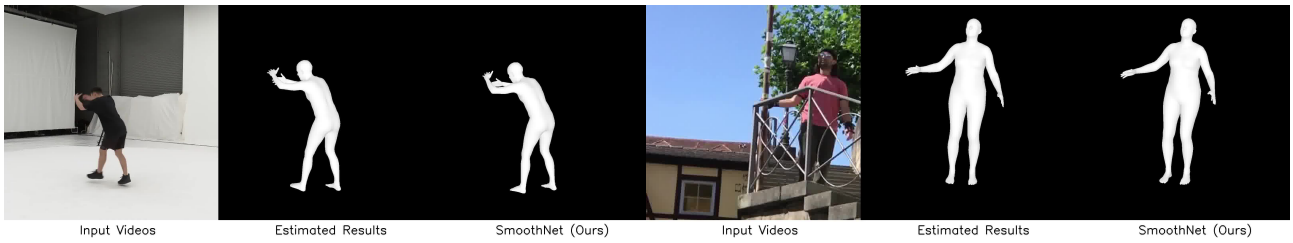
SmoothNet: A Plug-and-Play Network for Refining Human Poses in Videos

Ailing Zeng[1], Lei Yang[2], Xuan Ju[1], Jiefeng Li[3], Jianyi Wang[4], Qiang Xu[1]
[1]The Chinese University of Hong Kong, [2]Sensetime Group Ltd.,
[3]Shanghai Jiao Tong University, [4]Nanyang Technological University



(a) Rare Pose Refinement (b) Occluded Pose Refinement

Figure 1. Existing human pose estimators suffer from severe jitter problem (videos on the middle of (a)(b)), causing untrustworthy perceptions. This work proposes SMOOTHNET to alleviate long-term jitters and significant errors with a simple yet effective refinement network (videos on the right of (a)(b)). *This is a video figure that is best viewed by Adobe Reader.*

Abstract

When analyzing human motion videos, the output jitters from existing pose estimators are highly-unbalanced. Most frames only suffer from slight jitters, while significant jitters occur in those frames with occlusion or poor image quality. Such complex poses often persist in videos, leading to consecutive frames with poor estimation results and large jitters. Existing pose smoothing solutions based on temporal convolutional networks, recurrent neural networks, or low-pass filters cannot deal with such a long-term jitter problem without considering the significant and persistent errors within the jittering video segment.

Motivated by the above observation, we propose a novel plug-and-play refinement network, namely SMOOTHNET, *which can be attached to any existing pose estimators to improve its temporal smoothness and enhance its per-frame precision simultaneously. Specially,* SMOOTHNET *is a simple yet effective data-driven fully-connected network with large receptive fields, effectively mitigating the impact of long-term jitters with unreliable estimation results. We conduct extensive experiments on twelve backbone networks with seven datasets across 2D and 3D pose estimation, body recovery, and downstream tasks. Our results demonstrate that the proposed* SMOOTHNET *consistently outperforms existing solutions, especially on those clips with high errors and long-term jitters. See more qualitative videos on the project page*[1].

[1]https://ailingzeng.site/smoothnet

1. Introduction

2D human pose estimation [8, 32, 45, 52], 3D human pose estimation [16, 46, 65, 66, 68], and human mesh recovery [9, 24, 29, 30, 37] are a series of essential tasks in computer vision that have broad applications such as human-computer interaction and marker-less motion capture. For these tasks, per-frame precision and frame-by-frame smoothness are both critical optimization metrics. Most existing works employ an end-to-end framework for pose estimation. Despite some solutions introducing smoothness enhancement modules [9, 26, 29, 46, 61], they often suffer from severe jitter problems for complicated motions due to the challenges of co-optimizing precision and smoothness simultaneously.

Consequently, a recent trend is to perform pose refinement after an early-stage pose/body estimator. There are mainly two types of pose refinement methods: learning-based pose refinement and conventional low-pass filters. Learning-based refinement networks [27, 37, 57] typically use temporal convolutional networks (TCNs) or recurrent neural networks (RNNs) to learn jitter patterns for pose smoothing. While showing some benefits, their performance is not guaranteed. Applying low-pass filters [6, 10, 14, 19, 23, 47, 56, 64] could reduce jitter to an arbitrarily small value. However, there is a natural jitter-lag tradeoff, resulting in possible precision losses.

With the continuous improvement in human pose/body estimations, the output jitters from recent solutions are highly-unbalanced. Most frames in videos only suffer from

slight jitters and can be easily smoothed with existing pose refinement solutions. The challenges lie with those images that largely deviate from training samples or contain complex or unseen poses due to occlusion or poor image quality. These rare/complex poses often persist in videos for a consecutive sequence of frames, leading to significant estimated errors and serious jitters for a continuous period. Without considering that all estimated results in a relatively long jittering video segment are not trustworthy, present refinement solutions try to smooth poses using local neighboring frames, thereby generating unsatisfactory results.

Motivated by the above observation, we propose SMOOTHNET, a novel plug-and-play pose refinement network that can deal with long-term and significant jitters. We summarize the contributions of this work as follows:

- We analyze the jitter problem in existing pose/body estimators and empirically show that significant jitters usually exist in consecutive rare/complex poses with large pose estimation errors.
- We propose a simple yet effective data-driven fully-connected network (FCN) with large receptive fields, which has the capacity to generate smoothed poses under long-term and significant jitters. Specially, unlike existing smoothing solutions largely affected by local frames with unreliable estimated poses, SMOOTHNET can suppress their impact and use estimated poses beyond the video segment with severe jitters to achieve better results. Moreover, by explicitly modeling the velocity and the acceleration with adjacent frames, SMOOTHNET is motion-aware and converges with better-refined poses efficiently.
- We conduct extensive experiments to validate the performance of SMOOTHNET on **7** datasets, **12** backbones, and **3** motion representations. Our results show that SMOOTHNET consistently outperforms existing solutions, especially for those video clips with high errors and long-term jitters.

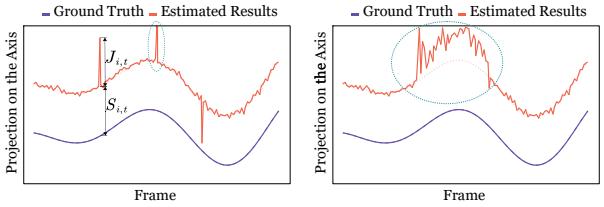
2. Preliminaries and Motivation

2.1. Human Pose Estimation

For human pose estimation from videos, we input L images $\mathbf{X}$ to the pose estimator f, and then output estimated poses $\hat{\mathbf{Y}} \in \mathbb{R}^{L \times C}$, where $C = N \times D$. N is the number of keypoints associated with datasets, and D denotes the number of output dimensions, *e.g.*, $N = 17$ and $D = 3$ for 3D pose estimation with 17 detected skeleton keypoints. The above process can be formulated as follows.

$$\hat{\mathbf{Y}} = f(\mathbf{X}). \tag{1}$$

In a supervised manner, the estimator can be updated by employing the corresponding ground truth $\mathbf{Y} \in \mathbb{R}^{L \times C}$. The target of these tasks is to minimize the distance between the estimation results $\hat{\mathbf{Y}}$ and the ground truth $\mathbf{Y}$.



(a) Sudden Jitters. (b) Long-term Jitters.

Figure 2. Pose estimation results with jitters. The horizontal coordinate represents frames in time and the vertical coordinate represents output values of the axis.

In terms of motion representations, human pose estimation can be divided into 2D [8, 32, 45, 52] and 3D [39, 46, 65, 66, 68] pose estimation, which output 2D or 3D positions of the target person. Meanwhile, 3D pose estimation can be further divided into model-free and model-based methods, where the latter ones are also referred to as human mesh recovery [9, 22, 24, 29, 30, 33, 37].

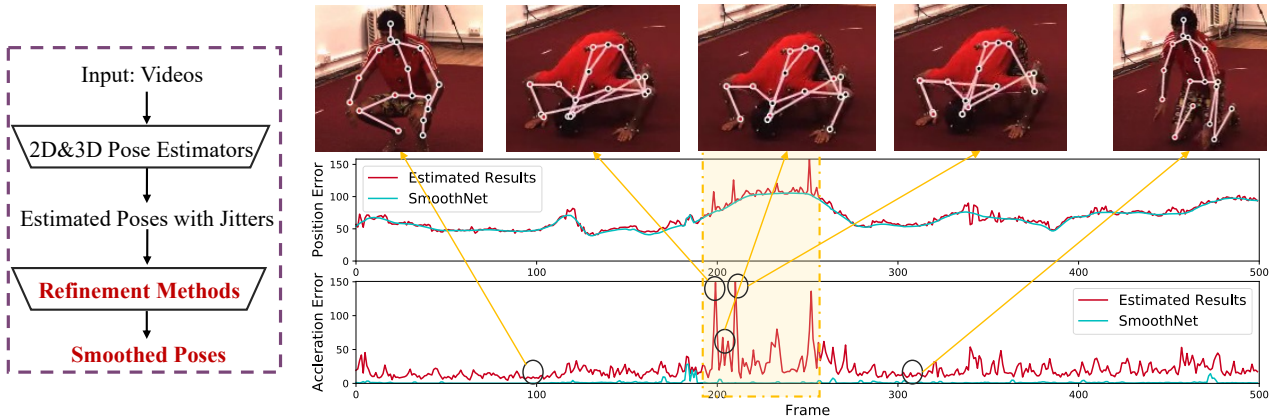
2.2. The Jitter Problem from Pose Estimators

As can be observed in Figure 2, pose estimation errors can be divided into two parts: the jitter between adjacent frames and an overall bias from the ground truth. In particular, in terms of jitters, there are frequent slight jitters due to the uncertainty of pose estimators (e.g., inevitably inconsistent annotations in the training datasets [1, 35]), sudden large jitters (e.g., caused by motion blur), and long-term significant jitters associated with rare/complex poses appearing consecutively in the videos.

Generally speaking, multi-frame pose estimation approaches [9, 26, 29, 37, 46, 65, 66] show advantages over single-frame ones. Specifically, some works apply temporal models (e.g., GRUs [9, 29, 37], TCNs [46, 65], and Transformers [59, 69]) for feature extraction, ensuring the pose estimators have continuous inputs on time sequences. Other methods employ regularizers or loss functions for smoothness [26, 41, 43, 54, 57, 67] to constrain the temporal consistency across successive frames. However, such end-to-end frameworks still suffer from severe jitter problems since co-optimizing per-frame precision and frame-by-frame smoothness simultaneously will meet bottlenecks (as quantitatively explored in supplementary materials). Consequently, a number of pose refinement strategies are proposed in the literature.

2.3. Pose Refinement

Existing related methods roughly fall into two classes:
Learning-based Methods: Some networks [12, 41, 44, 60] only use spatial information (e.g., keypoints or image features) to refine poses without considering smoothness.



(a) Overall Framework (b) Estimated & Refined Results

Figure 3. (a) is the whole framework of 2D, 3D human pose estimation and body recovery with the proposed refinement methods. The proposed SMOOTHNET is a plug-and-play refinement network that smooth estimated results from the above tasks. (b) demonstrates the estimated results of a state-of-the-art estimator RLE [32] and how SMOOTHNET can improve its precision (upper) and smoothness (lower).

There are a few solutions [21, 27, 57] that take smoothness into consideration with spatial-temporal modeling. Specifically, Jiang *et al*. [21] design a transformer-based network to smooth 3D poses in sign language recognition. Kim *et al*. [27] propose a non-local attention mechanism with convolutions represented by *quaternions*. Considering the occlusions on multi-person scenes, Vege *et al*. [57] use an energy optimization with visibility scores to adaptively filter the keypoint trajectories. However, since human topologies in various datasets are very different, relying on spatial features can lead to high computational costs and make it hard to generalize cross datasets. Recently, several generative models [37, 48, 67] are proposed to reconstruct smooth and precise motions, but they may produce unrealistic poses on unseen motions. Despite promising results in precision, the above refinement methods are usually specific to a particular motion representation, jitter source, or backbone, limiting their applications. In contrast, general low-pass filters [47, 64] can be used for pose refinement as well, and their smoothing capability is usually higher than learning-based solutions (as shown in Section 4.2).

Low-Pass Filters: 1D Low-pass filters are simple yet effective solutions for pose smoothing, especially for slight jitters. Kalman Filters [4, 23, 49] are optimal state estimators with prediction and update steps under linearity and Gaussian noise assumptions. One-Euro filter [6] proposes a first-order filter's design with an adaptive cutoff frequency and exceeds the performance of Moving average and Kalman filters. However, these approaches fail when the input of the current frame is inaccurate because of cumulative errors. In order to exploit temporal information beyond subsequent data, another idea proposed to smooth jitters based on the sliding window algorithm. Specifically, moving averages [18] calculates the mean values over a specified period of time. Savitzky-Golay filter [47] uses a local polynomial least-squares function to fit the sequence within a given window size. Gaussian filter [64] modifies the input signal by convolution with a Gaussian function to obtain the minimum possible group delay. Although these methods have been broadly used in various areas, they still face a trade-off between jitters and lags, especially long-term jitters.

2.4. Motivation

In human motion videos, there are both easily estimated sequences and complex ones (e.g., under occluded or rare poses). As shown in Figure 3(b), existing pose estimators can output relatively accurate estimation results for most frames. Interestingly, the video segment with large position estimation errors also suffers from significant jitters. The reason is simple: such video segments contain rare/complex poses that are challenging to estimate, causing frequent estimation changes among adjacent frames.

Existing pose refinement methods fail to deal with such long-term and significant jitters.

- Existing learning-based solutions employ RNNs or TCNs to learn the jitter patterns [27, 37, 57]. On the one hand, the highly-unbalanced jitter patterns require many distinct learnable kernels, which is less compatible to the shared kernel design philosophy in RNN or TCN designs. On the other hand, these solutions focus more on extracting local temporal features. Consequently, they are inevitably affected by the large fluctuations within the jittering windows.

- While low-pass filters can reduce jitters within a timing window to an arbitrarily small value, they have a natural jitter-lag tradeoff. In order to mitigate long-term jitters, filters need to be applied on long timing windows, thereby resulting in more lags with potential precision losses.

Motivated by the above, we propose to employ fully-connected networks with large receptive fields to learn the jitter patterns from a global perspective.

3. Method

To cope with unbalanced jitter patterns in human pose estimators, especially long-term and significant jitters, our proposed SMOOTHNET g learns the noisy estimation $\hat{\mathbf{Y}} \in \mathbb{R}^{L\times C}$ generated from any pose/body estimators f (as mentioned in Eq. 1).

$$\hat{\mathbf{G}} = g(\hat{\mathbf{Y}}), \tag{2}$$

where g is the proposed SMOOTHNET, and $\hat{\mathbf{G}} \in \mathbb{R}^{L\times C}$ is the refined poses produced by our approach.

3.1. Basic SmoothNet

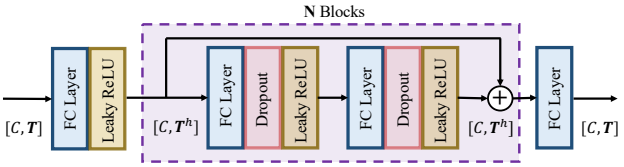


Figure 4. A simple yet effective SMOOTHNET design.

To capture the long-term temporal relationship, a natural idea is to increase the effective receptive field. As fully-connected layer has been widely used to model long-term dependency in various tasks [11,53], we draw on its expressive power to learn long-term jitter patterns. Besides, according to the principle of superposition of movements [13], a movement can be decomposed as the superposition of several movements that are performed independently. Based on such principle, each axis i in channel C can be processed independently. Rather than implementing the FC layers on spatial dimensions [37,39,41,46], we apply FC layers along time axis, which is rarely explored. The proposed network is shown in Figure 4, where we construct multiple fully connected (FC) layers with residual connections along time axis. The computation of each layer can be formulated as follows.

$$\hat{Y}_{i,t}^{l+1} = \sigma(\sum_{t=0}^{T} w_t^l * \hat{Y}_{i,t}^l + b^l), \tag{3}$$

where w_t^l and b^l are learnable weights and bias at the t_{th} frame and they are shared among different i_{th} axis, respectively. σ is the non-linear activation function (LeakyReLU is chosen). To process $\hat{\mathbf{Y}}$ with SMOOTHNET, we adopt a sliding-window scheme similar to filters [31,47,64], where we first extract a chunk with size T, yield refined results thereon and then move to next chunk with a step size s.

3.2. Motion-aware SmoothNet

As our goal is to capture jitter patterns, which is mainly presented as acceleration errors, it is straightforward to model acceleration in addition to position. As Figure 5 illustrates, we explicitly model the movement function in our network, *i.e.*, velocity and acceleration. Given the prior with physical meaning, it is beneficial to leverage first-order and second-order motion information, making the learning process converge better and faster than *Basic* SMOOTHNET. Specifically, given the input $\hat{\mathbf{Y}}$, we first compute the velocity and acceleration for each axis i, according to the Equation 4.

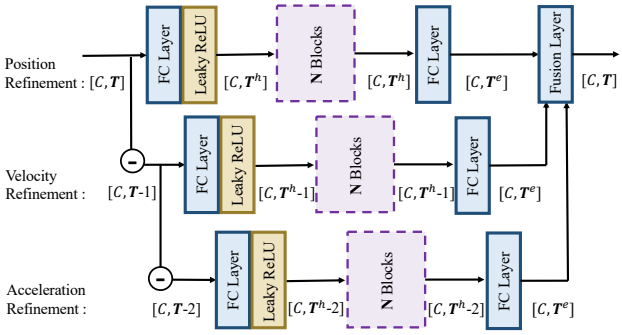


Figure 5. The motion-aware SMOOTHNET design. It explicitly models the velocity and acceleration with adjacent frames to achieve better pose refinement.

$$\hat{V}_{i,t} = \hat{Y}_{i,t} - \hat{Y}_{i,t-1}, \quad \hat{A}_{i,t} = \hat{V}_{i,t} - \hat{V}_{i,t-1}. \tag{4}$$

As shown in Figure 5, the top branch is the baseline stream to refine noisy positions $\hat{\mathbf{Y}}$. The other two branches input the corresponding noisy velocity $\hat{\mathbf{V}}$ and acceleration $\hat{\mathbf{A}}$. To capture the long-term temporal cue, we also employ Equation 3 to refine velocity and acceleration. To aggregate information from different order of motions, we concatenate the top embedding of three branches and perform a linear fusion layer to obtain the final refined poses $\hat{\mathbf{G}}$. Similar to the basic scheme in Section 3.1, this motion-aware scheme also works in a sliding-window manner to process the whole input sequence.

3.3. Loss Function

SMOOTHNET aims to minimize both position errors and acceleration errors during training, the objective functions are defined as follows.

$$L_{pose} = \frac{1}{T\times C}\sum_{t=0}^{T}\sum_{i=0}^{C}|\hat{G}_{i,t} - Y_{i,t}|, \tag{5}$$

$$L_{acc} = \frac{1}{(T-2)\times C}\sum_{t=0}^{T}\sum_{i=0}^{C}|\hat{G}''_{i,t} - A_{i,t}|, \tag{6}$$

where $\hat{G}''_{i,t}$ is the computed acceleration from predicted pose $\hat{G}_{i,t}$ and $A_{i,t}$ is the ground-truth acceleration. We simply add L_{pose} and L_{acc} as our final target.

4. Experiments

We validate the proposed SMOOTHNET from quantitative and qualitative results in the following sections. Due to the page limitation, we leave more analysis and discussions to Supplementary material. For more experimental details, please refer to the provided code.

4.1. Experimental Settings

Backbones. We validate the generalization ability on both smoothness and precision of the proposed SMOOTHNET covering three related tasks as shown in Figure 6.

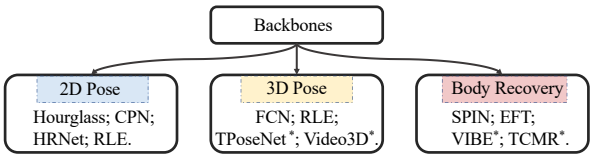


Figure 6. List of backbones to be refined by SMOOTHNET. * means multi-frame approaches.

Training set. To prepare training data, we first save the outputs of existing methods, including estimated 2D positions, estimated 3D positions, or estimated SMPL parameters. Then, we take these outputs as the inputs of SMOOTHNET and use the corresponding ground-truth data as the supervision to train the model. In particular, we use the outputs of FCN on Human3.6M, SPIN in *3DPW* [58] and VIBE on *AIST++* [34] to train the networks.

Testing set. We validate SMOOTHNET with different backbones on *Human3.6M* [20], *3DPW* [58], *MPI-INF-3DHP* [40], *AIST++* [34] and MuPoTS-3D [42] datasets.

Downstream task. Skeleton-based action recognition inputs 3D poses to classify actions as an important downstream task. To explore the effectiveness of more robust inputs, we first smooth the serious jitters of the ground-truth 3D positions from *NTU-RGBD 60* [50] and *NTU-RGBD 120* [36], and then train existing classifiers.

Evaluation Metrics. To measure the jitter errors, we follow the related works [9, 26, 29] to adopt the mean per joint acceleration error (Accel). Its unit is *mm/ frame*^{2} for 3D poses and *pixel/ frame*^{2} for 2D poses. We highlight that *Accel* is the primary metric in the following discussion. To evaluate the precision for each frame, there are two commonly used metrics, namely the *mean per joint position error (MPJPE)* and the *Procrustes Analysis MPJPE (PA-MPJPE)* that remove effects on the inherent scale, rotation, and translation issues. For 3D pose estimation, their unit is *mm*. For 2D pose estimation, to validate the accurate localization precision, we simply use *pixel* as the unit.

Implementation Details The basic SMOOTHNET is an eight-layer model including the first layer, three cascaded blocks with a residual connection, and the last layer as a decoder. The motion-aware SMOOTHNET contains three parallel branch with the first layer, one cascaded block and the last layer for each branch. The parameters of SMOOTHNETis 0.33M and the average inference time is less than $1e^{-5}$ second per frame. The input window size T is 64 and the moving step size s is 1. In addition, we use the sliding window average algorithm [31] based on refined results to further reduce spikes.

Table 1. Comparison with widely-used filters on pose estimation results from VIBE [29] of AIST++ dataset [34].

	Method	Accel	MPJPE	PA-MPJPE	Test FPS
Human Mesh Recovery	VIBE [29]	33.16	108.82	73.97	-
	w/ One-Euro [6]	23.59	108.51	73.84	28.31k
	w/ Savitzky-Golay [47]	5.84	105.80	72.15	269.63k
	w/ Gaussian1d [64]	4.95	104.54	72.50	210.47k
	w/ One-Euro [6]	4.71	154.60	111.10	26.80k
	w/ Savitzky-Golay [47]	4.76	117.85	86.90	162.09k
	w/ Gaussian1d [64]	4.54	104.60	71.90	211.93k
	w/ Ours	**4.36**	**92.46**	**69.75**	**833.92k**

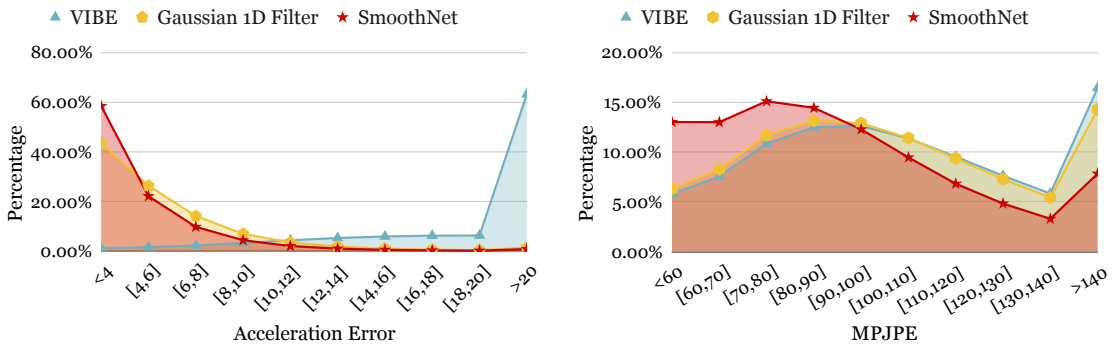
4.2. Comparison with Existing Solutions

4.2.1 Comparison with Filters

We compare three commonly used filters with SMOOTHNET combined VIBE [29] on AIST++ dataset. SMOOTHNET is trained on training sets of VIBE-AIST++.

In Table 1, we combine SMOOTHNET with the temporal backbone VIBE, where it can boost the acceleration error by 86.85% and MPJPE by 15.03%. Since the performance of filters will be heavily influenced by their hyperparameters, we try grid search in Table 1 to find the comparable Acceleration errors (lower parts) and MPJPE (upper parts) with us, respectively. From the upper parts, we observe that filters would suffer from over-smoothing problems in order to obtain similar acceleration error with us, resulting in delayed and unrealistic poses with high position errors. From the lower parts, we find that they are hard to achieve lower MPJPE due to the lack of prior on these noisy poses. That shows more precise filters will result in the less effective at mitigating jitters. Specifically, for the frame-by-frame approach, like the One-Euro filter, we found it is more difficult to achieve a lower smoothing effect as sliding-window-based filters [47, 64], so we opt for compromise results. Under a large variation of noisy motions, like diverse and complex dances in AIST++, SMOOTHNET , as a learning-based approach with a long-range receptive field, shows superiorities in both reducing jitters J and biased errors S. Besides, as our method can benefit from GPU parallelism, it yields faster inference speeds than previous methods.

Moreover, we further analyze the *MPJPE* and *Accel* distribution of VIBE, VIBE with Gaussian 1D Filter and VIBE with SMOOTHNETin Figure 7. In terms of *Accel*, 98.7% of VIBE's original output falls above $4\ mm/frame^2$. Processed by Gaussian filter and SMOOTHNET , the percentage decreases to 56.5% and 41.6% respectively, where the



(a) *Accel* Distribution (b) *MPJPE* Distribution

Figure 7. Comparison of smoothness and precision distributions on AIST++.

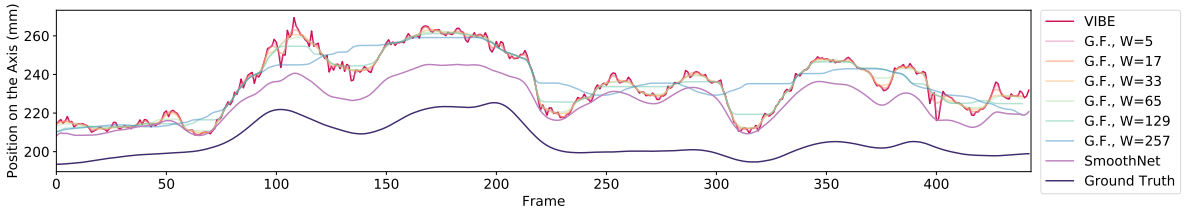


Figure 8. Performance comparison between filters and SMOOTHNET on refining the estimated results of VIBE.

performance of SMOOTHNET is 26.4% better than the filer. This illustrates that our method can relieve larger jitters than Gaussian 1D Filter. As for MPJPE, 62.1% of VIBE's output is larger than 90 mm. Gaussian 1D Filter reduced the proportion to 59.3% (only 4.5%) by removing certain jitters. The advantage of our model over traditional filters is proven by decreasing the percentage to 43.1% (by 30.6% improvement), indicating SMOOTHNET can mitigate the influence on long-term jitters with unreliable poses. This is because SMOOTHNET has the capability to learn the long-range noisy patterns and then cut down the biased error S in a data-driven manner.

Qualitative Results. We visualize the qualitative results on a specific axis to demonstrate the effectiveness of SMOOTHNET. Specially, Figure 8 illustrates the output positions of VIBE, VIBE with several Gaussian filters (G.F.,) of different window sizes, VIBE with our method and the ground truth. The filter can relieve jitter errors with the increase of window size but suffer from over-smoothness when the window size is larger than 65, leading to worse position errors. Instead, with a learnable design and long-range receptive fields, SMOOTHNET can not only relieve jitters but also narrow down biased errors consistently.

4.2.2 Comparison with RefineNet

Comparable learning-based methods are HPRNet [27] and RefineNet [57]. Since the former one is not open source with different training and testing settings, we can only compare with RefineNet, which is designed for multi-person 3D pose estimation, on MuPoTS-3D [42] dataset. This model is trained on multi-person MPI-INF-3DHP and MuCo-Temp datasets, while SMOOTHNET are train on VIBE-AIST++ (the same model in Section 4.2.1), without any finetuning to explore the generalization ability of SMOOTHNET cross datasets and backbones. Additionally, RefineNet has compared with the interpolation and One-Euro filter [6], and it can surpass both two methods, therefore we will not compare them again. We compare them on the universal coordinates, where each person is rescaled according to their hip and has a normalized height. In Table 2, we first analyze the effectiveness of the combination with the TPoseNet backbone [57], which is a temporal residual convolution network for 2D-to-3D pose estimation proposed as the baseline of RefineNet. Although MPJPE of RefineNet drops the most as it has been trained on the relevant datasets, we discover that its acceleration error is the highest, indicating that the smoothing ability of RefineNet is not even as good as the filters [47, 64]. Furthermore, our method improves filters on Accel by 12.80%. At the same time, the learning-based method shows more powerful for reducing the biased errors by 2.4% than filters. Finally, we try to refine the RefineNet with filters and SMOOTHNET. The bottom half of Table 2 demonstrates all methods have performance gain. Among them, SMOOTHNET still obtains the largest improvements by 13.4% and 2.5% in *Accel* and *MPJPE* respectively, and it can be complementary to existing learning-based methods.

4.3. Combination with Existing Methods

As a plug-and-play network, SMOOTHNET can combine with the existing methods, we show the results on both skeleton-based methods in Section 4.3.1 and SMPL-based

Table 2. Experimental results on MuPoTS.

Method	Accel	MPJPE	PA-MPJPE
TPoseNet [57]	12.71	103.34	68.35
TPoseNet w/ RefineNet [57]	9.50	**93.91**	**65.13**
TPoseNet w/ Savitzky-Golay	8.11	102.83	68.37
TPoseNet w/ Gaussian1d	7.89	102.73	68.40
TPoseNet w/ Ours	**6.88**	100.31	67.13
RefineNet w/ Savitzky-Golay	7.89	93.67	65.01
RefineNet w/ Gaussian1d	7.68	93.56	65.07
RefineNet w/ Ours	**6.65**	**91.20**	**64.11**

methods in Section 4.3.2.

4.3.1 2D and 3D Pose Estimation

In Table 3, we compare the results of skeleton-based methods on Human3.6M dataset. The Acceleration errors of all the backbones combined with our method are significantly reduced, and MPJPEs are also reduced to some extent. In specific, *Accel* and its MPJPE are reduced to a greater extent for the single-frame networks. Also, we observe that *Accel* is similar to different backbones, representing that SMOOTHNET has capability to relieve kinds of jitters. Since the SMOOTHNET is only train on FCN-Human3.6M, the improvements on FCN [39] will be larger than other backbones by 95.3%, 4.46% and 3.9% in *Accel*, MPJPE and PA-MPJPE respectively. To explore the effect on significant errors and long-term jitters, we further calculate the worst 10% of MPJPEs (MP.-10%) and their corresponding *Accel* (AC-10%) as the poorer estimated poses for each backbone. All estimated errors on 3D poses are decreased by about 25%, especially 85.9% for the trained backbone FCN getting the best-refined MPJPE (62.94mm). Those results can also validate its ability to generalize across backbones well.

Table 3. Results of SMOOTHNET on 2D and 3D pose estimators on Human3.6M. * means multi-frame methods. AC., MP., PA., are the abbreviations of Accel, MPJPE and PA-MPJPE.

	Method	AC.	MP.	PA.	MP.-10%	AC.-10%
2D Pose Estimation	Hourglass [45]	1.55	7.98	6.20	24.38	2.38
	Hourglass w/ours	**0.16**	**7.68**	**6.01**	**24.06**	**0.17**
	CPN [8]	2.91	6.67	5.18	20.89	3.96
	CPN w/ours	**0.14**	**6.31**	**4.81**	**20.44**	**0.17**
	HRNet [52]	1.01	4.60	4.20	9.70	2.03
	HRNet w/ours	**0.15**	**4.51**	**4.14**	**9.42**	**0.17**
	RLE [32]	0.90	5.32	4.82	10.43	1.59
	RLE w/ours	**0.13**	**5.21**	**4.76**	**10.27**	**0.16**
3D Pose Estimation	FCN [39]	19.18	54.48	42.20	445.26	23.2
	FCN w/ours	**0.90**	**52.05**	**40.54**	**62.94**	**1.01**
	RLE [32]	7.76	48.91	38.66	91.29	10.90
	RLE w/ours	**0.85**	**48.11**	**38.19**	**76.88**	**0.84**
	[46] (T=27)*	5.07	50.39	39.13	95.38	4.82
	[46] (T=27)* w/ours	**0.87**	**49.87**	**38.98**	**70.91**	**0.87**
	[46] (T=81)*	3.06	48.98	38.27	93.56	4.11
	[46] (T=81)* w/ours	**0.87**	**48.75**	**38.01**	**70.02**	**0.87**
	[46] (T=243)*	2.82	48.13	37.71	92.76	3.80
	[46] (T=243)* w/ours	**0.87**	**47.98**	**37.62**	**70.27**	**0.86**

4.3.2 Human Mesh Recovery

In Table 4, we give the results of SMPL-Based methods for body recovery on 3DPW, MPI-INF-3DHP and Human3.6M dataset. SMOOTHNET is trained with the outputs from SPIN [30], where it has large range noises and is tested across different backbones. Overall, our method has a consistent improvement in smoothness and precision. In specific, SMOOTHNET can reduce *Accel* on SPIN and VIBE by a large margin due to its effective smoothness ability. For the temporal baseline VIBE, our method improves by about 80% and 2% on Accel and MPJPE, respectively. Moreover, because TCMR has used some smooth strategies in their models, their original *Accel* is relatively small. But we find its first and last few frames could not be smoothed out, resulting in larger *Accel* In this condition, our method can relieve those jitters and further enhance its performance. Meanwhile, we add the post-processing slerp filter to minimize Euclidean distance on quaternion from MEVA on TCMR. The filter can improve *Accel* but cause over-smoothness, leading to higher position errors.

4.4. Ablation Study

Analysis on Long-range Scheme. To further verify the effect and necessity of such long-range receptive fields, we compare ways of doing local convolution with small kernel size (here is 3), such as temporal convolution networks [2]. We conduct experiments on TCNs with different window sizes T and compare them with SMOOTHNET. Similar to Section 4.2.1, we use inputs from VIBE-AIST++. Table 5 indicates (i) the performance of TCN increases as the increase of window size; (ii) the *Accel* of TCNs are still worse than filters [64], implying local aggregation for noisy poses with the shared kernels can not handle time-varied jitters well; (iii) but *MPJPE* of TCNs are lower than filters, indicating the learning-based methods can further reduce biased errors S with learning the noisy pose prior; (iv) SMOOTHNET reveals its superiority with long-range receptive fields for each layer to capture global temporal coherence and suppress the influence of unreliable estimations, gaining more on both smoothness and precision.

Analysis on Model Design. To capture long-range noisy poses correlations, we first propose a simple basic model with the residual fully connected network on temporal dimension, called *Basic* SMOOTHNET. To further add known motion function explicitly to deep models, we design a motion-aware temporal network as the SMOOTHNET. Figure 9 illustrates the training and testing precision curves of these two models on 3DPW. We can observe that (i) *Basic* model tends to somewhat overfit; (ii) SMOOTHNET fits better and obtain slightly lower position errors. In comprehensive studies, we summarize the motion-aware SMOOTHNET can fit better than the basic one, while the basic one can still obtain impressive results with its simple design.

Table 4. Evaluation of state-of-the-art methods on 3DPW [58], MPI-INF-3DHP [40], and Human3.6M [20]. All methods do not use 3DPW [58] on training. * means multi-frame backbones.

Method	3DPW			MPI-INF-3DHP			Human3.6M		
	Accel ↓	MPJPE ↓	PA-MPJPE ↓	Accel ↓	MPJPE ↓	PA-MPJPE ↓	Accel ↓	MPJPE ↓	PA-MPJPE ↓
SPIN [30]	29.8	102.4	60.1	29.6	106.8	67.0	18.6	68.5	46.5
SPIN w/ours	**6.0**	**97.6**	**58.6**	**6.2**	**103.5**	**65.9**	**2.8**	**67.5**	**46.3**
VIBE* [29]	23.2	83.0	52.0	22.8	96.4	63.7	15.8	78.1	53.7
VIBE* w/ours	**6.2**	**82.1**	**51.6**	**6.0**	**95.2**	**63.2**	**2.9**	**76.2**	**53.1**
TCMR* [9]	6.8	86.5	52.7	8.5	97.6	63.5	3.9	73.6	52.0
TCMR w/MEVA* [37]	6.2	88.7	55.0	-	-	-	3.1	77.2	55.4
TCMR* w/ours	**5.9**	**86.0**	**52.4**	**5.8**	**96.9**	**63.1**	**2.8**	**73.1**	**51.7**

Table 5. Comparison results with TCNs (T) on VIBE-AIST++.

Method	Gaussian1d	TCN(27)	TCN(81)	TCN(243)	Ours
Accel	4.95	14.46	11.84	10.07	**4.36**
MPJPE	104.54	103.53	101.17	99.76	**92.46**
PA-MPJPE	72.50	72.99	72.30	71.92	**69.75**

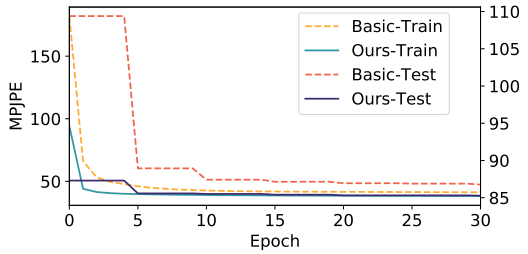


Figure 9. Impact of model designs.

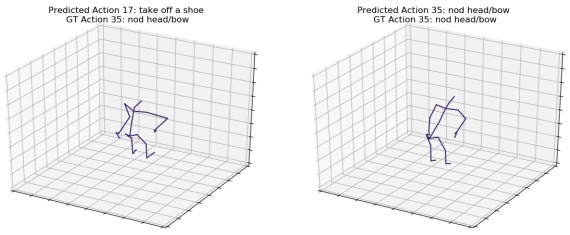
Impact on Window Size. The same sliding-window based methods [9, 27, 29, 47], the window size W has a large impact on smoothness. We demonstrate the effects on different window size from 2 to 128 frames in Table 6. As the window size becomes longer, the *Accel* will decrease consistently, but the *MPJPE* and *PA-MPJPE* will be reduced first and then increases, indicating that 64 frames will be a suitable size to balance the smoothness and precision.

Table 6. Effect of window size W on VIBE-AIST++ [34].

W	VIBE	2	8	16	32	64	128
Accel	33.16	18.98	6.49	4.99	4.48	4.36	**4.34**
MPJPE	108.82	102.88	96.59	93.90	93.04	**92.46**	93.20
PA-MPJPE	73.97	73.55	70.94	69.79	**69.56**	69.75	70.57

4.5. Impact on Downstream Task

3D skeleton-based action recognition is essential for human motion understanding. The two largest and widely-used action classification datasets, NTU RGB+D 60 and 120 [36, 50], are collected with Kinect. They are quite noisy (see Figure 10(a)), which has an adversarial impact on their robustness. Since SMOOTHNET is a pose smoothing network, we exploit it to denoise the original 3d poses (shown in Figure 10(b)). In Table 7, we present the Top-1 accuracy among different inputs to train the popular networks [51, 63]. With SMOOTHNET, the accuracy can be further increased, especially it improves by 1.4% and 1.3% on *X-Sub* and *X-Set* respectively for the more complex dataset with 120 classes.



(a) Existing Ground Truth [50] (b) SMOOTHNET

Figure 10. Illustration of the ground truth of the existing skeleton-based action recognition dataset [50] and processed by SMOOTHNET. *This is a video that is best viewed by Adobe Reader.*

Table 7. Comparison against existing methods on originally noisy or smoothed 3D positions of NTU RGB+D 60 and 120 dataset in terms of Top-1 accuracy(%). Higher values will be better.

Method	NTU RGB+D 60		NTU RGB+D 120	
	X-Sub (%)	X-View (%)	X-Sub (%)	X-Set (%)
ST-GCN [63]	84.3	92.7	71.3	72.4
ST-GCN w/ours	**84.9**	**92.8**	**72.9**	**73.0**
2s-AGCN [51]	88.9	95.1	82.9	84.9
2s-AGCN w/ours	**89.7**	**95.3**	**84.1**	**86.0**

5. Conclusion

In this work, we propose SMOOTHNET, a simple yet effective pose refinement network to improve the temporal smoothness and per-frame precision of existing pose/body estimators. Compared to existing solutions, SMOOTHNET can deal with long-term significant jitters that occurred often with rare or occluded poses, as verified with comprehensive experiments on a large number of backbone networks and datasets.

Limitations: As a post-processing model, although our approach consistently improves the performance of existing backbones, the final pose estimation performance is constrained by the given backbones. Therefore, it would be interesting to explore the inter-play between the backbone network and SMOOTHNET, and we leave it for future work.

References

[1] Mykhaylo Andriluka, Leonid Pishchulin, Peter Gehler, and Bernt Schiele. 2d human pose estimation: New benchmark and state of the art analysis. In *Proceedings of the IEEE Conference on computer Vision and Pattern Recognition*, pages 3686–3693, 2014. 2

[2] Shaojie Bai, J. Zico Kolter, and Vladlen Koltun. An empirical evaluation of generic convolutional and recurrent networks for sequence modeling. *ArXiv*, abs/1803.01271, 2018. 7

[3] 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 *Computer Vision – ECCV 2016*, Lecture Notes in Computer Science. Springer International Publishing, Oct. 2016. 0

[4] Michael Burke and Joan Lasenby. Estimating missing marker positions using low dimensional kalman smoothing. *Journal of biomechanics*, 49(9):1854–1858, 2016. 3

[5] Yujun Cai, Liuhao Ge, Jun Liu, Jianfei Cai, Tat-Jen Cham, Junsong Yuan, and Nadia Magnenat Thalmann. Exploiting spatial-temporal relationships for 3d pose estimation via graph convolutional networks. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2272–2281, 2019. 0

[6] 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. 1, 3, 5, 6, 2

[7] Kang Chen, Yupan Wang, Song-Hai Zhang, Sen-Zhe Xu, Weidong Zhang, and Shi-Min Hu. Mocap-solver: a neural solver for optical motion capture data. *ACM Transactions on Graphics (TOG)*, 40(4):1–11, 2021. 2

[8] Yilun Chen, Zhicheng Wang, Yuxiang Peng, Zhiqiang Zhang, Gang Yu, and Jian Sun. Cascaded pyramid network for multi-person pose estimation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7103–7112, 2018. 1, 2, 7

[9] 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 *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1964–1973, 2021. 1, 2, 5, 8, 0, 3, 4

[10] Huseyin Coskun, Felix Achilles, Robert S. DiPietro, Nassir Navab, and Federico Tombari. Long short-term memory kalman filters: Recurrent neural estimators for pose regularization. *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 5525–5533, 2017. 1

[11] Xiaohan Ding, Chunlong Xia, Xiangyu Zhang, Xiaojie Chu, Jungong Han, and Guiguang Ding. Repmlp: Re-parameterizing convolutions into fully-connected layers for image recognition. *arXiv preprint arXiv:2105.01883*, 2021. 4

[12] Mihai Fieraru, Anna Khoreva, Leonid Pishchulin, and Bernt Schiele. Learning to refine human pose estimation. In *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, pages 205–214, 2018. 2

[13] Mark G Fischman. Programming time as a function of number of movement parts and changes in movement direction. *Journal of Motor Behavior*, 16(4):405–423, 1984. 4

[14] Joela F Gauss, Christoph Brandin, Andreas Heberle, and Welf Löwe. Smoothing skeleton avatar visualizations using signal processing technology. *SN Computer Science*, 2(6):1–17, 2021. 1, 2, 4

[15] Nima Ghorbani and Michael J Black. Soma: Solving optical marker-based mocap automatically. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 11117–11126, 2021. 2

[16] Kehong Gong, Jianfeng Zhang, and Jiashi Feng. Poseaug: A differentiable pose augmentation framework for 3d human pose estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8575–8584, 2021. 1

[17] Daniel Holden. Robust solving of optical motion capture data by denoising. *ACM Transactions on Graphics (TOG)*, 37(4):1–12, 2018. 2

[18] J Stuart Hunter. The exponentially weighted moving average. *Journal of quality technology*, 18(4):203–210, 1986. 3

[19] Rob J Hyndman. Moving averages., 2011. 1, 4

[20] 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 transactions on pattern analysis and machine intelligence*, 36(7):1325–1339, 2013. 5, 8, 0

[21] Tao Jiang, Necati Cihan Camgoz, and Richard Bowden. Skeletor: Skeletal transformers for robust body-pose estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3394–3402, 2021. 3

[22] Hanbyul Joo, Natalia Neverova, and Andrea Vedaldi. Exemplar fine-tuning for 3d human model fitting towards in-the-wild 3d human pose estimation. *arXiv preprint arXiv:2004.03686*, 2020. 2, 3, 4

[23] Rudolph Emil Kalman. A new approach to linear filtering and prediction problems. 1960. 1, 3

[24] Angjoo Kanazawa, Michael J Black, David W Jacobs, and Jitendra Malik. End-to-end recovery of human shape and pose. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7122–7131, 2018. 1, 2

[25] Angjoo Kanazawa, Michael J Black, David W Jacobs, and Jitendra Malik. End-to-end recovery of human shape and pose. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7122–7131, 2018. 0

[26] Angjoo Kanazawa, Jason Y Zhang, Panna Felsen, and Jitendra Malik. Learning 3d human dynamics from video. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5614–5623, 2019. 1, 2, 5

[27] Do-Yeop Kim and Ju-Yong Chang. Attention-based 3d human pose sequence refinement network. *Sensors*, 21(13):4572, 2021. 1, 3, 6, 8

[28] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 0

[29] Muhammed Kocabas, Nikos Athanasiou, and Michael J Black. Vibe: Video inference for human body pose and shape estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5253–5263, 2020. 1, 2, 5, 8, 0, 4

[30] 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 *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 2252–2261, 2019. 1, 2, 7, 8, 0, 4

[31] Chang-Hung Lee, Cheng-Ru Lin, and Ming-Syan Chen. Sliding-window filtering: an efficient algorithm for incremental mining. In *Proceedings of the tenth international conference on Information and knowledge management*, pages 263–270, 2001. 4, 5

[32] Jiefeng Li, Siyuan Bian, Ailing Zeng, Can Wang, Bo Pang, Wentao Liu, and Cewu Lu. Human pose regression with residual log-likelihood estimation. In *ICCV*, 2021. 1, 2, 3, 7, 4

[33] Jiefeng Li, Chao Xu, Zhicun Chen, Siyuan Bian, Lixin Yang, and Cewu Lu. Hybrik: A hybrid analytical-neural inverse kinematics solution for 3d human pose and shape estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3383–3393, 2021. 2

[34] Ruilong Li, Shan Yang, David A. Ross, and Angjoo Kanazawa. Learn to dance with aist++: Music conditioned 3d dance generation, 2021. 5, 8, 0

[35] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer, 2014. 2

[36] Jun Liu, Amir Shahroudy, Mauricio Lisboa Perez, Gang Wang, Ling-Yu Duan, and Alex Kot Chichung. Ntu rgb+ d 120: A large-scale benchmark for 3d human activity understanding. *IEEE transactions on pattern analysis and machine intelligence*, 2019. 5, 8, 0

[37] Zhengyi Luo, S Alireza Golestaneh, and Kris M Kitani. 3d human motion estimation via motion compression and refinement. In *Proceedings of the Asian Conference on Computer Vision*, 2020. 1, 2, 3, 4, 8

[38] Utkarsh Mall, G Roshan Lal, Siddhartha Chaudhuri, and Parag Chaudhuri. A deep recurrent framework for cleaning motion capture data. *arXiv preprint arXiv:1712.03380*, 2017. 2

[39] Julieta Martinez, Rayat Hossain, Javier Romero, and James J Little. A simple yet effective baseline for 3d human pose estimation. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2640–2649, 2017. 2, 4, 7, 0, 1

[40] Dushyant Mehta, Helge Rhodin, Dan Casas, Pascal Fua, Oleksandr Sotnychenko, Weipeng Xu, and Christian Theobalt. Monocular 3d human pose estimation in the wild using improved cnn supervision. In *2017 international conference on 3D vision (3DV)*, pages 506–516. IEEE, 2017. 5, 8, 0

[41] Dushyant Mehta, Oleksandr Sotnychenko, Franziska Mueller, Weipeng Xu, Mohamed Elgharib, Pascal Fua, Hans-Peter Seidel, Helge Rhodin, Gerard Pons-Moll, and Christian Theobalt. Xnect: Real-time multi-person 3d motion capture with a single rgb camera. *ACM Transactions on Graphics (TOG)*, 39(4):82–1, 2020. 2, 4

[42] Dushyant Mehta, Oleksandr Sotnychenko, Franziska Mueller, Weipeng Xu, Srinath Sridhar, Gerard Pons-Moll, and Christian Theobalt. Single-shot multi-person 3d pose estimation from monocular rgb. *2018 International Conference on 3D Vision (3DV)*, pages 120–130, 2018. 5, 6, 0

[43] Dushyant Mehta, Srinath Sridhar, Oleksandr Sotnychenko, Helge Rhodin, Mohammad Shafiei, Hans-Peter Seidel, Weipeng Xu, Dan Casas, and Christian Theobalt. Vnect: Real-time 3d human pose estimation with a single rgb camera. *ACM Transactions on Graphics (TOG)*, 36(4):1–14, 2017. 2

[44] Gyeongsik Moon, Ju Yong Chang, and Kyoung Mu Lee. Posefix: Model-agnostic general human pose refinement network. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7773–7781, 2019. 2

[45] Alejandro Newell, Kaiyu Yang, and Jia Deng. Stacked hourglass networks for human pose estimation. In *European conference on computer vision*, pages 483–499. Springer, 2016. 1, 2, 7

[46] Dario Pavllo, Christoph Feichtenhofer, David Grangier, and Michael Auli. 3d human pose estimation in video with temporal convolutions and semi-supervised training. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7753–7762, 2019. 1, 2, 4, 7, 0

[47] William H Press and Saul A Teukolsky. Savitzky-golay smoothing filters. *Computers in Physics*, 4(6):669–672, 1990. 1, 3, 4, 5, 6, 8, 2

[48] Davis Rempe, Tolga Birdal, Aaron Hertzmann, Jimei Yang, Srinath Sridhar, and Leonidas J Guibas. Humor: 3d human motion model for robust pose estimation. *arXiv preprint arXiv:2105.04668*, 2021. 3, 2

[49] Joël Le Roux. An introduction to kalman filter. 2003. 3

[50] Amir Shahroudy, Jun Liu, Tian-Tsong Ng, and Gang Wang. Ntu rgb+ d: A large scale dataset for 3d human activity analysis. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1010–1019, 2016. 5, 8, 0

[51] Lei Shi, Yifan Zhang, Jian Cheng, and Hanqing Lu. Skeleton-based action recognition with multi-stream adaptive graph convolutional networks. *arXiv preprint arXiv:1912.06971*, 2019. 8, 0

[52] Ke Sun, Bin Xiao, Dong Liu, and Jingdong Wang. Deep high-resolution representation learning for human pose estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5693–5703, 2019. 1, 2, 7

[53] Ilya Tolstikhin, Neil Houlsby, Alexander Kolesnikov, Lucas Beyer, Xiaohua Zhai, Thomas Unterthiner, Jessica Yung, Andreas Steiner, Daniel Keysers, Jakob Uszkoreit, et al. Mlp-mixer: An all-mlp architecture for vision. *arXiv preprint arXiv:2105.01601*, 2021. 4

[54] Shashank Tripathi, Siddhant Ranade, Ambrish Tyagi, and Amit Agrawal. Posenet3d: Learning temporally consistent 3d human pose via knowledge distillation. In *2020 International Conference on 3D Vision (3DV)*, pages 311–321. IEEE, 2020. 2

[55] Shuhei Tsuchida, Satoru Fukayama, Masahiro Hamasaki, and Masataka Goto. Aist dance video database: Multi-genre, multi-dancer, and multi-camera database for dance information processing. In *ISMIR*, pages 501–510, 2019. 0

[56] Charles Van Loan. *Computational frameworks for the fast Fourier transform*. SIAM, 1992. 1

[57] Márton Véges and A Lőrincz. Temporal smoothing for 3d human pose estimation and localization for occluded people. In *International Conference on Neural Information Processing*, pages 557–568. Springer, 2020. 1, 2, 3, 6, 7

[58] 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 *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 601–617, 2018. 5, 8, 0

[59] 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 *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 13033–13042, 2021. 2

[60] Chunyu Wang, Haibo Qiu, Alan L Yuille, and Wenjun Zeng. Learning basis representation to refine 3d human pose estimations. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 8925–8932, 2019. 2

[61] Jingbo Wang, Sijie Yan, Yuanjun Xiong, and Dahua Lin. Motion guided 3d pose estimation from videos. *ArXiv*, abs/2004.13985, 2020. 1

[62] Weiyao Wang, Du Tran, and Matt Feiszli. What makes training multi-modal classification networks hard? In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12695–12705, 2020. 1

[63] Sijie Yan, Yuanjun Xiong, and Dahua Lin. Spatial temporal graph convolutional networks for skeleton-based action recognition. *arXiv preprint arXiv:1801.07455*, 2018. 8, 0

[64] Ian T Young and Lucas J Van Vliet. Recursive implementation of the gaussian filter. *Signal processing*, 44(2):139–151, 1995. 1, 3, 4, 5, 6, 7, 2

[65] Ailing Zeng, Xiao Sun, Fuyang Huang, Minhao Liu, Qiang Xu, and Stephen Ching-Feng Lin. Srnet: Improving generalization in 3d human pose estimation with a split-and-recombine approach. In *ECCV*, 2020. 1, 2, 0

[66] Ailing Zeng, Xiao Sun, Lei Yang, Nanxuan Zhao, Minhao Liu, and Qiang Xu. Learning skeletal graph

neural networks for hard 3d pose estimation. In *Proceedings of the IEEE International Conference on Computer Vision*, 2021. 1, 2

[67] Siwei Zhang, Yan Zhang, Federica Bogo, Marc Pollefeys, and Siyu Tang. Learning motion priors for 4d human body capture in 3d scenes. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 11343–11353, 2021. 2, 3

[68] Long Zhao, Xi Peng, Yu Tian, Mubbasir Kapadia, and Dimitris N Metaxas. Semantic graph convolutional networks for 3d human pose regression. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3425–3435, 2019. 1, 2

[69] Ce Zheng, Sijie Zhu, Matias Mendieta, Taojiannan Yang, Chen Chen, and Zhengming Ding. 3d human pose estimation with spatial and temporal transformers. *arXiv preprint arXiv:2103.10455*, 2021. 2

[70] Yi Zhou, Connelly Barnes, Jingwan Lu, Jimei Yang, and Hao Li. On the continuity of rotation representations in neural networks. *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5738–5746, 2019. 0, 3

Supplementary Material: SmoothNet: A Plug-and-Play Network for Refining Human Poses in Videos

This supplementary material presents more experimental details, including data description, implementation details, additional experimental results, ablation studies and description of the qualitative results.

1. Experimental Details

1.1. Dataset Description

– *Human3.6M* [20] consists of 3.6 million video frames with 15 actions from 4 camera viewpoints with 50 fps. 3D human joint positions are captured accurately from high-speed motion capture system. We can use the camera intrinsic parameters to calculate their accurate 2D joint positions. Following previous works [5, 39, 46, 65], we adopt the standard cross-subject protocol with 5 subjects (S1, S5, S6, S7, S8) as training set and another 2 subjects (S9, S11) as test set.

– *3DPW* [58] an in-the-wild dataset consisting of more than 51,000 frames with accurate 3D poses in challenging sequences with 30fps. It is always used to validate the effectiveness of model-based methods [9, 25, 29, 30].

– *AIST++* [34] is a new and more challenging dataset that comes from the AIST Dance Video DB [55]. It contains 1,408 sequences of 3D human dance motion with 60 fps, providing 3D human keypoint annotations and camera parameters for 10.1M images, covering 30 different subjects in 9 views. We follow the original settings to split the training and testing sets based on different subjects.

– *MPI-INF-3DHP* [40] contains both constrained indoor scenes and complex outdoor scenes, covering a greater diversity of poses and actions, where it is usually taken as a cross-dataset setting to verify the generalization ability of the proposed methods. That is, we use this dataset as the test set.

– *MuPoTS-3D* [42] is a test set of multi-person 3D human pose estimation, containing 20 indoor and outdoor video sequences. We also use it as the test set.

– *NTU-RGBD 60* [50] and *NTU-RGBD 120* [36] are the most used datasets for action recognition. Since they are captured by three Kinect V2 cameras concurrently,providing 3D skeletal positions of 25 body joints at each frame, many works perform skeleton-based action recognition on them. Specifically, *NTU-RGBD 60* contains 60 action classes and 56,880 video samples, and *NTU-RGBD 120* adds another 60 classes and another 57,600 video samples. We follow the common training and testing protocols as the methods [51, 63] split into Cross Subjects and Cross Viewpoints.

1.2. Implementation Details

For data preprocessing, we normalize 2D positions into [-1, 1] by the width and length of the videos, and we use root-relative 3D positions with the unit of meter, where they can range in [-1, 1]. For SMPL estimation, we use the original 6D rotation matrix without any normalization.

For the usage of motion representations, in the training stage, we use 3D positions to train SMOOTHNET by default. Because SMOOTHNET shares its weights as well as biases among different spatial dimension, it can be used directly cross different motion representations. In the inference stage, we can use the trained model to test on different motion representations. If the number of skeleton points is N, the outputs of 2D (C = $2 * N$) and 3D (C = $3 * N$) pose estimation are a series of 2D and 3D positions. The outputs of mesh recovery are the pose parameters as 6D rotation matrix [70] (C = $6 * N$), 10 shape parameters and 3 camera parameters. Different datasets have different N, such as N is 17 on Human3.6M, MPI-INF-3DHP and MuPoTS-3D, N is 24 on 3DPW and AIST++, N is 25 on NTU-RGBD 60 and NTU-RGBD 120.

For the usage of the AIST++ dataset [34], because of lacking of enough keypoints as constraints, we find that some inaccurate fitting from SMPLify [3] causes misleading supervision in 6D rotation matrices and high errors. We simply threw away the test videos with MPJPE (computed by the estimated results of VIBE and the given ground truth) greater than 170mm.

For training details, the initial learning rate is 0.001, and it decays exponentially with the rate of 0.95. We train the proposed model for 70 epochs using Adam [28] optimizer. The mini-batch size is 128. Our experiments can be conducted on a GPU with an NVIDIA GTX 1080 Ti.

For hyper parameters of filters, in the upper part of Table 1 in the main paper, we set the window size of Savitzky-Golay filter as 257 and the polyorder (order of the polynomial used to fit the samples) as 2 to obtain the comparable Acceleration errors with us. For Gaussian1d filter, we set the sigma (standard deviation for Gaussian kernel) as 4 and window size as 129. For One-Euro filter, the cutoff (the minimum cutoff frequency) is $1e^{-4}$ and the lag value (the speed coefficient) is 0.7. Meanwhile, in the lower part of Table 1, to obtain comparable MPJPEs, we set 17 as the window size with the polyorder as 2 for the Savitzky-Golay filter. We apply 11 as the window size with sigma as 2 the for Gaussian1d filter, and modify the cutoff to 0.5 for One-Euro filter. In addition, we follow the common tools

to implement *One-Euro*[2], *Savitzky-Golay*[3] and *Gaussian1d filters*[4].

2. Experimental Analyses

2.1. Rethink Existing End-to-End Frameworks

To explore the bottleneck of existing methods on optimizing precision and smoothness concurrently, we take experiments on popular 3D skeleton-based methods [39, 46] and SMPL-based methods [29, 30]. In terms of the single-frame approaches (T = 1), we implement the simple baseline [39] for 3D pose estimation tested on the Human3.6M dataset and remove GRUs in VIBE [29] for body recovery tested on the 3DPW dataset. For multi-frame methods (T > 1), we apply the video-based 3D pose estimator [46], conducting temporal convolution networks with dilated convolution along the time axis and the official VIBE. The difference between single-frame methods and multi-frame methods is different from aggregation strategies along the temporal dimension. Two evaluation metrics, mean position errors (MPJPE) and acceleration errors (*Accel*), are used.

Figure 1 illustrates the training and testing performance for both MPJPE and *Accel*. For the single-frame models (*T = 1*) [30, 39], we observe that the position errors decrease, but the acceleration errors become larger as the epochs increases, indicating that the single-frame methods extracting only spatial information are likely to sacrifice smoothness in exchange for localization performance improvement. It is important to exploit temporal information explicitly.

For multi-frame approaches [29, 46] *(T = 27)*, they make use of temporal information by TCNs [46] and GRUs [29] respectively and improve both precision and smoothness. Yet, their loss function is applied to each frame and their smoothness is still far from satisfactory, which is intuitively not beneficial for smoothness optimization.

Accordingly, we further add an acceleration (*Acc.*) loss on the per-frame L1 loss, which constrains the estimated acceleration to be as close as to the ground truth's acceleration. As shown in the right ones (*T = 27 w/ Acc. loss)*, although the acceleration errors decrease further, the position errors increase instead. It implies that it is hard to achieve optimal precision and smoothness simultaneously within an end-to-end framework. The reasons behind this may lie in that temporal and spatial information may generalize and overfit at different rates as two different modalities [62]. This observation motivates us to adopt the refinement paradigm.

To quantitatively explore the combination strategies of refinement methods with existing backbones, such as training two models together (the one-stage strategy) or training them separately (the two-stage method), we try each of them on 3d pose estimation and body recovery. Specifically, if SMOOTHNET is trained together with the backbones in an end-to-end manner (w/ B), it belongs to the one-stage strategy. And if SMOOTHNET is trained separately, it is called the two-stage method. As shown in Table 1, we can observe that (i) the temporal model [46] with multiple frames as inputs will gain in both *Accel* (smoothness) and MPJPE (precision), but the computational costs will be increased; (ii) adding acceleration loss or SMOOTHNET in an end-to-end way can benefit *Accel* but harm MPJPE; (iii) adding intermediate L1 supervision between the backbones and SMOOTHNET (w/ B $\circ$) shows a slight drop in performance, but after adding an additional acceleration loss will improve both metrics. Compared with one-stage strategies, two-stage solutions with a refinement network show their strengths in boosting both smoothness and precision with a lightweight SMOOTHNET.

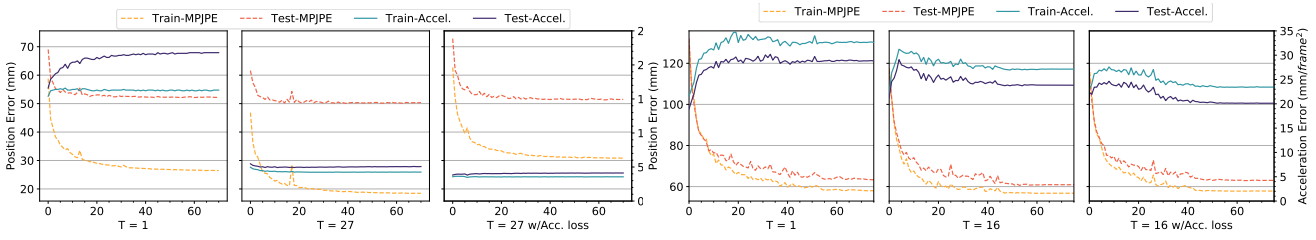
2.2. More Comparison with Filters

In Section 4.2.1 of the main paper, we compare the performance with filters on human body recovery. We further show more results on the tasks of 2D pose estimation and 3D pose estimation on Human3.6M. In Table 3, the upper

[2] https://github.com/mkocabas/VIBE/blob/master/lib/utils/one_euro_filter.py

[3] https://docs.scipy.org/doc/scipy/reference/generated/scipy.signal.savgol_filter.html

[4] https://docs.scipy.org/doc/scipy/reference/generated/scipy.ndimage.gaussian_filter1d.html



(a) 3D Skeleton-based Methods [39, 46]

(b) SMPL-based Methods [29]

Figure 1. Comparison mean position and acceleration errors during training and testing stages of single-frame (*T = 1*) [39] and temporal (*T = 27* or *T = 16*) [29, 46] pose estimation and mesh recovery methods. *w/Acc. loss* adds an acceleration loss in the training stage. In (b) *T = 1*, we simply remove GRUs and set sequence length as 1.

Table 1. Comparison of the 3D pose estimation results from VideoPose3d [46] of different training strategies on Human3.6M. $\times$ means acceleration loss added in the loss function. B means to add SMOOTHNET behind the origin network trained in an end-to-end manner. $\circ$ adds an intermediate L1 supervision between the backbone and SMOOTHNET.

	Strategy	*Accel*	MPJPE	Params.
Backbones	In = 1	21.99	52.40	6.39M
	In = 27	5.07	50.39	8.61M
	In = 27 w/ $\times$	4.12	51.48	8.61M
	In = 27 w/ B	2.78	52.65	8.65M
	In = 27 w/ B $\times$	2.87	52.18	8.65M
	In = 27 w/ B $\circ$	5.46	51.06	8.65M
	In = 27 w/ B $\circ$ $\times$	2.69	50.94	8.65M
Ours	In = 1 w/ ours	0.90	50.21	**0.03M**
	In = 27 w/ ours	**0.89**	**49.87**	**0.03M**

Table 2. Comparison results of the body recovery from VIBE [29] of different training strategies on 3DPW. $\times$ means acceleration loss added in the loss function. B means to add SMOOTHNET behind the backbones trained in an end-to-end manner.

	Strategy	*Accel*	MPJPE	PA-MPJPE	MPJVE
Backbones	In = 1	32.69	84.54	57.94	102.05
	In = 16	27.11	82.06	56.77	99.76
	In = 16 $\times$	23.43	84.51	57.81	101.62
	In = 16 w/ B	25.79	86.56	59.93	105.08
Ours	In = 1 w/ ours	6.12	82.98	57.27	100.67
	In = 16 w/ ours	**6.05**	**81.42**	**56.21**	**98.83**

half table of each task compares the results of filters with the closest MPJPEs to ours, and the lower half table compares the performance of filters with the most similar *Accel* to ours. We can conclude that our approach achieves better performance on both precision and smoothness, validating that the learnable network with long-range effective receptive field will be a better solution.

Table 3. Comparison of most used filters with different estimated poses from CPN [8] and FCN [39] on Human3.6M.

	Method	*Accel*	MPJPE	PA-MPJPE	Test FPS
2D Pose	CPN [8]	2.91	6.67	5.18	-
	w/One-Euro [6]	2.32	6.64	5.14	32.89k
	w/Savitzky-Golay [47]	0.29	6.48	4.99	884.96k
	w/Gaussian1d [64]	0.17	6.48	4.97	46.73k
	w/One-Euro [6]	0.27	11.06	6.46	34.34k
	w/Savitzky-Golay [47]	0.17	7.36	5.71	855.00k
	w/Gaussian1d [64]	**0.14**	6.81	5.17	228.22k
	w/Ours	**0.14**	**6.31**	**4.81**	**978.62k**
3D Pose	FCN [39]	19.18	54.57	42.23	-
	w/One-Euro [6]	8.41	54.21	42.03	29.55k
	w/Savitzky-Golay [47]	1.30	53.52	41.55	632.24k
	w/Gaussian1d [64]	1.18	53.54	41.51	32.36k
	w/One-Euro [6]	0.99	123.15	83.23	30.98k
	w/Savitzky-Golay [47]	0.98	79.47	61.28	609.76k
	w/Gaussian1d [64]	0.93	55.14	43.22	169.76k
	w/Ours	**0.91**	**52.05**	**40.54**	**675.68k**

2.3. Smoothness on Synthetic Data

Due to the lack of pairwise labeled data, some approaches [7, 14, 15, 17, 38, 48] for Mocap sensors denoising verify the validity of their approaches on synthetic noise, like Gaussian noises. We follow their methods to generate the noisy poses, adding different levels of Gaussian noises on the ground truth data. We take the Human3.6M dataset as an example. In the training stage, we generate Gaussian noises with the probability p and noise variance σ on the ground truth 2D or 3D positions for 2D or 3D pose estimation respectively as synthetic training data. SMOOTHNETcan be trained on these synthetic data. In the inference stage, we also add the same noise level to the test set as the synthetic test data to test the corresponding SMOOTHNET. Table 4 gives the corresponding results of our model. SMOOTHNET can refine the noises/jitters at a large margin without any spatial correlations. For instance, in terms of 3D pose estimation, either as the variance of Gaussian noises increase from 10mm to 100mm or the probability changing from 0.1 to 0.9, SMOOTHNET can decrease *Accel* and MPJPE at a large margin. Those results indicates SMOOTHNET will be also beneficial to remove different synthetic noises.

Table 4. Comparison of 3D pose with different synthetic noises from *Gaussion Noise* on Human3.6M. p is the probability of adding noise, and σ means the variance. pix. is the abbreviation of pixel.

	Gaussion Noise	In *Accel*	Out *Accel*	In *MPJPE*	Out *MPJPE*
2D Pose	p = 0.5, σ = 10 pix.	10.10	**0.20**	3.56	**0.83**
	p = 0.5, σ = 50 pix.	50.53	**0.35**	17.80	**2.02**
	p = 0.5, σ = 100 pix.	101.06	**0.31**	35.59	**1.42**
	p = 0.1, σ = 50 pix.	14.31	**0.19**	3.90	**0.67**
	p = 0.5, σ = 50 pix.	50.53	**0.35**	17.80	**2.02**
	p = 0.9, σ = 50 pix.	72.26	**0.57**	28.97	**6.00**
3D Pose	p = 0.5, σ = 10mm	26.25	**0.84**	9.68	**3.54**
	p = 0.5, σ = 50mm	131.25	**1.55**	48.42	**7.00**
	p = 0.5, σ = 100mm	262.49	**1.24**	96.84	**20.38**
	p = 0.1, σ = 50mm	40.68	**1.03**	11.46	**2.46**
	p = 0.5, σ = 50mm	131.25	**1.55**	48.42	**7.00**
	p = 0.9, σ = 50mm	184.32	**2.10**	74.46	**16.85**

2.4. More Ablation Study

Impact on Loss Function. As mentioned in Section 3.3 of the main paper, we use L_{pose}+L_{acc} as our final objective function. Here we explore how the loss functions affect the performance in Table 5. First, we find that only single-frame supervision L_{pose} would be slightly worse than our result by 5.51% in *Accel*, while the MPJPEs are competitive. It shows the precision can be optimized well by the L_{pose}. Next, only with L_{acc} will make all results worst. Last, adding L_{pose} and L_{acc} together to train the SMOOTHNET will benefit both smoothness and precision, proving that L_{acc} companies with L_{pose} can play its smooth role.

Table 5. Comparison of refined results by different loss functions based on the outputs of the SMPL-based method EFT [22] on the 3DPW dataset.

Method	*Accel*	MPJPE	PA-MPJPE
EFT	32.49	90.33	52.19
L_{pose}	6.12	85.23	**50.30**
L_{acc}	7.63	446.54	356.61
L_{pose}+L_{acc}	**5.80**	**85.16**	50.31

Impact on Motion Representation. Motivated by this natural smoothness characteristic, we can unify various continuous representations and make SMOOTHNET generalize across these representations. In particular, 2D, 3D position, and 6D rotation matrix are continuous representations of the same space in neural networks. In contrast, the rotation representations as axis-angle or quaternion are discontinuous in the real Euclidean spaces [70], which may be hard for neural networks to learn. Accordingly, we explore the effects of these representations used to train SMOOTHNET on EFT [22]. Table 6 shows the training results on each motion representation. We can see that the representation as axis-angle or quaternion obtains worse results on smoothness and precision. They may encounter some sudden changes/flips leading to poor results due to the discontinuity of the expression. Instead, the 6D rotation matrix and 3D position will be more suitable to learn and improve all metrics. Furthermore, 3D positions reach the best performance by decreasing 82.15% in *Accel*, 5.72% in *MPJPE*, and 3.60% in *PA-MPJPE*.

Table 6. Comparison of refined results trained by different motion representations based on the outputs of EFT [22] on the 3DPW dataset.

Method	*Accel*	MPJPE	PA-MPJPE
EFT	32.49	90.33	52.19
Angle-Axis	77.89	172.17	51.38
Quaternion	28.50	91.23	51.03
6D Rotation	6.23	87.16	50.86
3D Position	**5.80**	**85.16**	**50.31**

Last, to explore whether there is also better generalization between different continuous modalities, such as 3D position and 6D Rotation, cross-modality tests were carried out demonstrated in Table 7. We can summarize some observations: (i) when tested across modalities, all results will be worse relative to the modality the model trained on; (ii) SMOOTHNET trained in 3D coordinates, smoothed directly over the representation of the 6D rotation matrix, can achieve even better performance than training on the 6D rotation matrix itself. Hence, these results motivate us to use 3D positions as supervision by default, where the 3D positions contain more information than 2D positions and their ground truth are usually more precise than 6D rotation matrix (explicitly find in the AIST++ dataset, like Figure 2).

Table 7. Comparison of refined results by *cross motion representations testing* based on the outputs of EFT [22] on the 3DPW dataset. *Cross-Test* means training the SMOOTHNET on a motion representation while testing it on another modality directly.

Method	*Accel*	MPJPE	PA-MPJPE
EFT	32.49	90.33	52.19
6D Rotation	6.23	87.16	50.86
Cross-Test on 3D Position	7.10	88.13	51.79
3D Position	**5.80**	**85.16**	**50.31**
Cross-Test on 6D Rotation	5.91	86.54	50.72

Effect of Normalization Strategies. Normalization is an effective way to calibrate biased errors and improve the generalization ability. As a plug-and-play network, we also explore how different normalization strategies influence the results, especially the generalization ability. In the main paper, we do not use any normalization by default. In Table 8, we compare three normalization strategies. Particularly, *w/o Norm.* denotes taking the original estimated results without normalization, and *Sequence Norm.* indicates normalizing each input axis $\hat{\mathbf{Y}}_i$ with means and variances computed from input sequences along axis. Because the estimated inputs are always noisy and the bias shift between the training data and testing data, the above normalization methods will be affected. Instead, using the mean and variance from the ground truth (with †) along each axis can avoid such influences and we can explore the upper bound performance under the *Sequence Norm.* normalization.

Hence, we first compare the performance of different normalization based on the outputs of EFT [22] on the 3DPW dataset in the upper part of Table 8. We can discover that the smoothing ability for all normalizations is similar, and the main difference lies in the degree of biased error removal. In specific, under the *Sequence Norm.* † normalization, the MPJPE can decreases from $85.16mm$ to $61.65mm$, improved by 27.5%. To explore the generalization ability cross backbones, we further test SMOOTHNETtrained on EFT-3DPW on TCMR [9]-3DPW. From the lower part of the table, we can get similar conclusions as above. In specific, SMOOTHNETcan reduce *Accel* from $6.77mm/frame^2$ to about $6mm/frame^2$, and the upper bound of MPJPE can be $68.51mm$ (improvement by 20.8%) from the refinement stage.

3. Qualitative Results

As jitters seriously affect visual effect, we visualization a number of results according to different tasks, like 2D pose estimation, 3D pose estimation, and model-based body recovery. For 2D and 3D pose estimation, we show two

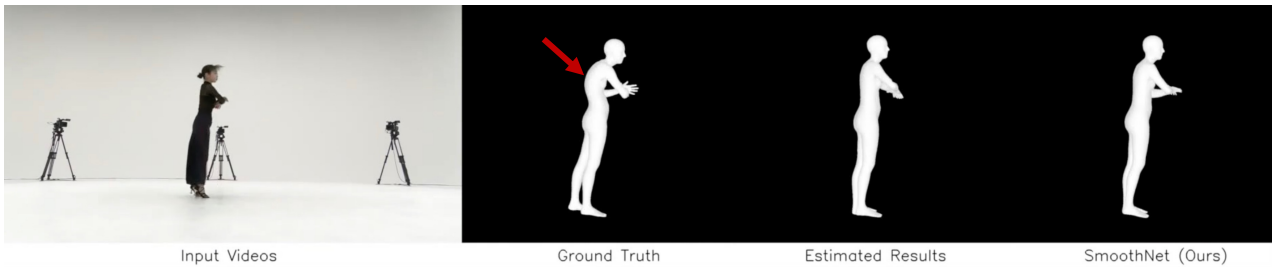


Figure 2. Comparison the results of the ground truth, VIBE [29] with VIBE w/ SMOOTHNET on AIST++ dataset.

Table 8. Comparison of the results of different normalization based on the outputs of EFT [22] and *cross-backbone testing* on the outputs of TCMR [9] on 3DPW dataset. † means using the same mean and variance as the ground truth to explore the upper bound performance.

Method	*Accel*	MPJPE	PA-MPJPE
EFT [22]	32.49	90.33	52.19
w/o Norm.	**5.80**	85.16	50.31
Sequence Norm.	5.82	88.21	51.06
Sequence Norm. †	**5.80**	**61.65**	**44.28**
TCMR [9]	6.77	86.46	52.67
w/o Norm.	**5.91**	86.04	52.42
Sequence Norm	6.00	86.34	52.87
Sequence Norm †	5.92	**68.51**	**49.15**

kinds of actions on Human3.6M respectively with the corresponding *Accel* and MPJPE for each frame. The estimated 2D poses are from the single-frame SOTA method RLE [32], and the estimated 3D poses are from the single-frame method FCN [39]. For model-based methods, the estimated results come from VIBE [29] on AIST++ dataset and SPIN [30] on 3DPW dataset.

We can observe that the jitters in a video are highly-unbalanced, where most frames suffer from slight jitters while long-term significant jitters will be accompanied with large biased errors. SMOOTHNET can relieve not only small jitters but long-term jitters well. And it can boost both smoothness and precision significantly. Specifically, unlike low-pass filters [14, 19, 47], our method can estimate the high-frequency movements well, like the action *Posing* (3d_pose/smooth3d_SubS9_ActPosing_Cam0_SmoothNet.mp4). Finally, we observe that some ground truth of AIST++ is not quite accurate and smooth, especially for the 6D rotation matrices from SMPL fitting with less constraints. Instead, Their 14 skeletal 3D positions are more precise from multi-view 2D keypoints and camera parameters, which will be more suitable as the supervision. Meanwhile, SMOOTHNET is able to cross modality to smooth results from 3D positions to rotation matrices. Moreover, the data-driven models, like VIBE [29], are basically no back bulge, illustrated the red arrow in Figure 2. Thus, our method can benefit from both the precise 3D positions from the ground truth of AIST++ at training stage and the relatively better estimated rotation matrices from VIBE as our inputs at inference stage to obtain more precise and smooth results.