.

Dataset and PVD. When training on each dataset, we use their respective training sets to construct PVD (specific construction method can be found in Sec III-B). We do not merge the

TABLE I: Comparison with related works on Human3.6M, AIST++ datasets with 2D, 3D, and SMPL human pose estimator Hourglass [14], FCN [40], and SPIN [19], respectively. WS denotes the window size in training, inference, and post-process. ↓ indicates that SynSP++ outperforms SynSP.

Method	WS	Human3.6M / 2D			Human3.6M / 3D			AIST++ / SMPL		
		MPJPE↓	PA-MPJPE↓	Accel↓	MPJPE↓	PA-MPJPE↓	Accel↓	MPJPE↓	PA-MPJPE↓	Accel↓
Input	N/A	9.42	7.64	1.54	54.55	42.2	19.17	107.72	74.40	33.20
One-Euro [10]	1	10.69	7.98	0.34	55.20	42.73	3.80	108.97	75.27	14.70
Gaussian1d [24]	32	9.37	7.56	0.51	53.67	41.60	2.43	104.84	72.18	10.05
Savitzky-Golay [23]	32	9.35	7.55	0.17	53.48	41.19	1.34	104.58	72.30	6.07
SmoothNet [9]	32	9.25	7.57	**0.15**	52.72	40.92	1.03	103.00	71.19	**5.72**
SynSP [8]	8	8.13	6.09	**0.15**	51.36	40.13	1.02	84.63	59.02	6.08
SynSP++	8	**7.74**↓	**5.99**↓	**0.15**	**50.44**↓	**39.96**↓	**1.00**↓	**83.11**↓	**58.50**↓	6.04↓

training sets of multiple datasets because: 1) Including too many examples in PVD would slow down the search speed. 2) Each dataset has different annotation styles, such as the annotated positions of hip joints and shoulder locations being more influenced by annotators than eye positions. This can lead to a potential confounding effect on our model.

Training. We initialize the learning rate as 0.001 and decay it by 0.9 per epoch. SynSP++ is trained for 30 epochs with Adam optimizer, requiring approximately four hours and 8GB of GPU memory with a batch size 6000. The computational setup consists of an Intel(R) Xeon(R) Gold 6271C CPU @ 2.60GHz for the CPU and a Tesla V100 SXM2 32GB for the GPU. For the comparative experiments of 2D, 3D, and SMPL representations, we chose the image-based estimators with poor performance to show the powerful restoration ability of SynSP++, *i.e.*, Hourglass, FCN, and SPIN. For video-based methods that perform better with the temporal information, we chose VIBE and TCMR estimators for the experiments. These experiments above cover Human3.6M, 3DPW, CMU-Mocap, and AIST++ datasets.

D. Comparsion with State-Of-The-Arts.

This section demonstrates that SynSP++ consistently outperforms previous methods in refining the predicted pose sequences generated by pose estimators. Then, we compare with pose prior models on the pose denoise task by smoothing the pose sequences with artificial noise. Finally, there are also some methods [25], [26] with a super-long window length, and we also compare them.

2D Representation. The 2D representation is inherently unsuitable for PVD due to its lack of one dimension of information, making it unable to perform real three-dimensional matrix similarity transformation. As a result, two 2D poses obtained from different angles observing the same pose cannot be aligned, leading to a substantial increase in their L2 distance, which hampers the effectiveness of the PVD module. Nevertheless, SynSP++ can still outperform other methods thanks to the analysis and optimization of input poses with multiple decoders in the Refine Decoder module.

In Tab. I, we compare SynSP++ with other motion refinement methods in the Human3.6M dataset with 2D image-based pose estimator Hourglass [14]. SynSP++ can outperforms SmoothNet and SynSP by 16.3% and 4.8% on MPJPE, respectively. We believe that the improvement in accuracy is due to network structure of SynSP++ and the searched poses from PVD according to Tab. VII.

TABLE II: Comparison of various video-based pose estimators. + represents the combination of estimators and motion refinement methods. * indicates multi-view inputs. ↓ indicates that SynSP++ outperforms SynSP. ↑ is vice versa.

Dataset	Method	WS	MPJPE↓	Accel↓
3DPW	MAED [47]	16	79.00	-
	MPS-Net [5]	16	84.30	-
	TCMR [20]	16	86.46	6.75
	TCMR+SmoothNet [9]	16+32	86.50	6.00
	TCMR+SynSP [8]	16+8	86.10	5.90
	TCMR+SynSP++	16+8	**85.94**↓	**5.87**↓
	PARE [48]	1	79.00	25.60
	PARE+SmoothNet [9]	1+32	78.10	5.91
	PARE+SynSP [8]	1+8	**76.20**	6.16
	PARE+SynSP++	1+8	77.64↑	**5.90**↓
AIST++	VIBE [18]	16	106.9	31.60
	VIBE+SmoothNet [9]	16+32	97.47	**4.15**
	VIBE+SynSP [8]	16+8	77.00	4.32
	VIBE+SynSP++	16+8	**73.10**↓	4.29↓
H36M	TransFusion* [21]	1	25.52	-
	PPT* [22]	1	25.16	11.60
	PPT*+SmoothNet [9]	1+32	23.97	1.31
	PPT*+SynSP [8]	1+8	21.51	1.10
	PPT*+SynSP++	1+8	**21.43**↓	**1.07**↓

3D Representation. Poses of 3D representation can take full advantage of SynSP++, as they can be aligned through rotation, translation, and scaling in three dimensions by matrix similarity transformation. In Tab. I, we compare SynSP++ with other motion refinement methods in the Human3.6M dataset with 3D pose estimator FCN [40]. As shown in Tab. I, we can still outperform SmoothNet and SynSP by 4.3% and 1.8% on MPJPE, respectively.

SMPL Representation. Then we compare SynSP++ with other motion refinement methods in AIST++ dataset with SMPL pose estimator SPIN [19] as shown in Tab. I. SynSP++ also shows superior performance in SMPL representation.

Comparison with Video-Based Pose Estimators. The video-based estimators already include temporal information. Compared with image-based estimators, motion refinement methods have difficulty optimizing the sequences from video-based estimators. In Tab. II, we compare SynSP++ with

TABLE III: Comparison on Human3.6M dataset for the denoise task. $N(0,\sigma)$ represents a Gaussian distribution with a mean of 0 and a variance of σ, and $U(k)$ represents a uniform distribution at [0,k]. M↓ and A↓ denote MPJPE↓ and Accel↓, respectively. ↓ indicates that SynSP++ outperforms SynSP.

Method	$N(0,100)$		$N(0,400)$		$U(50)$		$U(100)$	
	M↓	A↓	M↓	A↓	M↓	A↓	M↓	A↓
Noisy input	65.6	160.3	131.1	320.5	94.8	231.3	189.5	462.7
GFPose [34]	42.8	-	64.6	-	50.9	-	89.4	-
SmoothNet [9]	42.4	10.7	57.9	14.2	49.4	**10.2**	74.3	16.3
SynSP [8]	37.6	8.8	56.1	13.0	45.6	10.9	69.4	15.7
SynSP++	**36.9**↓	**8.7**↓	**56.0**↓	**12.7**↓	**45.4**↓	10.8↓	**67.2**↓	**15.3**↓

TABLE IV: Comparison on CMU-Mocap dataset for the denoise task. $N(0,\sigma)$ represents a Gaussian distribution with a mean of 0 and a variance of σ, and $U(k)$ represents a uniform distribution at [0,k]. M↓ and A↓ denote MPJPE↓ and Accel↓, respectively. ↓ indicates that SynSP++ outperforms SynSP.

Method	$N(0,100)$		$N(0,400)$		$U(50)$		$U(100)$	
	M↓	A↓	M↓	A↓	M↓	A↓	M↓	A↓
Noisy input	61.2	143.8	122.5	287.6	88.7	207.9	177.4	415.8
GFPose [34]	40.4	-	60.0	-	45.6	-	84.4	-
SmoothNet [9]	35.1	4.3	41.5	4.9	37.7	4.2	44.9	5.1
SynSP [8]	20.6	3.7	28.0	4.4	24.4	4.1	32.4	4.6
SynSP++	**19.0**↓	**3.5**↓	**26.7**↓	**4.2**↓	**23.0**↓	**3.8**↓	**31.1**↓	**4.4**↓

SynSP and SmoothNet in AIST++ and 3DPW datasets with 3D pose video-based estimator VIBE [18] and TCMR [20], respectively. We can find that the high-performance video-based estimator, TCMR, is hard to be refined. SmoothNet improves the smoothness (Accel error) by processing a longer window length than the video-based estimators. SynSP greatly increases precision and smoothness in a shorter window length with its outstanding network design, while SynSP++ can further improve performance through its novel architecture, as shown in Fig. 3. Besides, the image-based estimator, PARE [48], has excellent performance in precision, SynSP++ can also make up for its shortcomings in smoothness and outperform some video-based estimators. We can also observe the superior performance of SynSP++ on the multi-view pose estimator, *i.e.*, PPT [22]. As shown in Tab. II, we can notice that SynSP++ can further improve the precision and acceleration of PPT, which has superior performance by considering multi-view connections.

Comparison with Pose Priors Methods. To ensure a fair comparison with human pose prior models, we also conducted denoise experiments following the methodology of the SOTA work [34] among pose priors methods. Therefore, we artificially introduced uniform and Gaussian noise with varying intensities to the Human3.6M and CMU-Mocap datasets. Compared with datasets constructed by various pose estimators, artificial noise datasets have the following two characteristics: 1) By adding these irregular noises to every frame of the pose sequence, noticeable jitter in the joints occurred, increasing Accel error; 2) Artificial noise datasets can better reflect the model's performance since errors induced by artificial noises are more random.

Here, a window size of GFPose, SmoothNet, SynSP, and SynSP++ are 1, 32, 8, and 8, respectively, while GFPose, as a generative model for human pose prior task, has 1000 sampling steps, *i.e.*, 1000 times of model inference. GFPose has not presented its Accel errors, since it does not consider temporal information. As presented in Tab. III and Tab. IV, SynSP++ can effectively solve MPJPE (Precision) problems. Across noisy Human3.6M and CMU-Mocap datasets, SySP++ demonstrates excellent noise reduction capabilities and surpasses SynSP.

Discussion on Long-Window Methods. There are also some methods [25], [26] with a long-window input length, and the comparison with them is shown in Tab. V. Both DeciWatch [25] and HANet [26] employ a larger window size setting, which may potentially result in overfitting on normal-scale datasets, such as Human3.6M and 3DPW while offering an advantage on large-scale datasets like AIST++. Overall, our methods, SynSP and SynSP++, can still obtain competitive performance compared with the methods having extremely longer window lengths.

TABLE V: Comparison with long-window methods. ↓ indicates that SynSP++ outperforms SynSP. ↑ is vice versa.

Dataset	Method	WS	MPJPE	Accel
Human3.6M	FCN	1	54.55	19.17
	FCN+DeciWatch [25]	1+101	52.80	1.50
	FCN+HANet [26]	1+101	51.80	2.00
	FCN+SynSP [8]	1+8	51.36	1.02
	FCN+SynSP++	1+8	**50.44**↓	**1.00**↓
3DPW	PARE	1	79.00	25.60
	PARE+DeciWatch [25]	1+101	77.20	6.90
	PARE+HANet [26]	1+101	77.10	6.80
	PARE+SynSP [8]	1+8	**76.20**	6.16
	PARE+SynSP++	1+8	77.64↑	**5.90**↓
AIST++	SPIN	1	107.72	33.20
	SPIN+DeciWatch [25]	1+101	71.30	5.70
	SPIN+HANet [26]	1+101	**69.20**	**5.40**
	SPIN+SynSP [8]	1+8	84.63	6.08
	SPIN+SynSP++	1+8	83.11↓	6.04↓

Visualization of Output. In Fig. 8, We also present a visualization comparison between SynSP++ and SmoothNet on Human3.6M with estimator Hourglass. We can observe that SynSP++ improved the localization accuracy in several joints. This improvement comes from SynSP++ not only dealing with joint dimensions and spatial dimensions as shown in Fig. 2, but also from similar searched pose sequences from PVD.

E. Experimental Analysis.

In this section, we reintroduce the design motivations for each module of SynSP++ and demonstrate their performance enhancements for SynSP++ through ablation experiments. We mainly conduct our experiments and analysis on the AIST++ dataset with VIBE estimator for three key reasons: 1) the AIST++ dataset is large enough to showcase the model's performance; 2) 3D representation is more suitable for the PVD module; 3) Optimizing output pose sequences from VIBE, a video-based pose estimator, poses a significant challenge.

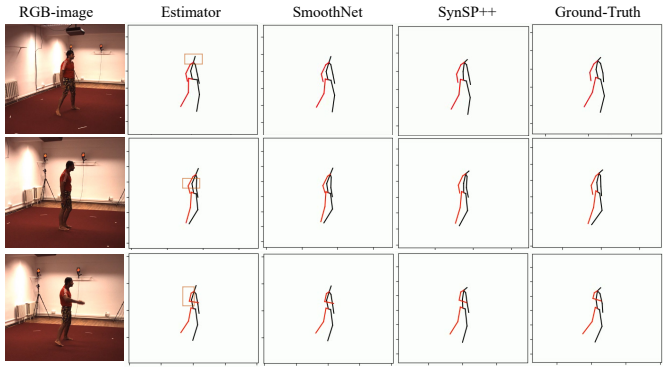

Fig. 8: Visualization comparison with SOTA method. The orange box highlights the better refinement by SynSP++.

TABLE VI: Contrast experiments of SynSP on the number of input views. † means that we randomly combine several people for input during the training process instead of using multiple views for one person. * means that in the case of †, several people with similar poses are input simultaneously during the inference process.

View Number	MPJPE	PA-MPJPE	Accel
1	51.4	40.1	1.02
2	47.2	37.6	1.02
3	46.5	37.1	1.00
4	**41.8**	**33.3**	**0.98**
4†	50.4	39.5	1.01
4*	49.7	39.1	1.01

Model Design Principle. The experiment result of SynSP [8] on Human3.6M is shown in Tab. VI; we find that the SynSP framework enables the simultaneous input of multi-view poses to generate an optimized final pose. We also conduct experiments by inputting multiple similar poses instead of multi-view poses and observe enhanced performance as illustrated in Tab. VI. This improvement served as the motivation behind the development of SynSP++.

Due to the limitation of parameter quantity, small models are unable to remember all human reasonable pose sequences. Therefore, we propose the PVD module to search for the most similar pose sequence and train the model to acquire example learning ability, similar to the in-context learning capability of Large Language Models (LLM). However, if input poses are fused with searched poses without sufficient processing, performance would decrease due to a lack of parameters. Hence, we introduce the Refine Decoder module and construct multiple decode layers to process input poses with enough parameters fully. We also generate PQE and SDE information simultaneously as inputs for the model, indicating the quality of input poses and similarity between searched poses, respectively. This allows the final Global Decoder module to adjust its attention based on these two pieces of information, optimizing low-quality parts of input poses by referencing high-similarity parts of searched poses and pose quality cues.

Ablation Experiment. To verify the rationality of the structure of SynSP++, we conduct ablation experiments on each module in SynSP++. The PVD + Global Decoder, PQE, SDE, and Refine Decoder components are excluded from SynSP++. Then, we obtain a conventional Transformer structure comprising an encoder and a decoder, serving as the comparative analysis baseline model. To ensure the representativeness of ablation experiments, all experiments are performed on the largest dataset of single-view, *i.e.*, AIST++, with 3D video-based estimator VIBE.

TABLE VII: Results of ablation study. The line without a tick represents the results of the baseline. GD serves as the Global Decoder, while RD is the Refine Decoder.

Line	PVD+GD	SDE	RD	PQE	MPJPE	Accel
1					81.58	4.60
2	√				78.93	4.59
3	√	√			77.87	4.55
4			√		77.90	4.36
5			√	√	76.17	4.35
6	√		√		74.04	4.32
7	√	√	√	√	73.10	4.29

As shown in Tab. VII, we do the ablation experiment on four modules: PVD + Global Decoder, PQE, SDE, and Refine Decoder. We interpret the data in the table according to the row number of the data: 1) Baseline with a conventional Transformer structure. 2) PVD leads to a performance increase in SynSP++. As discussed in the Model Design Principle above, PVD can provide reference information from similar poses for the model, which mainly improves the precision of our model. 3) Adding SDE based on PVD further brings performance improvement to the model because the SDE module helps enhance attention towards similar poses in the self-attention structure. 4) Refine Decoder specifically processes input poses and improves overall model performance through stacking decoder modules. 5) Adding a PQE module based on Refine Decoder enables the model to get quality information on the input poses. Then, it brings further performance improvement. 6&7) As mentioned in the Model Design Principle above, these modules collaborate, ultimately allowing the network to optimize low-quality input poses by referring to high-similarity parts of searched poses.

Analysis on PVD. To fully demonstrate the PVD module, we render 3D graphs of input poses and align searched poses under different L2 distances. Additionally, we conduct experiments on the impact of different L2 distances on model performance, which confirm that the more similar the searched poses are to the input poses, the greater their ability to repair them, especially with the objective of smoothness.

We use a visualization to compare the input poses with their corresponding searched poses. The PVD module is applied to the AIST++ dataset using the VIBE estimator, where the input poses are obtained from the validation set. The searched poses are extracted from the training set. As depicted in Fig. 9, for each input pose, its corresponding searched pose with minimal L2 distance mostly exhibits similar posture and motion trend, providing valuable reference knowledge for SynSP++. Notably, within this dataset, the average L2 distance of validation set poses in PVD is 0.18, smaller than the mean value of L2 distance illustrated in Fig. 9. This observation

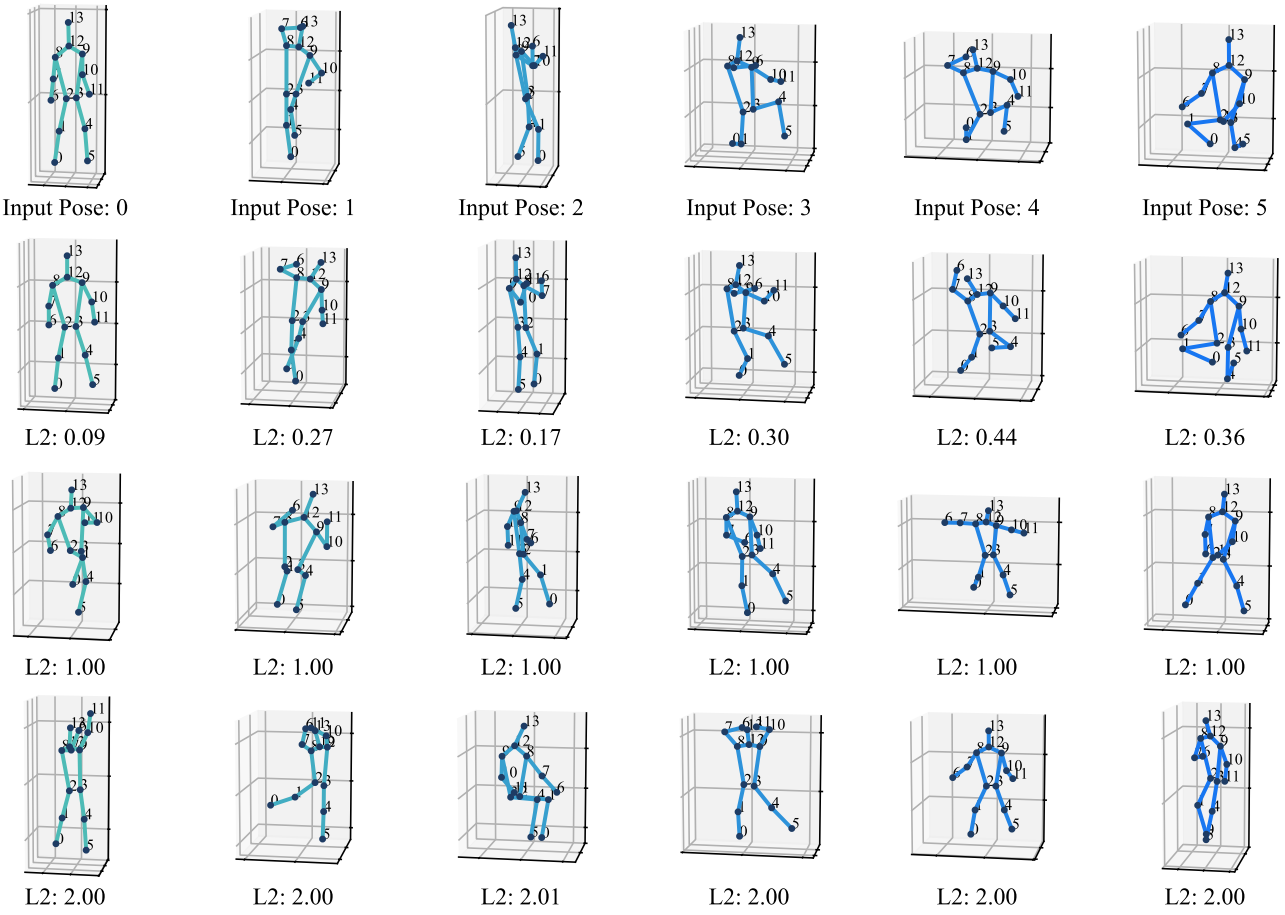

Fig. 9: Visualization images of the input and searched poses. The first row shows the input pose: n, where n is the index number. The following three rows display the most similar poses, poses with an L2 distance greater than 1, and poses with an L2 distance greater than 2, respectively. The subheadings for these three rows indicate their respective L2 distances from the input pose.

TABLE VIII: The performance of SynSP++ in different L2 distance ranges on AIST++ dataset with VIBE estimator.

L2 Distance	MPJPE	Accel
[0, $+\infty$)	73.10	4.29
[1, $+\infty$)	73.18	4.37
[2, $+\infty$)	73.24	4.44
Irrelevant Poses	74.01	4.45

suggests that most positions within the validation set can find suitable matches successfully.

To demonstrate the improvement of SynSP++ with similar searched poses, we provide the model with searched poses of varying L2 distances during the inference stage on the AIST++ dataset using estimator VIBE. The considered ranges for L2 distances are [0, $+\infty$), [1, $+\infty$), and [2, $+\infty$). Additionally, we randomly select uncorrelated poses to replace the searched poses for further comparison, and the corresponding results are presented in Table VIII. According to the results, the PVD module significantly enhances performance, particularly in acceleration. This demonstrates that even small models can acquire the ability for example learning.

Time Efficiency Analysis. We conduct latency comparison experiments on the Human3.6M dataset using FCN [40] as the 3D pose estimator with a batch size of 1416, as shown in Tab. IX, and we assume that FPS (Frame Per Second) for pose estimator is 12. Considering the practical application environment, we report not only the inference latency of the model, as shown in the 3rd column of Tab. IX, but also the latency associated with the Sliding Window Average Algorithm (SWAA), which includes the delay in waiting for pose estimators and the computation cost of SWAA (we have optimized SWAA specifically for parallel computation on GPUs). The bold cell in each row of Tab. IX represents the primary factor contributing to latency. For video processing speed of 12 frames per second for the pose estimator (based on the general inference speed of various pose estimators), a window size of 8 would require a waiting time of 8/12 = 667ms; similarly, a window size of 32 would require a waiting time of 32/12 = 2667ms. Furthermore, it takes 217ms for PVD module to output 1416 searched poses, Therefore, the total time consumed by SynSP++ pipeline is 932ms.

Overall, the main time cost primarily arises from the internal waiting latency for SmoothNet, SynSP, and SynSP++. The main time delay for the pose priors method GFPose is its 1000 sampling steps. Hence, SynSP++ is more suitable for

TABLE IX: Time efficiency analysis on SynSP++ including model inference time and time cost for sliding window. * represents the total time, including the execution time consumed by PVD, and it takes about 217ms for the PVD module to output searched poses with a batch size of 1416. Notably, PVD only costs 4ms with batch size of 1.

Methods	WS	Model (ms)	SWAA		Total (ms)
			Wait (ms)	Execution (ms)	
GFPose	1	**3772**	-	-	3772
SmoothNet	32	1.21	**2667**	802	3470
SynSP	8	11.0	**667**	0.3	678
SynSP++	8	47.6	**667**	0.3	932*

near real-time pose refinement tasks.

V. Conclusion

In this paper, we propose SynSP++, which is more general and superior to the previous work, SynSP. We mainly explore two aspects in this study: 1) we employ two adversarial objectives, *i.e.*, smoothness and precision, for the human motion refinement task to generate quality indicators in input poses and utilize it to improve model performance; 2) we extensively explore the sample learning ability of small model with a plug-in pose knowledge base, Pose Vector Database, enabling our model to reference similar examples with only a few parameters. Specifically, SynSP++ takes input poses, searched poses, and searched distances as input and outputs Position Branch and Acceleration Branch biased towards precision and smoothness, respectively. Then, SynSP++ employs the discrepancy between the two branches as a quality cue for the input poses. Finally, in Global Decoder, SynSP++ efficiently fuses the quality cue, similar searched poses, and the searched distances to refine the pose sequence further. We firmly believe that the two contributions in our work have broader implications beyond motion refinement tasks and will further explore their applications in the future.

Acknowledgment

The paper is supported by National Natural Science Foundation of China No.62102039, No.No.82274685, the Fundamental Research Funds for the Central Universities No.500422813.

References

[1] Y. Li, K. Li, S. Jiang, Z. Zhang, C. Huang, and R. Y. Da Xu, "Geometry-driven self-supervised method for 3d human pose estimation," in *AAAI*, vol. 34, no. 07, 2020, pp. 11 442–11 449.

[2] Y. Cheng, B. Yang, B. Wang, and R. T. Tan, "3d human pose estimation using spatio-temporal networks with explicit occlusion training," in *AAAI*, vol. 34, no. 07, 2020, pp. 10 631–10 638.

[3] L. Jin, X. Wang, X. Nie, W. Wang, Y. Guo, S. Yan, and J. Zhao, "Rethinking the person localization for single-stage multi-person pose estimation," *IEEE TMM*, 2023.

[4] T. Wang, L. Jin, Z. Wang, X. Fan, Y. Cheng, Y. Teng, J. Xing, and J. Zhao, "Decenternet: Bottom-up human pose estimation via decentralized pose representation," in *ACM MM*, 2023, pp. 1798–1808.

[5] W.-L. Wei, J.-C. Lin, T.-L. Liu, and H.-Y. M. Liao, "Capturing humans in motion: Temporal-attentive 3d human pose and shape estimation from monocular video," in *CVPR*, 2022, pp. 13 211–13 220.

[6] J. Wu, H. Zheng, B. Zhao, Y. Li, B. Yan, R. Liang, W. Wang, S. Zhou, G. Lin, and Y. Fu, "Ai challenger : A large-scale dataset for going deeper in image understanding," in *ICME*, 2017.

[7] J. Chu, L. Jin, X. Fan, Y. Teng, Y. Wei, Y. Fang, J. Xing, and J. Zhao, "Single-stage multi-human parsing via point sets and center-based offsets," in *ACM MM*, 2023, pp. 1863–1873.

[8] T. Wang, L. Jin, Z. Wang, J. Li, L. Li, F. Zhao, Y. Cheng, L. Yuan, L. Zhou, J. Xing *et al.*, "Synsp: Synergy of smoothness and precision in pose sequences refinement," in *CVPR*, 2024.

[9] 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 *ECCV*, 2022, pp. 625–642.

[10] G. Casiez, N. Roussel, and D. Vogel, "1€ filter: a simple speed-based low-pass filter for noisy input in interactive systems," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, 2012, pp. 2527–2530.

[11] M. Fabbri, F. Lanzi, S. Calderara, S. Alletto, and R. Cucchiara, "Compressed volumetric heatmaps for multi-person 3d pose estimation," in *CVPR*, 2020.

[12] D. Pavllo, C. Feichtenhofer, D. Grangier, and M. Auli, "3d human pose estimation in video with temporal convolutions and semi-supervised training," in *CVPR*, 2019.

[13] I. Cohen, Y. Huang, J. Chen, J. Benesty, J. Benesty, J. Chen, Y. Huang, and I. Cohen, "Pearson correlation coefficient," *Noise reduction in speech processing*, pp. 1–4, 2009.

[14] A. Newell, K. Yang, and J. Deng, "Stacked hourglass networks for human pose estimation," in *ECCV*, 2016, pp. 483–499.

[15] L. Jin, X. Wang, X. Nie, L. Liu, Y. Guo, and J. Zhao, "Grouping by center: Predicting centripetal offsets for the bottom-up human pose estimation," *IEEE TMM*, 2022.

[16] L. Jin, C. Xu, X. Wang, Y. Xiao, Y. Guo, X. Nie, and J. Zhao, "Single-stage is enough: Multi-person absolute 3d pose estimation," in *CVPR*, 2022, pp. 13 086–13 095.

[17] M. Loper, N. Mahmood, J. Romero, G. Pons-Moll, and M. J. Black, "Smpl: A skinned multi-person linear model," *ACM TOG*, vol. 34, no. 6, pp. 1–16, 2015.

[18] M. Kocabas, N. Athanasiou, and M. J. Black, "Vibe: Video inference for human body pose and shape estimation," in *CVPR*, 2020, pp. 5253–5263.

[19] N. Kolotouros, G. Pavlakos, M. J. Black, and K. Daniilidis, "Learning to reconstruct 3d human pose and shape via model-fitting in the loop," in *ICCV*, 2019, pp. 2252–2261.

[20] H. Choi, G. Moon, J. Y. Chang, and K. M. Lee, "Beyond static features for temporally consistent 3d human pose and shape from a video," in *CVPR*, 2021, pp. 1964–1973.

[21] H. Ma, L. Chen, D. Kong, Z. Wang, X. Liu, H. Tang, X. Yan, Y. Xie, S.-Y. Lin, and X. Xie, "Transfusion: Cross-view fusion with transformer for 3d human pose estimation," 2021.

[22] H. Ma, Z. Wang, Y. Chen, D. Kong, L. Chen, X. Liu, X. Yan, H. Tang, and X. Xie, "Ppt: Token-pruned pose transformer formonocular andmulti-view human pose estimation," in *ECCV*, 2022.

[23] W. H. Press and S. A. Teukolsky, "Savitzky-golay smoothing filters," *Computers in Physics*, vol. 4, no. 6, pp. 669–672, 1990.

[24] I. T. Young and L. J. Van Vliet, "Recursive implementation of the gaussian filter," *Signal processing*, vol. 44, no. 2, pp. 139–151, 1995.

[25] A. Zeng, X. Ju, L. Yang, R. Gao, X. Zhu, B. Dai, and Q. Xu, "Deciwatch: A simple baseline for 10× efficient 2d and 3d pose estimation," in *ECCV*. Springer, 2022, pp. 607–624.

[26] K.-M. Jin, B.-S. Lim, G.-H. Lee, T.-K. Kang, and S.-W. Lee, "Kinematic-aware hierarchical attention network for human pose estimation in videos," in *WCACV*, 2023, pp. 5725–5734.

[27] F. Bogo, A. Kanazawa, C. Lassner, P. Gehler, J. Romero, and M. J. Black, "Keep it smpl: Automatic estimation of 3d human pose and shape from a single image," in *ECCV*, 2016, pp. 561–578.

[28] G. Pavlakos, V. Choutas, N. Ghorbani, T. Bolkart, A. A. Osman, D. Tzionas, and M. J. Black, "Expressive body capture: 3d hands, face, and body from a single image," in *CVPR*, 2019, pp. 10 975–10 985.

[29] G. Tiwari, D. Antić, J. E. Lenssen, N. Sarafianos, T. Tung, and G. Pons-Moll, "Pose-ndf: Modeling human pose manifolds with neural distance fields," in *ECCV*, 2022, pp. 572–589.

[30] H. Y. Ling, F. Zinno, G. Cheng, and M. Van De Panne, "Character controllers using motion vaes," vol. 39, no. 4. ACM New York, NY, USA, 2020, pp. 40–1.

[31] D. Rempe, T. Birdal, A. Hertzmann, J. Yang, S. Sridhar, and L. J. Guibas, "Humor: 3d human motion model for robust pose estimation," in *ICCV*, 2021, pp. 11 488–11 499.

[32] A. Davydov, A. Remizova, V. Constantin, S. Honari, M. Salzmann, and P. Fua, "Adversarial parametric pose prior," in *CVPR*, 2022, pp. 10 997–11 005.

[33] M. Petrovich, M. J. Black, and G. Varol, "Action-conditioned 3d human motion synthesis with transformer vae," in *ICCV*, 2021, pp. 10 985–10 995.
[34] H. Ci, M. Wu, W. Zhu, X. Ma, H. Dong, F. Zhong, and Y. Wang, "Gfpose: Learning 3d human pose prior with gradient fields," 2023.
[35] J. Wang, X. Long, Y. Gao, E. Ding, and S. Wen, "Graph-pcnn: Two stage human pose estimation with graph pose refinement," in *ECCV*, 2020, pp. 492–508.
[36] G. Moon, J. Y. Chang, and K. M. Lee, "Posefix: Model-agnostic general human pose refinement network," in *CVPR*, 2019, pp. 7773–7781.
[37] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," vol. 30, 2017.
[38] J. Wang, X. Yi, R. Guo, H. Jin, P. Xu, S. Li, X. Wang, X. Guo, C. Li, X. Xu *et al.*, "Milvus: A purpose-built vector data management system," in *Proceedings of the 2021 International Conference on Management of Data*, 2021, pp. 2614–2627.
[39] NumPy. Broadcasting. (2024.06.30). [Online]. Available: https://numpy.org/doc/stable/reference/generated/numpy.broadcast.html
[40] J. Martinez, R. Hossain, J. Romero, and J. J. Little, "A simple yet effective baseline for 3d human pose estimation," in *ICCV*, 2017, pp. 2640–2649.
[41] C. Ionescu, D. Papava, V. Olaru, and C. Sminchisescu, "Human3. 6m: Large scale datasets and predictive methods for 3d human sensing in natural environments," *IEEE TPAMI*, vol. 36, no. 7, pp. 1325–1339, 2013.
[42] T. Von Marcard, R. Henschel, M. J. Black, B. Rosenhahn, and G. Pons-Moll, "Recovering accurate 3d human pose in the wild using imus and a moving camera," in *ECCV*, 2018, pp. 601–617.
[43] R. Li, S. Yang, D. A. Ross, and A. Kanazawa, "Ai choreographer: Music conditioned 3d dance generation with aist++," in *ICCV*, 2021, pp. 13 401–13 412.
[44] C. M. University. Cmu graphics lab motion capture database: http://mocap.cs.cmu.edu/. [Online]. Available: http://mocap.cs.cmu.edu/
[45] Y. Cheng, B. Wang, B. Yang, and R. T. Tan, "Monocular 3d multi-person pose estimation by integrating top-down and bottom-up networks," in *CVPR*, 2021, pp. 7649–7659.
[46] A. Benzine, F. Chabot, B. Luvison, Q. C. Pham, and C. Achard, "Pandanet: Anchor-based single-shot multi-person 3d pose estimation," in *CVPR*, 2020.
[47] Z. Wan, Z. Li, M. Tian, J. Liu, S. Yi, and H. Li, "Encoder-decoder with multi-level attention for 3d human shape and pose estimation," in *ICCV*, 2021, pp. 13 033–13 042.
[48] M. Kocabas, C.-H. P. Huang, O. Hilliges, and M. J. Black, "Pare: Part attention regressor for 3d human body estimation," in *ICCV*, 2021, pp. 11 127–11 137.

Lei Jin is currently an Associate Research Fellow with the Beijing University of Posts and Telecommunications (BUPT), Beijing, China. He graduated from Beijing University of Posts and Telecommunications, his major research areas include computer vision, data mining, pattern recognition, with in-depth research in sub-fields such as human pose estimation, human action recognition, and human parsing, with related research results published in high-level conferences and journals such as CVPR, AAAI, NIPS, IJCAI, and ACMMM, and so on.

Tao Wang is currently pursuing a doctorate in Beijing University of Posts and Telecommunications (BUPT), Beijing, China. His major research areas include human pose estimation and human motion refinement, with related research results published in high-level conferences such as CVPR and ACMMM.

Dunbo Cai is a PhD from Jilin University, expert from China Mobile, senior technology researcher in mobile cloud, research direction: Artificial Intelligence, Automatic Reasoning and Constraint Planning, Intelligent Action Planning, selected for Suzhou "Gusu Talent" Program, and member of the first session of the Quantum Computing Committee of the Chinese Electronics Society. Hosted one National Natural Science Foundation Youth Project, participated in four National Natural Science Foundation Projects, and one Major Science and Technology Project in the 14th Five Year Plan. Published over 30 international journal and conference papers in fields such as quantum computing, artificial intelligence, and network communication.

Ling Qian is the PhD from Tsinghua University. He has hosted 5 National Natural Science Foundation projects, 1 National Key R&D Program of the Ministry of Science and Technology, 4 strategic emerging industry projects of the State owned Assets Supervision and Administration Commission exceeding 60 million yuan, and more than 10 provincial and ministerial level key projects. Successfully self-developed China Mobile's new generation cloud computing network - Tianchi Network. Designed a distributed routing protocol specifically for cloud data centers. The relevant research work has been published in INFOCOM ICC, IWQoS, ISCC, Top academic conferences and journals such as Journal of Computer Science, Computer Research and Development, Knowledge Based Systems, ICASP, etc.

Junliang Xing is a Professor with Tsinghua University and the recipient of the National Science Fund for Excellent Young Scholar. He obtained his dual bachelor's degrees in Computer Science and Mathematics at Xi'an Jiaotong University in 2007 and his doctorate in Computer Science in 2012. Then he worked in the National Laboratory of Pattern Recognition, Institute of Automation, Chinese Academy of Sciences as an assistant researcher, associated researcher, and researcher in 2012, 2015, and 2018, respectively. His research interests are human-computer interactive learning, computer gaming, and computer vision. He has published more than 100 peer-reviewed papers in international conferences and journals and got more than 13,000 citations from Google Scholar.

Jian Zhao is leader of Evolutionary Vision+x Oriented Learning (EVOL) Lab and Young Scientist at Institute of AI (TeleAI), China Telecom, and Researcher and Ph.D. Supervisor at Northwestern Polytechnical University (NWPU). He received his Ph.D. degree from National University of Singapore (NUS) in 2019 under the supervision of Assist. Prof. Jiashi Feng and Assoc. Prof. Shuicheng Yan. He is the SAC of VALSE, the committee member of CSIG-BVD, and the member of the board of directors of BSIG. He has over 40 influential papers on human-centric image understanding, and accolades including the Lee Hwee Kuan Gold and ACM MM Best Student Paper awards. Additionally, he has organized key workshops and challenges at CVPR, ECCV, and other venues.

Xuelong Li is the Chief Technology Officer (CTO) of China Telecom since 2023. Before that, he was a full professor at The Northwestern Polytechnical University (2018-2023). Throughout his career, he has been recognized with various prestigious fellowships from organizations like IEEE, ACM, and AAAI, among others. Additionally, he has made significant contributions to the academic community through his leadership roles in various editorial and selection committees, reflecting his influence in the fields of artificial intelligence and technology.