SynSP: Synergy of Smoothness and Precision in Pose Sequences Refinement

Tao Wang*[1], Lei Jin*†[1], Zheng Wang[2], Jianshu Li[3], Liang Li[4],
Fang Zhao[5], Yu Cheng[6], Li Yuan[7], Li Zhou[7], Junliang Xing[8], Jian Zhao[9,10]

[1]Beijing University of Posts and Telecommunications, [2]Wuhan University, [3]Ant Group,
[4]Institute of computing technology Chinese Academy of Sciences, [5]Nanjing University, [6]National University of Singapore,
[7]Peking University, [8]Tsinghua University, [9]China Telecom Institute of AI, [10]Northwestern Polytechnical University

wangtao@bupt.edu.cn, jinlei@bupt.edu.cn, wangzwhu@whu.edu.cn, jianshu@u.NUS.edu,
liang.li@ict.ac.cn, zhaofang0627@gmail.com, e0321276@u.nus.edu, yuanli-ece@pku.edu.cn,
zhouli2025@126.com, jlxing@tsinghua.edu.cn, jian_zhao@nwpu.edu.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. Prior investigations have primarily concentrated on striking a balance between the two objectives, i.e., smoothness and precision, while optimizing the predicted pose sequences. However, it has come to our attention that the tension between these two objectives can provide additional quality cues about the predicted pose sequences. These cues, in turn, are able to aid the network in optimizing lower-quality poses. To leverage this quality information, we propose a motion refinement network, termed **SynSP**, to achieve a **Syn**ergy of **S**moothness and **P**recision in the sequence refinement tasks. Moreover, SynSP can also address multi-view poses of one person simultaneously, fixing inaccuracies in predicted poses through heightened attention to similar poses from other views, thereby amplifying the resultant quality cues and overall performance. Compared with previous methods, SynSP benefits from both pose quality and multi-view information with a much shorter input sequence length, achieving state-of-the-art results among four challenging datasets involving 2D, 3D, and SMPL pose representations in both single-view and multi-view scenes. Github code: https://github.com/InvertedForest/SynSP.*

1. Introduction

Human pose estimation [4, 16, 21, 37] has been widely used in automatic driving, human-computer interaction, motion analysis, *etc*. Although current methods have achieved outstanding performance, both image-based and video-based pose estimators still sometimes exhibit noticeable errors in

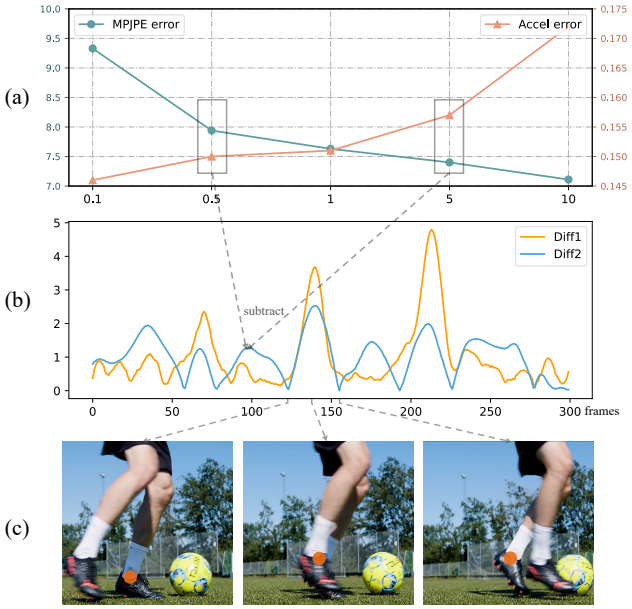


Figure 1. Illustration of our motivation. (a): The x-axis represents the ratio of weights for Precision and Smoothness loss in the training process of SynSP, while the y-axis represents the value of MPJPE (Precision) and Acceleration error (Smoothness). These two errors exhibit a conflicting relationship. (b): Diff1 denotes the difference between the predicted sequence (from pose estimator) and the gound-truth sequence, and Diff2 denotes the difference between two output sequences from training SynSP 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. † Lei Jin is corresponding author.

predicting pose sequences. The pose sequences generated by image-based pose estimators often show significant jitters due to a lack of inter-frame correlation, while video-based pose estimators alleviate jitters and improve accuracy by leveraging temporal information of input videos. However, training video-based pose estimators requires substantial amounts of video data and computational resources [7, 38, 39].

Consequently, researchers propose motion refinement methods [3, 42] 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 Mean Per Joint Position Error (MPJPE) [11, 29]. **Smoothness** objective does not strive to make the output sequences as smooth as possible but rather to align the acceleration of the output 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 lead to errors in the other greatly increased, as shown in Fig. 1.

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 SynSP training process, there is a noticeable rise in Acceleration error (Smoothness) while MPJPE (Precision) decreases, which is a common phenomenon in previous methods [42]. However, we note that the quality of the predicted sequences from estimator is related to the difference of outputs from SynSP 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 [9] 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.

In this paper, we propose SynSP to utilize this quality information. Specifically, we employ a transformer network and encode the quality information by Pose Quality Encoding (PQE) module to guide the network's attention towards low-quality poses in sequences. Additionally, with the support of multi-view dataset: Human3.6M, we further find that multi-view information is effective to enhance the reliability of PQE. Therefore, we introduce the Pose Similarity Encoding (PSE) module to exploit the multi-view information by guide the network's attention towards similar poses from multi-view.

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 frame in the sequences.
- We further observe that similar poses can effectively rectify erroneous poses. Consequently, we propose Pose Similarity Encoding to encode pose similarity information and refine poses of multiple views simultaneously.
- 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 method SmoothNet, SynSP's MPJPE improves by **16.9%** on Human3.6M dataset with 3D pose estimator FCN, while reducing input sequence length to **25%** and minimizing time delay to just **17%**.

2. Related Work

Human Pose Estimation. Human pose estimation task [14–16, 27] is to extract 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 [23] is a skinned vertex based model that represents a wide variety of body shapes in natural human poses, while needs more parameters to represent.

Human pose estimators can be divided into image-based methods [27] and video-based methods [17]. Pose sequences from image-based methods often show a lot of jitters when these methods are applied to videos, while video-based methods require more data and computing resources for training. 2D pose estimator, Hourglass [27] 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 [6] and VIBE [17], can generate smooth pose sequences with temporal information. We conduct the experiments based on pose estimators mentioned above.

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 are closer to those of the ground-truth sequences. The motion refinement methods can be divided to

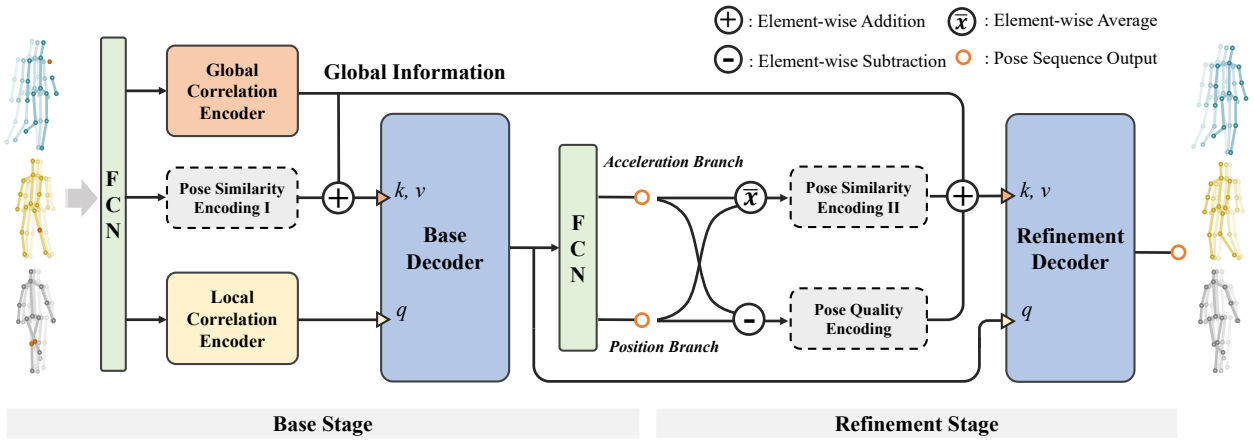


Figure 2. The architecture of SynSP. FCN represents fully-connected network. In the **base stage**, with pose sequences of multiple views as input, SynSP first encodes these sequences individually in the Local Correlation Encoder, and merge multiple view's information in the Global Correlation Encoder with the Pose Similarity Encoding as their weight. After obtaining the individual embedding as the query, and incorporating the biased global information as the key and value, we get two outputs through the decoder module. In the **refinement stage**, we utilize these two detached outputs to generate Pose Quality Encoding and more precise Pose Similarity Encoding. By reusing embedding generated earlier, we achieve superior results using only one decoder with these encodings.

low-pass filters and deep learning based methods. The traditional low-pass filters, *e.g.*, Savitzky-Golay [30], Gaussian1d [40], and One-Euro [3], 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 low-pass filter does look smoother, it is difficult to ensure precision at the same time. Deep learning based method, *i.e.*, SmoothNet [42], takes the two objectives into account and aims to improve both precision and smoothness. Our SynSP is also based on deep learning and can achieve better results by considering the pose sequence quality and multi-view correlation information.

Additionally, there are also several human pose prior methods [2, 22, 28, 32], 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 task of pose generation/completion/refinement. HuMoR [31] introduces a novel conditional VAE which enables expressive and general motion reconstruction and generation; GAN-based method [10] proposes a simple yet effective prior for SMPL model to bound it to realistic human poses; GFPose [8] employs a novel score-based generative framework to model plausible 3D human poses.

3. Method

3.1. Overview

Task Description. Given predicted pose sequences generated by human pose estimators [16], motion refinement methods aim to fit these predicted sequences to the ground-truth sequences, making these prediction more precise and smooth. Usually, the predicted pose sequence $\mathbf{P}$ has a shape of $\mathbb{R}^{V\times T\times(J\times D)}$, where V stands for the number of views for one individual; T refers to the frame number of these predicted pose sequences; J is the number of joints associated with datasets; D denotes the spatial dimensions of these joints. We use $P_{(v,t,j)}$ to represent the j_{th} joint's position at t_{th} frame from v_{th} view throughout the paper. The spatial dimension is included in the joint dimension, ignoring the spatial dimensions to simplify it.

Overall Architecture. We proposed SynSP to use this quality information from the certain tension relationship between the two objectives, *i.e.*, smoothness and precision. We also introduce multi-view information fusion as an **optional** enhancement for SynSP. As illustrated in Fig. 2, SynSP is mainly based on the transformer structure [34] and composed of two stages. Firstly, under the guidance of Pose Similarity Encoding I, Base Decoder can utilize both individual and global information to generate two branches, which are respectively biased towards smoothness and precision. Next, these two branches can generate Pose Quality Encoding, and a more precise Pose Similarity Encoding II, which further helps Refinement Decoder generate refined results.

3.2. Base Stage

In the base stage, SynSP takes predicted sequences from pose estimators as input and generates two sequences that focus on smoothness and precision, respectively.

Pose Similarity Encoding. Similarity for poses from multi-view is effective in enhancing the performance of SynSP.

On one hand, similar poses from different views can complement each other and improve the robustness of joint position. On the other hand, this encoding minimizes the network's attention on poses with significant distortions to reduce their impact for the refinement task.

Pose Similarity Encoding (PSE) is calculated as follows:

$$P^{rel}_{(v,t,j)} = \hat{P}_{(v,t,j)} - P_{(v,t,c)},$$
$$PSE_{(v_1,v_2,t)} = exp(-\sum_{j=1}^{J}(P^{rel}_{(v_1,t,j)} - P^{rel}_{(v_2,t,j)})^2), \quad (1)$$

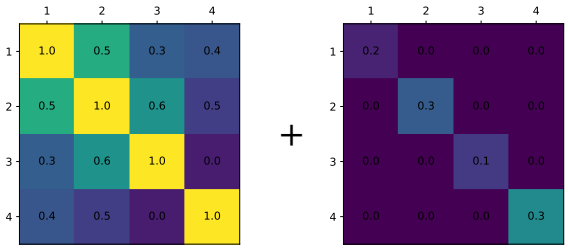
where c represents the index of the "pelvis" keypoint in the joint list. v_1 and v_2 are used only for distinction, and both have the same domain as v. Overall, PSE has a shape of $\mathbb{R}^{V\times V\times T}$, and we present a visualization example of PSE for one frame in Fig. 3.

Global Correlation Encoder. In multi-view scenes, the correlation information between multiple views can help SynSP to deduce the location of the joint with jitters globally. Global Correlation Encoder (GCE) module is introduced to produce the keys and values required by the Base Decoder with the global correlation information between multiple views. First, the predicted sequences **P** are projected to embedding **E** through a full connected layer along the joint dimension.

Here, we use $E_{(v,t,e)}$ to denote an item of projected embedding **E**, where e refers to the index of the embedding dimension. Then, the view (v) and the frame (t) dimensions of **E** are merged, so that pose information from multiple views and temporal information can be sensed by the encoder simultaneously. Afterwards, $E_{((v,t),e)}$ is sent to a encoder module of transformer [34] to interact pose sequences from multiple views and get the global information $E^g_{((v,t),e)}$. Then this information is added with $PSE_{(v_1,v_2,t)}$ to adjust $v_{1\ th}$ view's attention to other views' information by broadcast operation: $E^g_{((v_2,t),e)} + PSE_{(v_1,v_2,t)} = E^g_{(1,(v_2,t),e)} + PSE_{(v_1,(v_2,t),1)} = E^g_{(v_1,(v_2,t),e)}$, which is the final global information from GCE module.

Local Correlation Encoder. Local Correlation Encoder (LCE) module focuses on the temporal dimension with single-view and produces the query required by the Base Decoder. LCE reuses the projected embedding **E** from the full connected layer mentioned in GCE as input, and directly sends it to another standard encoder module of transformer [34] to obtain local information in the frame (t) dimension (take the view (v) dimension as input batch). Finally, for $v_{1\ th}$ view, t_{th} frame, e_{th} item, LCE produces the local information $E^l_{(v_1,t,e)}$ with exploring the correlation of the time dimension.

Base Decoder. At the end of base stage, we stack 5 standard decoding layers of transformer [34] into Base Decoder. The Base Decoder takes the local information $E^l_{(v_1,t,e)}$ as the query, to interact with global information $E^g_{(v_1,(v_2,t),e)}$,



Pose Similarity Encoding Pose Quality Encoding

Figure 3. Visualization of the two encodings of one frame. *PSE*: the diagonal has a constant value of 1. *PQE*: the value outside the diagonal is constant 0. In this way, the addition of the two encodings cannot affect each other.

which serves as keys and values for the Base Decoder. At the last layer of Base Decoder, we use two decoder layers to generate two branches biased towards smoothness and precision respectively at the same time.

$$K, V = E^g_{(v_1,(v_2,t),e)},\ Q^0 = E^l_{(v_1,t,e)},$$
$$Q^{d_{th}}_{(v_1,t,e)} = \textbf{DLayer}^{d_{th}}(Q^{(d-1)_{th}}, K, V), d \in \{1,2,3\},$$
$$P^{pos}_{(v_1,t,j)} = \mathrm{FCN}(\textbf{DLayer}^{4_{\mathrm{th}}}(\mathrm{Q}^{3_{\mathrm{th}}}, \mathrm{K}, \mathrm{V})),$$
$$P^{acc}_{(v_1,t,j)} = \mathrm{FCN}(\textbf{DLayer}^{5_{\mathrm{th}}}(\mathrm{Q}^{3_{\mathrm{th}}}, \mathrm{K}, \mathrm{V})), \quad (2)$$

where Q, K, and V denote the query, key, and value for Base Decoder respectively, $\textbf{DLayer}^{d_{th}}$ denotes the d_{th} decoder layer, $\mathrm{FCN}(\cdot)$ is fully connected network. In this way, SynSP can generate two pose squences 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. (4), respectively. Next, we can also further optimize these sequences in the refinement stage, with a small amount of calculation and parameters as the cost.

3.3. Refinement Stage

In the refinement stage, we aim to seek a better way to fuse Acceleration Branch and Position Branch, which concentrate on different objectives.

Acceleration and Position Branches. The output sequences from Acceleration Branch and Position Branch are detached from computation graph for a stabler training process as the predictions from base stage are independent of the refinement stage. These outputs can not only generate more accurate Pose Similarity Encoding, but also provide the quality information about these pose sequences.

Pose Quality Encoding. We have empirically observed that when outputs from the two branches show a greater discrepancy, the prediction quality is poorer. Specifically, we computed the Pearson Correlation Coefficient between the prediction quality and the difference between two outputs is about 0.48 (more details in the supplementary materials),

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. Each item of PQE is calculated as follows:

$$
\begin{aligned}
P^{dis}_{(v_1,v_2,t,j)} &= |P^{acc}_{(v_1,t,j)} - P^{pos}_{(v_2,t,j)}|, \\
PQE'_{(v_1,v_2,t)} &= \frac{1}{J}\sum_{j=1}^{J} exp(-\frac{P^{dis}_{(v_1,v_2,t,j)}}{\max\limits_{j\in J} P^{dis}_{(v_1,v_2,t,j)} + \sigma}), \\
PQE_{(v_1,v_2,t)} &= \begin{cases} PQE'_{(v_1,v_2,t)}, & v_1 = v_2; \\ 0, & v_1 \neq v_2. \end{cases}
\end{aligned} \tag{3}
$$

Here, we calculate PQE by the distance P^{dis} between the outputs P^{acc} and P^{pos}, which are generated by Acceleration Branch and Position Branch, respectively. To ensure that extreme values do not interfere with PQE, P^{dis} is 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.

Unlike PSE, which is calculated based on correlations among multiple views on one person, PQE is computed by P^{dis} for a single view across multiple frames. Since PQE and PSE have distinct considerations regarding multiple views versus single view, they can be combined through addition rather than concatenation without mutual influence as illustrated in Fig. 3, thereby enhancing the computational efficiency of SynSP.

Refinement Decoder. The output sequences of the two branches exhibit enhanced accuracy and smoothness compared to the input sequences. A straightforward approach would involve averaging these outputs and replicating the base stage pipeline once again. However, our experimental findings indicate that this approach does not yield significant benefits but doubles the computational cost. Please refer to supplementary material for details. This may be attributed to the fact that while the optimized output is indeed smooth and precise, it lacks crucial positional information from the original jitter input, rendering optimization of an already optimized output meaningless.

To circumvent this issue and maintain a lightweight model, we reuse the embedding from the Global Correlation Encoder and Base Decoder. The former embedding retains the original position information, while the latter embedding contains optimization information about the output. In this way, we can produce superior outcomes by using just one additional decoder. We stack 4 standard decoding layers of transformer [34] in Refinement Decoder, and the inner calculation of Refinement Decoder is similar with Base Decoder. As shown in Tab. 5, this approach effectively improves the positional accuracy of pose sequences.

Analysis on the Refinement Stage. In the refinement stage, the difference between the position and acceleration branches serves as a guiding factor for the quality of predicted sequence. Next, within the refinement decoder, greater emphasis is placed on frames with lower pose quality to facilitate improvements in the final performance. Regarding why this difference is associated with pose quality, we posit that the two branches are easy to remain consistent under the normal circumstances, nevertheless, in challenging scenarios such as occlusion or sudden movement, it is more likely for the two branches to exhibit dissimilar behavior. Whether in pose estimator or motion refinement model, pose quality is is obviously related with scenarios.

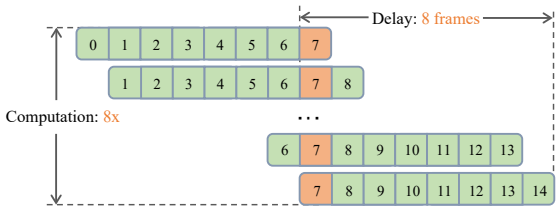


Figure 4. Illustration of Sliding Window Average Algorithm (SWAA), which is widely used by motion refinement works. This figure proves that the amount of computation and latency is proportional to the window length.

3.4. Training and Inference

Different from previous work, we only use the window length T=8 for training and inference. Compared with the window length of 32, this setting reduces the time delay of $(32-8)/12 = 2$s while the inference speed of pose estimator is 12fps, and have $8/32 = 0.25$ times the previous amount of computation as shown in Fig. 4.

Training. The losses of the three outputs, *i.e.*, Acceleration Branch output, Position Branch output, and final output, share the same calculation method. We have two losses, $\mathcal{L}_{acc}$ and $\mathcal{L}_{pos}$, for each one of the three outputs:

$$
\begin{aligned}
A_{(v,t,j)} &= (P_{(v,t+2,j)} - P_{(v,t+1,j)}) - (P_{(v,t+1,j)} - P_{(v,t,j)}), \\
\mathcal{L}_{\text{pos}} &= \frac{1}{V\cdot T\cdot J}\sum_{v=1}^{V}\sum_{t=1}^{T}\sum_{j=1}^{J}|P_{(v,t,j)} - \hat{P}_{(v,t,j)}|, \\
\mathcal{L}_{\text{acc}} &= \frac{1}{V\cdot T\cdot J}\sum_{v=1}^{V}\sum_{t=1}^{T}\sum_{j=1}^{J}|A_{(v,t,j)} - \hat{A}_{(v,t,j)}|, \\
\mathcal{L}_{\text{total}} &= \mathcal{L}_{\text{pos}} + \lambda\mathcal{L}_{\text{acc}}.
\end{aligned} \tag{4}
$$

In our experiments, λ is set as 0, 1, and 0.5 for Position Branch, Acceleration Branch, and the final output, respectively, which is a trade-off between smoothness and precision. For all branches, $\mathcal{L}_{\text{pos}}$ is crucial to ensuring training stability. Furthermore, to balance $\mathcal{L}_{\text{pos}}$ and $\mathcal{L}_{\text{acc}}$, the parameter λ should be set as 0.5, determined by the absolute value of each element. Thus, for the final output, we have set λ to 0.5, whereas in the Acceleration branch, λ is assigned as 1 to prioritize smoothness.

Inference. During inference, we obtain outputs from the refinement stage. Then the outputs go through a

Table 1. Comparison with related works on Human3.6M, AIST++ datasets with 2D, 3D, and SMPL human pose estimator Hourglass [27], FCN [26], and SPIN [19], respectively. WS denotes the windows size in training, inference, and post-process. * indicates that SynSP is trained and tested in multi-view scenes.

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 [3]	1	10.69	7.98	0.34	55.20	42.73	3.80	108.97	75.27	14.70
Gaussian1d [40]	32	9.37	7.56	0.51	53.67	41.60	2.43	104.84	72.18	10.05
Savitzky-Golay [30]	32	9.35	7.55	0.17	53.48	41.19	1.34	104.58	72.30	6.07
SmoothNet [42]	32	9.25	7.57	**0.15**	52.72	40.92	1.03	103.00	71.19	**5.72**
SynSP	8	8.13	6.09	**0.15**	51.36	40.13	1.02	**84.63**	**59.02**	6.08
SynSP*	8	**7.62**	**5.64**	**0.15**	**41.78**	**33.32**	**0.98**	-	-	-

post-processing called sliding window average algorithm adopted by most works as shown in Fig. 4 (we have accelerated it by parallel computing to 1% of the original latency, refer to supplementary material for more details). It can be seen that the delay and computation needs for model and post-processing becomes smaller with shorter window length.

Single-View Scenes. While $V = 1$ in the equations above, SynSP can be directly applied to adapt single-view scenes. In single-view scenes, *PSE* module is no longer needed. Since multi-view scenes are not considered by current related works and not our main motivation for the proposed SynSP, we conduct sufficient single-view experiments for a comprehensive comparison.

4. Experiments

4.1. Experiment Settings

Dataset. To demonstrate the generalizability and efficiency of our method, we conduct experiments of SynSP in three human pose representations of 2D, 3D, and SMPL for single-view and multi-view scenes on Human3.6M [12], AIST++ [20], 3DPW [35], and CMU-Mocap [33] datasets. For the fairness of the comparison, we use these datasets made by SmoothNet [42] with kinds of pose estimators in the experiments. To compare with pose prior models, we also follow GFPose [8] to add the same degree of Gaussian noise or uniformly distributed noise to Human3.6M and CMU-Mocap datasets as input for the pose denoise task.

Metrics. Following the related works [42], we use Accel error, MPJPE, and PA-MPJPE (Procrustes-aligned MPJPE) [1, 5] 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.

Implementation Details. SynSP consists of two 4-layer encoder modules (GCE, LCE), a 5-layer decoder module (Base Decoder), and a 4-layer decoder module (Refinement Decoder). We initialize the learning rate as 0.001 and decay it by a factor of 0.95 per epoch. SynSP is trained for 70 epochs with Adam optimizer, requiring approximately two hours and 7GB of GPU memory with a batch size of 4096. 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.

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

In our experiments, we first demonstrate that SynSP consistently outperforms previous methods in both single-view and multi-view scenes with several pose estimators. Then we compare with pose prior models on the pose denoise task by smoothing the pose sequences with artificial noisy. (There are also some methods[13, 41] with a super-long window length, and the comparison with them is shown in the supplementary materials.)

Table 2. Comparison of various video-based pose estimators. + represents the combination of estimators and motion refinement methods. * indicates multi-view inputs.

Dataset	Method	WS	MPJPE↓	Accel↓
3DPW	MAED [36]	16	79.00	-
	MPS-Net [38]	16	84.30	-
	TCMR [6]	16	86.46	6.75
	TCMR+SmoothNet [42]	16+32	86.50	6.00
	TCMR+SynSP	16+8	86.10	**5.90**
	PARE [18]	1	79.00	25.60
	PARE+SmoothNet [42]	1+32	78.10	5.91
	PARE+SynSP	1+8	**76.20**	6.16
AIST++	VIBE [17]	16	106.9	31.60
	VIBE+SmoothNet [42]	16+32	97.47	**4.15**
	VIBE+SynSP	16+8	**77.00**	4.32
H36M	TransFusion* [24]	1	25.52	-
	PPT* [25]	1	25.16	11.60
	PPT*+SmoothNet [42]	1+32	23.97	1.31
	PPT*+SynSP	1+8	**21.51**	**1.10**

Table 3. 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.

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 [8]	42.8	-	64.6	-	50.9	-	89.4	-
SmoothNet [42]	42.4	10.7	57.9	14.2	49.4	**10.2**	74.3	16.3
SynSP	**37.6**	**8.8**	**56.1**	**13.0**	**45.6**	10.9	**69.4**	**15.7**

Table 4. 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.

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 [8]	40.4	-	60.0	-	45.6	-	84.4	-
SmoothNet [42]	35.1	4.3	41.5	4.9	37.7	4.2	44.9	5.1
SynSP	**20.6**	**3.7**	**28.0**	**4.4**	**24.4**	**4.1**	**32.4**	**4.6**

2D Representation. In Tab. 1, we compare SynSP with other motion refinement methods in Human3.6M dataset with 2D pose estimator Hourglass [27]. With the setting of 8 window length, our SynSP outperforms all the methods while it is about 2 seconds faster (except the One-Euro) according to Fig. 4. Compared with SmoothNet, which is trained on multiple datasets, *i.e.*, AIST-VIBE-3D, 3DPW-SPIN-3D, and Human3.6M-FCN-3D, SynSP can still outperforms it by 16.9% for MPJPE and 25.2% for PA-MPJPE with smaller window size. According to the ablation experiment results in Sec. 4.3, the significant accuracy improvement of SynSP is attributed to the mutual patching between the pose quality information (PQE) and multi-view information (PSE).

3D Representation. In Tab. 1, we also compare SynSP with other motion refinement methods in Human3.6M dataset with 3D pose estimator FCN [26]. Similar with the comparison in 2D Representation, we can still outperforms SmoothNet by 20.8% for MPJPE and 18.6% for PA-MPJPE in Human3.6M dataset. There is also a decrease of 4.9% for the Accel error. Furthermore, SynSP obtains these results only with 8 window size, while SmoothNet and most filters need 32 frames.

SMPL Representation. Then we compare SynSP with other motion refinement methods in AIST++ dataset with SMPL pose estimator SPIN as shown in Tab. 1. SynSP shows superior performance in SMPL representation.

Comparison with Video-based Pose Estimators. The video-based estimators already include temporal information. Compared with the image-based estimators, motion refinement methods have difficult to optimize the sequences from video-based estimators. In Tab. 2, we compare SynSP with SOTA methods and SmoothNet in AIST++ and 3DPW datasets with 3D pose video-based estimator VIBE [17] and TCMR [6], respectively. We can find that the high performance video-based estimators, TCMR, is hard to be refined. SmoothNet improves the smoothness 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 architecture as shown in Fig. 2. The image-based estimator, PARE [18], has excellent performance in precision, SynSP can also make up for its shortcomings in smoothness and outperform some video-based estimators.

Comparison with Multi-view Pose Estimators. We further verify the performance of SynSP with the multi-view pose estimator, *i.e.*, PPT [25]. As shown in Tab. 2, we can notice that SynSP can further improve the precision of PPT, which has superiority performance by considering multi-view connections. We also present the result of other multi-view methods such as TransFusion for comparison. This experiment results further proves the multi-view refinement ability of PSE for video-based estimators.

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 [8] among pose priors methods. Here, window size of GFPose, SmoothNet, and SynSP are 1, 32, 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 take temporal information into account. As presented in Tab. 3 and Tab. 4, SynSP can effectively solve MPJPE (Precision) problems. Across noisy Human3.6M and CMU-Mocap datasets, SynSP can improve the pose quality by about 54.5% in terms of MPJPE and 96.0% in terms of Accel errors for Gaussian noise with a variance of 100. For Gaussian noise with a large variance of 400, SynSP can improve the pose quality by about 67.2% in terms of MPJPE and 97.2% int terms of Accel errors. We also observe consistent improvement in uniform noise as shown in Tab. 4.

4.3. Ablation Study.

To ensure representativeness of ablation experiments, all experiments were performed on the largest dataset, Human3.6M, with 3D estimator FCN.

Modules of SynSP. In order to verify the rationality of the structure of SynSP, we conducted ablation experiments on each module in SynSP. In Tab. 5, the line without a tick represents the results of baseline, which is the network of base stage without PSE I. And this network surpasses Smooth-

Table 5. Ablation study on PSE I, RS (Refinement Stage), PSE II, and PQE modules. * denotes directly using the output from base stage as input of refinement stage to re-optimize these sequnence.

PSE I	RS	PSE II	PQE	MPJPE	Accel
				51.2	1.09
√				48.7	1.07
	√*			51.4	1.08
	√			49.6	1.07
√	√			46.7	1.05
√	√	√		46.4	1.04
√	√		√	42.5	0.99
√	√	√	√	41.8	0.98

Net in precision by 2.8%. PSE I, RS, PSE II, and PQE all have gains in results, and the combination of PSE I, RS, and PQE has brought greater benefits. We believe that this benefit mainly comes from the combination of SynSP with global information PSE I, which can help generate more accurate quality information according to the poses from other views. Besides, utilizing the difference between acceleration and Position Branch improves both smoothness and precision of final prediction. It is worth noting that using the average value of the two branches as the input RS does not bring reasonable gains. We suspect that the output of base stage loses the original predicted position information, which affects the accuracy and makes the refinement stage useless compared with the baseline.

Time Efficiency Analysis. We conducted latency experiments on the Human3.6M dataset using FCN [26] as the 3D pose estimator with a batch size of 1416, as shown in Tab. 6. Considering the practical application environment, we report not only the inference latency of the model, as shown in the 3rd column of Tab. 6, 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. 6 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, the model inference time for SynSP is only about 10ms. Overall, the main time cost primarily arises from the internal waiting latency for SmoothNet and SynSP. For the pose priors method GFPose, the main time delay is its 1000 sampling steps. Hence, SynSP is more suitable for near real-time pose refinement task.

Analysis on Number of Views. As shown in Tab. 7, as more poses from different views are input at the same time during the training process and inference process, the performance of SynSP is more efficient. It is justifiable due to the introduction of more global information. If number of views is larger than 4, we believe that the performance of our model can be further promoted with only a marginal increase in model complexity.

Table 6. Time efficiency analysis on SynSP including model inference time and time cost for sliding window. Besides, we assume that FPS (Frame Per Second) for pose estimator is 12.

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	12.3	**667**	0.3	680

Table 7. Contrast experiments on the number of input views. † means that we randomly combine several people together 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 at the same time 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

5. Conclusion

This paper proposes SynSP, a motion refinement method that utilize the tension relationship between smoothness and precision through synergistic optimization. Specifically, SynSP outputs Position Branch and Acceleration Branch biased towards precision and smoothness, respectively. Then we employ the discrepancy between the two branches as a quality cue for the predicted sequence, which can be utilized in subsequent stages to further refine the pose sequence. In addition, we introduce Pose Similarity Encoding, achieving simultaneous refinement of multi-view poses and performance enhancement.

Acknowledgment

The paper is supported by National Key R & D Program of China (No. 2021YFB3300100), Natural Science Foundation of China No.62102039, the Fundamental Research Funds for the Central Universities No.500422813, and Natural Science Foundation of China No.62322211.

References

[1] 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. 6

[2] Federica Bogo, Angjoo Kanazawa, Christoph Lassner, Peter Gehler, Javier Romero, and Michael J Black. Keep it smpl: Automatic estimation of 3d human pose and shape from a single image. In *ECCV*, pages 561–578, 2016. 3

[3] Géry Casiez, Nicolas Roussel, and Daniel 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*, pages 2527–2530, 2012. 2, 3, 6

[4] Yu Cheng, Bo Yang, Bo Wang, and Robby T Tan. 3d human pose estimation using spatio-temporal networks with explicit occlusion training. In *AAAI*, pages 10631–10638, 2020. 1

[5] Yu Cheng, Bo Wang, Bo Yang, and Robby T. Tan. Monocular 3d multi-person pose estimation by integrating top-down and bottom-up networks. In *CVPR*, pages 7649–7659, 2021. 6

[6] Hongsuk Choi, Gyeongsik Moon, Ju Yong Chang, and Kyoung Mu Lee. Beyond static features for temporally consistent 3d human pose and shape from a video. In *CVPR*, pages 1964–1973, 2021. 2, 6, 7

[7] Jiaming Chu, Lei Jin, Xiaojin Fan, Yinglei Teng, Yunchao Wei, Yuqiang Fang, Junliang Xing, and Jian Zhao. Single-stage multi-human parsing via point sets and center-based offsets. In *ACM MM*, pages 1863–1873, 2023. 2

[8] Hai Ci, Mingdong Wu, Wentao Zhu, Xiaoxuan Ma, Hao Dong, Fangwei Zhong, and Yizhou Wang. Gfpose: Learning 3d human pose prior with gradient fields. *CVPR*, 2023. 3, 6, 7

[9] Israel Cohen, Yiteng Huang, Jingdong Chen, Jacob Benesty, Jacob Benesty, Jingdong Chen, Yiteng Huang, and Israel Cohen. Pearson correlation coefficient. *Noise reduction in speech processing*, pages 1–4, 2009. 2

[10] Andrey Davydov, Anastasia Remizova, Victor Constantin, Sina Honari, Mathieu Salzmann, and Pascal Fua. Adversarial parametric pose prior. In *CVPR*, pages 10997–11005, 2022. 3

[11] Matteo Fabbri, Fabio Lanzi, Simone Calderara, Stefano Alletto, and Rita Cucchiara. Compressed volumetric heatmaps for multi-person 3d pose estimation. In *CVPR*, 2020. 2

[12] Catalin Ionescu, Dragos Papava, Vlad Olaru, and Cristian Sminchisescu. Human3. 6m: Large scale datasets and predictive methods for 3d human sensing in natural environments. *IEEE TPAMI*, 36(7):1325–1339, 2013. 6

[13] Kyung-Min Jin, Byoung-Sung Lim, Gun-Hee Lee, Tae-Kyung Kang, and Seong-Whan Lee. Kinematic-aware hierarchical attention network for human pose estimation in videos. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 5725–5734, 2023. 6

[14] Lei Jin, Xiaojuan Wang, Xuecheng Nie, Luoqi Liu, Yandong Guo, and Jian Zhao. Grouping by center: Predicting centripetal offsets for the bottom-up human pose estimation. *IEEE TMM*, 2022. 2

[15] Lei Jin, Chenyang Xu, Xiaojuan Wang, Yabo Xiao, Yandong Guo, Xuecheng Nie, and Jian Zhao. Single-stage is enough: Multi-person absolute 3d pose estimation. In *CVPR*, pages 13086–13095, 2022.

[16] Lei Jin, Xiaojuan Wang, Xuecheng Nie, Wendong Wang, Yandong Guo, Shuicheng Yan, and Jian Zhao. Rethinking the person localization for single-stage multi-person pose estimation. *IEEE TMM*, 2023. 1, 2, 3

[17] Muhammed Kocabas, Nikos Athanasiou, and Michael J Black. Vibe: Video inference for human body pose and shape estimation. In *CVPR*, pages 5253–5263, 2020. 2, 6, 7

[18] Muhammed Kocabas, Chun-Hao P Huang, Otmar Hilliges, and Michael J Black. Pare: Part attention regressor for 3d human body estimation. In *ICCV*, pages 11127–11137, 2021. 6, 7

[19] Nikos Kolotouros, Georgios Pavlakos, Michael J Black, and Kostas Daniilidis. Learning to reconstruct 3d human pose and shape via model-fitting in the loop. In *ICCV*, pages 2252–2261, 2019. 2, 6

[20] Ruilong Li, Shan Yang, David A Ross, and Angjoo Kanazawa. Ai choreographer: Music conditioned 3d dance generation with aist++. In *ICCV*, pages 13401–13412, 2021. 6

[21] Yang Li, Kan Li, Shuai Jiang, Ziyue Zhang, Congzhentao Huang, and Richard Yi Da Xu. Geometry-driven self-supervised method for 3d human pose estimation. In *AAAI*, pages 11442–11449, 2020. 1

[22] Hung Yu Ling, Fabio Zinno, George Cheng, and Michiel Van De Panne. Character controllers using motion vaes. *ACM Transactions on Graphics (TOG)*, 39(4):40–1, 2020. 3

[23] Matthew Loper, Naureen Mahmood, Javier Romero, Gerard Pons-Moll, and Michael J Black. Smpl: A skinned multi-person linear model. *ACM TOG*, 34(6):1–16, 2015. 2

[24] Haoyu Ma, Liangjian Chen, Deying Kong, Zhe Wang, Xingwei Liu, Hao Tang, Xiangyi Yan, Yusheng Xie, Shih-Yao Lin, and Xiaohui Xie. Transfusion: Cross-view fusion with transformer for 3d human pose estimation. *arXiv preprint arXiv:2110.09554*, 2021. 6

[25] Haoyu Ma, Zhe Wang, Yifei Chen, Deying Kong, Liangjian Chen, Xingwei Liu, Xiangyi Yan, Hao Tang, and Xiaohui Xie. Ppt: Token-pruned pose transformer formonocular andmulti-view human pose estimation. In *ECCV*, 2022. 6, 7

[26] Julieta Martinez, Rayat Hossain, Javier Romero, and James J Little. A simple yet effective baseline for 3d human pose estimation. In *ICCV*, pages 2640–2649, 2017. 6, 7, 8

[27] Alejandro Newell, Kaiyu Yang, and Jia Deng. Stacked hourglass networks for human pose estimation. In *ECCV*, pages 483–499, 2016. 2, 6, 7

[28] Georgios Pavlakos, Vasileios Choutas, Nima Ghorbani, Timo Bolkart, Ahmed AA Osman, Dimitrios Tzionas, and Michael J Black. Expressive body capture: 3d hands, face, and body from a single image. In *CVPR*, pages 10975–10985, 2019. 3

[29] Dario Pavllo, Christoph Feichtenhofer, David Grangier, and Michael Auli. 3d human pose estimation in video with temporal convolutions and semi-supervised training. In *CVPR*, 2019. 2

[30] William H Press and Saul A Teukolsky. Savitzky-golay smoothing filters. *Computers in Physics*, 4(6):669–672, 1990. 3, 6

[31] Davis Rempe, Tolga Birdal, Aaron Hertzmann, Jimei Yang, Srinath Sridhar, and Leonidas J Guibas. Humor: 3d human motion model for robust pose estimation. In *ICCV*, pages 11488–11499, 2021. 3

[32] Garvita Tiwari, Dimitrije Antić, Jan Eric Lenssen, Nikolaos Sarafianos, Tony Tung, and Gerard Pons-Moll. Pose-ndf: Modeling human pose manifolds with neural distance fields. In *ECCV*, pages 572–589, 2022. 3

[33] Carnegie Mellon University. Cmu graphics lab motion capture database: http://mocap.cs.cmu.edu/. 6

[34] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *NeuIPS*, 30, 2017. 3, 4, 5

[35] Timo Von Marcard, Roberto Henschel, Michael J Black, Bodo Rosenhahn, and Gerard Pons-Moll. Recovering accurate 3d human pose in the wild using imus and a moving camera. In *ECCV*, pages 601–617, 2018. 6

[36] Ziniu Wan, Zhengjia Li, Maoqing Tian, Jianbo Liu, Shuai Yi, and Hongsheng Li. Encoder-decoder with multi-level attention for 3d human shape and pose estimation. In *ICCV*, pages 13033–13042, 2021. 6

[37] Tao Wang, Lei Jin, Zhang Wang, Xiaojin Fan, Yu Cheng, Yinglei Teng, Junliang Xing, and Jian Zhao. Decenternet: Bottom-up human pose estimation via decentralized pose representation. In *ACM MM*, pages 1798–1808, 2023. 1

[38] Wen-Li Wei, Jen-Chun Lin, Tyng-Luh Liu, and Hong-Yuan Mark Liao. Capturing humans in motion: Temporal-attentive 3d human pose and shape estimation from monocular video. In *CVPR*, pages 13211–13220, 2022. 2, 6

[39] 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. 2

[40] Ian T Young and Lucas J Van Vliet. Recursive implementation of the gaussian filter. *Signal processing*, 44(2):139–151, 1995. 3, 6

[41] Ailing Zeng, Xuan Ju, Lei Yang, Ruiyuan Gao, Xizhou Zhu, Bo Dai, and Qiang Xu. Deciwatch: A simple baseline for 10× efficient 2d and 3d pose estimation. In *ECCV*, pages 607–624. Springer, 2022. 6

[42] Ailing Zeng, Lei Yang, Xuan Ju, Jiefeng Li, Jianyi Wang, and Qiang Xu. Smoothnet: A plug-and-play network for refining human poses in videos. In *ECCV*, pages 625–642, 2022. 2, 3, 6, 7