Kinematic-aware Hierarchical Attention Network for Human Pose Estimation in Videos

Kyung-Min Jin[1], Byoung-Sung Lim[1], Gun-Hee Lee[2], Tae-Kyung Kang[1], and Seong-Whan Lee[1]

[1]Department of Artificial Intelligence, Korea University
[2]Department of Computer Science and Engineering, Korea University
{km_jin, bs_lim, gunhlee, tk_kang, sw.lee}@korea.ac.kr

Abstract

Previous video-based human pose estimation methods have shown promising results by leveraging aggregated features of consecutive frames. However, most approaches compromise accuracy to mitigate jitter or do not sufficiently comprehend the temporal aspects of human motion. Furthermore, occlusion increases uncertainty between consecutive frames, which results in unsmooth results. To address these issues, we design an architecture that exploits the keypoint kinematic features with the following components. First, we effectively capture the temporal features by leveraging individual keypoint's velocity and acceleration. Second, the proposed hierarchical transformer encoder aggregates spatio-temporal dependencies and refines the 2D or 3D input pose estimated from existing estimators. Finally, we provide an online cross-supervision between the refined input pose generated from the encoder and the final pose from our decoder to enable joint optimization. We demonstrate comprehensive results and validate the effectiveness of our model in various tasks: 2D pose estimation, 3D pose estimation, body mesh recovery, and sparsely annotated multi-human pose estimation. Our code is available at `https://github.com/KyungMinJin/HANet`.

1. Introduction

Human pose estimation, estimating each keypoint location from images, has long been studied in the computer vision field. It has been extended from identifying a person's location to action understanding [8, 20, 45, 1, 43, 42] or manipulating various computer interfaces with human movements, thereby becoming a core technology for various applications. In addition, with the advent of deep learning, it became possible to locate each keypoint robustly, and methods [44, 56, 6, 10, 12, 53, 23] have shown remarkable results. However, existing methods still fail to address highly-occluded cases, such as the presence of multiple persons or motion blur where certain body parts move quickly. Thus, their results include high-frequency jitters.

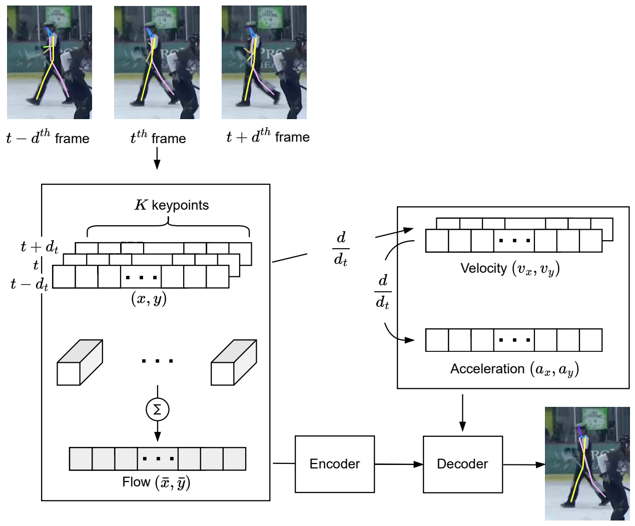


Figure 1. The overview of HANet with keypoint kinematic features. First, we compute flow as a weighted sum of keypoint coordinates from current, previous, and next frames and hierarchically encode the temporal features of body motion. In addition, we further consider velocity v and acceleration a from consecutive keypoint coordinates as supplement input of the decoder.

We found two significant issues in pose estimation for video. First, the large positional difference between consecutive frames' poses shares less visual similarity, resulting in high-frequency jitters. Second, occluded or blurred regions bring spatial ambiguities that significantly drop model performance, and make the task more challenging.

Existing methods [33, 58, 55, 34, 18, 40] tend to focus on one of the issues instead of both; reducing jitter by focusing on the temporal aspect or addressing occlusion by increasing the model complexity to capture spatial features well. In video-based pose estimation, existing methods typically use recurrent neural networks (RNN) [33], 3D convolutional neural networks (3D CNN) [49], and transformers [58, 55, 34, 18] to exploit temporal features. However, they show limitations where an input video includes severe occlusion or motion blur. Although there are methods [31, 5, 49, 40, 39] using 2D CNN to store spatio-temporal dependencies within parameters, they do not accurately comprehend the temporal features of human motion.

In this paper, we propose a novel architecture named **HANet** (Kinematic-aware Hierarchical Attention Network) that accurately refines human poses in video. We introduce a hierarchical network that effectively addresses jitter and occlusion by exploiting keypoints' movement, as shown in Fig. 1. First, we compute each keypoint's kinematic features: flow (the track of keypoint movement), velocity, and acceleration. Through these features, our framework kinematically learns the temporal aspect of keypoints to focus on frequently occluded or fast-moving body parts such as wrists and ankles. Second, the proposed hierarchical encoder projects multi-scale feature maps by exponentially increasing the number of channels and captures spatio-temporal features. We embed multi-scale feature maps to generate positional offsets which we add to refine input poses, estimated by off-the-shelf methods. Then, our decoder processes the refined input poses with keypoint velocity and acceleration to estimate the final poses. Lastly, we provide a cross-supervision that cooperatively optimizes refined input poses and final poses by choosing an online learning target along their training losses. Through this work, our method significantly reduces jitter and becomes robust to occlusion while improving performance. In summary, our main contributions are as follows:

- We propose a novel approach HANet using keypoint kinematic features, following the laws of physics. Our method effectively mitigates jitter and becomes robust to occlusion with these proposed features.

- We propose a hierarchical transformer encoder that incorporates multi-scale spatio-temporal attention. We leverage all layers' multi-scale attention maps and improve performance on sparse supervision.

- We propose online mutual learning that enables joint optimization of refined input poses and final poses by selecting online targets along their training losses.

- We conduct extensive experiments and demonstrate our framework's applicability on various tasks: 2D and 3D pose estimation, body mesh recovery, and sparsely-annotated multi-human 2D pose estimation.

2. Related Work

2.1. Pose Estimation in Images

Modern approaches for single-image pose estimation, one of the fundamental pattern recognition problems [11, 24, 25, 26] in computer vision field, are typically based on 2D CNN. Early methods [46] directly regress the joint coordinates from the images; however, recent approaches [44, 6, 50, 36, 38] have widely adopted joint position representations with maximum values from a heatmap that denotes joint presence probability. These methods can be divided into two ways: bottom-up and top-down. First, viewing a human skeleton as a graph, bottom-up approaches [6, 41] detect individual body parts and assemble these components into the person. Recently, many top-down methods [38, 50, 44] have been proposed that first perform human detection using an object detector and estimate poses for each individual. CPM [50] iteratively refines the output of each step, and Hourglass [38] adjusts the number of channels. HRNet [44] has achieved higher performance than ResNet [13] by maintaining high-resolution feature maps through multi-scale fusion and replaced ResNet, which served as the backbone of pose estimation. These top-down methods have significantly improved performance and show remarkable results, but they include high-frequency jitter when applied to video data.

With the success of the attention-based method in natural language processing (NLP), transformers combined with CNNs have brought inspiration for new approaches [3, 9, 54, 58, 18, 34, 14, 29, 19] to computer vision field. Thanks to the characteristic of self-attention, which shows superior performance in modeling long-range dependencies, the transformer can be used to capture spatio-temporal relations. With the advent of ViT [9], which outperformed CNN-based counterparts in classification on large image datasets [30], several methods [29, 54] applied transformers to pose estimation. Transpose [54] captures long-range relationships and reveals spatial dependencies that provide evidence of how the model handles occlusion. Tokenpose [29] tokenizes each keypoint and computes an attention map between keypoints. However, they increased the model size and the resolution of input images (or heatmaps), making it difficult to apply transformer for videos with many frames.

2.2. Pose Estimation in Videos

Fully-annotated benchmark datasets [17, 15, 48, 27] for videos are suitable for learning temporal features because they provide supervision for all frames and contain a few people with less occlusion. Meanwhile, [49, 55, 33, 40, 57] directly process 2D or 3D positions and capture temporal features of body motion using RNN or 3D CNN. LPM [33] extends CPM [50] using long short-term memory (LSTM) to capture temporal dependencies between poses.

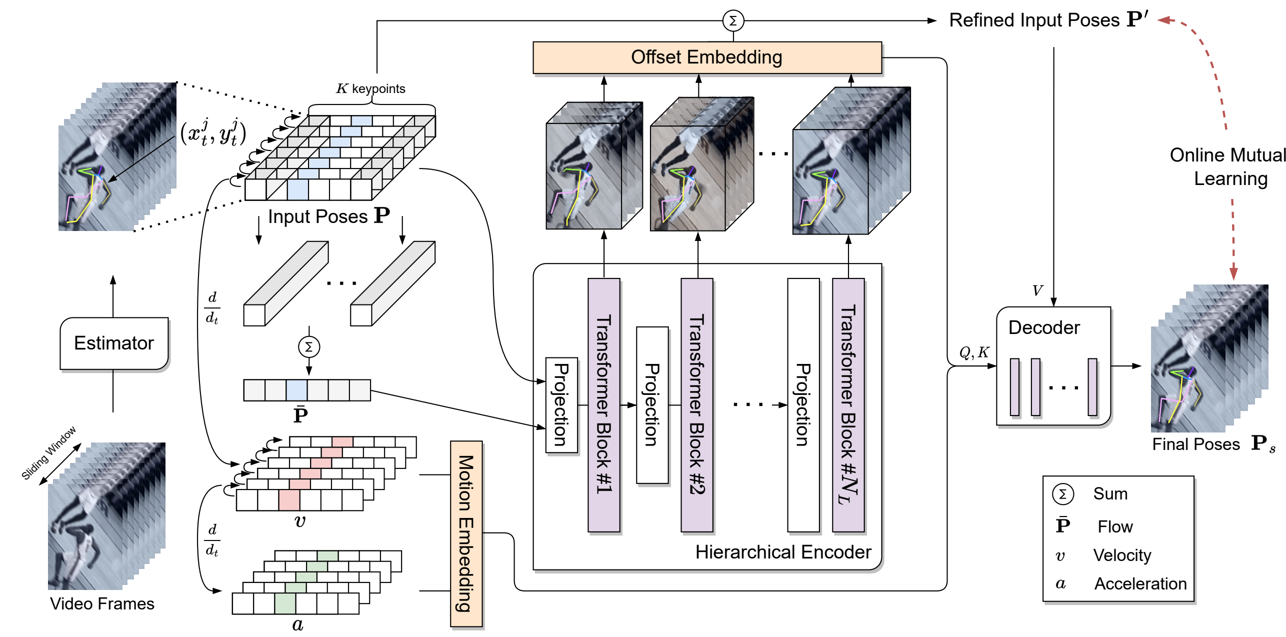


Figure 2. The overall architecture of HANet. First, keypoint kinematic features (flow, velocity, and acceleration) are computed from input poses. Then, our encoder captures the spatio-temporal relationships of keypoint movement. Each encoder layer features are embedded to offsets which refine input poses. Refined input poses are decoded with keypoint velocity and acceleration to produce final poses.

3D HRNet [49] intuitively uses temporal convolutions and learns the correlation between consecutive frames' keypoints by extending HRNet [44]. Recently, several approaches [58, 34, 18, 55] have also used vanilla transformers [47] for pose estimation in videos. [58] encodes 3D information using a spatial and temporal encoder. Also, [34, 14] use transformer to fuse multi-view features. Recently, DeciWatch [55] proposes a method that efficiently watches sparsely sampled frames with transformer taking advantage of the continuity of human motions without performance degradation. However, these methods may not be useful in many real-world scenarios, as they do not perform well in crowded scenes with severe occlusion.

Many CNN-based approaches [52, 49] have been proposed to address the occlusion issues in videos. The basic idea of CNN-based pose estimation methods for video is to encode spatio-temporal relationships between frames using convolution. However, there is a fundamental problem in that the receptive field is limited, which makes it challenging to capture long-range spatio-temporal relationships between joint positions. In addition, recent methods leverage memory-intensive heatmap-based estimation processes, making it challenging to sufficiently consider the temporal aspect of human motion. In contrast, we directly use input poses from estimators [51, 31, 21, 37, 22, 28] to train our framework HANet, which efficiently reduces memory usage and capturing the spatio-temporal dependencies by proposed hierarchical transformer encoder.

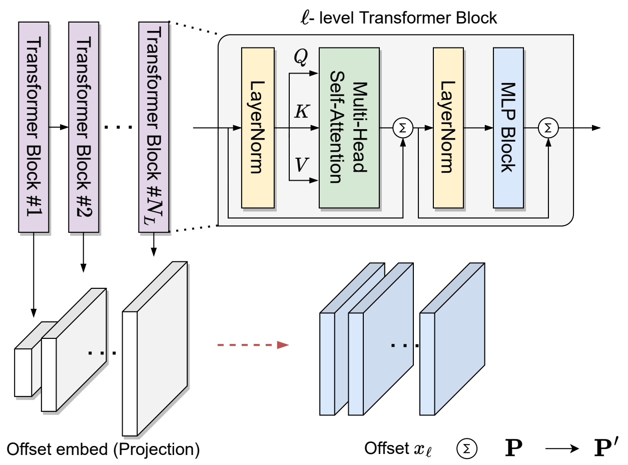
3. Method

We propose a novel framework that leverages keypoint kinematic features among consecutive frames to reduce jitter and learn temporal features of body motion. First, we use input poses $\mathbf{P} \in \mathbb{R}^{T/N \times (K \cdot D)}$ estimated from the off-the-shelf estimators [31, 51, 37, 21, 22, 28] to compute keypoint kinematic features within a sliding window of length T. Here, K is the number of keypoints, D denotes the dimension of each keypoint coordinates, and N is the interval [55] that we sampled poses in a sliding window. Then, we construct hierarchical transformer architecture that processes consecutive poses. The hierarchical encoder increases the number of channels exponentially and projects multi-scale feature maps to positional offsets that refine input poses $\mathbf{P}$. Finally, our decoder processes the offsets, keypoint velocity, and acceleration as keys and queries with refined input poses as values to estimate final poses. We discuss each component in more detail below.

3.1. Keypoint Kinematic Features

We consider keypoint kinematic features within a sliding window to address jitter, a fundamental problem of pose estimation in videos. In this paper, we leverage three kinematic features, a continual aspect of human motion, obtained from previous, current, and next frames' poses.

Keypoint Flow. First, we compute the track of keypoint movement using consecutive poses' coordinates $\mathbf{P}_t$ at each



ℓ- level Transformer Block
Transformer Block #1
Transformer Block #2
Transformer Block #N_L
LayerNorm
Q
K
V
Multi-Head Self-Attention
LayerNorm
MLP Block
Offset embed (Projection)
Offset x_ℓ
$\mathbf{P} \longrightarrow \mathbf{P}'$

Figure 3. Visualization of the proposed hierarchical encoder. Each encoder layer iteratively captures a spatio-temporal correlation between keypoints. Then attention map of each layer is linearly projected to produce positional offsets $\boldsymbol{x}_\ell$. We can get refined input poses $\mathbf{P}'$ by a weighted sum of offsets $\mathbf{X}$ and input poses $\mathbf{P}$.

frame $t \in \{1, 2, \dots, T\}$. We denote the keypoint flow as $\bar{\mathbf{P}}_t$ and define it as,

$$\bar{\mathbf{P}}_t = (\mathbf{P}_t + \mathbf{P}_{t+d_t} + \mathbf{P}_{t-d_t})/3, \tag{1}$$

where d_t denotes an interval from the previous and next poses to the current poses. A flow $\bar{\mathbf{P}}_t$ can be interpreted as a moving average or a track of keypoint movement.

Keypoint Velocity and Acceleration. We further consider kinematic information from the perspective of keypoint velocity and acceleration between consecutive input poses. We define the keypoint velocity and acceleration as below:

$$\begin{aligned} \boldsymbol{v}_t &= (\mathbf{P}_t - \mathbf{P}_{t-d_t})/d_t, \\ \boldsymbol{a}_t &= (\boldsymbol{v}_t - \boldsymbol{v}_{t-d_t})/d_t. \end{aligned} \tag{2}$$

We exploit them in our decoder as first and second-order derivative features to estimate final poses.

3.2. Hierarchical Encoder

After keypoint kinematic features are computed, the proposed hierarchical transformer encoder generates multi-scale pose features $\hat{\mathbf{Z}}_\ell \in \mathbb{R}^{T/N \times (C \cdot 2^\ell)}$ at each layer ℓ, representing spatio-temporal attention maps. Here, C denotes the initial embedding dimension. Long-range and short-range attention maps model the distribution of large and small body movements, respectively, and resolve occlusion of keypoints spatially within a frame or temporally specific frame in a video.

First, we embed previous, current, next poses, and keypoint flow $\bar{\mathbf{P}}_t$ by stacking them in keypoint dimension. This can be defined as:

$$\mathbf{Z}_0 = (\bar{\mathbf{P}}_t; \mathbf{P}_t; \mathbf{P}_{t+d_t}; \mathbf{P}_{t-d_t})\mathbf{W}_0, \tag{3}$$

where $\mathbf{W}_0 \in \mathbb{R}^{(K \cdot 4D) \times C}$ is a projection matrix and $\mathbf{Z}_0$ denotes the initial embedding features. $\mathbf{Z}_0$ is then projected to $\mathbf{Q}_0$, $\mathbf{K}_0$, and $\mathbf{V}_0$ representing queries, keys, and values. The hierarchical encoder exponentially expands queries, keys, and values using $\mathbf{W}_\ell \in \mathbb{R}^{C \cdot 2^{\ell-1} \times C \cdot 2^\ell}$. These are given by,

$$\begin{aligned} \mathbf{Q}_{\boldsymbol{\ell}} &= (\mathbf{Z}_{\ell-1})\mathbf{W}_\ell + \mathbf{E}_{pos,\ell}, \quad \boldsymbol{\ell} = 1 \dots N_L, \\ \mathbf{K}_{\boldsymbol{\ell}} &= (\mathbf{Z}_{\ell-1})\mathbf{W}_\ell + \mathbf{E}_{pos,\ell}, \quad \boldsymbol{\ell} = 1 \dots N_L, \\ \mathbf{V}_{\boldsymbol{\ell}} &= (\mathbf{Z}_{\ell-1})\mathbf{W}_\ell + \mathbf{E}_{pos,\ell}, \quad \boldsymbol{\ell} = 1 \dots N_L, \end{aligned} \tag{4}$$

where $\mathbf{E}_{pos,\ell} \in \mathbb{R}^{T/N \times (C \cdot 2^\ell)}$ is an ℓ-level positional embedding and N_L denotes the number of encoder layers. Then, the hierarchical encoder generates a tuple of multi-scale feature maps $(\mathbf{Z}_0, \mathbf{Z}_1, \dots, \mathbf{Z}_{N_L})$. We compute attention maps as a vanilla transformer [47] and apply LayerNorm (LN) [4] before every multi-head self-attention (MSA) and multi-layer perceptron (MLP) block. Leaky ReLU [35] is used for the MLP activation function. If we simply express projected ℓ-level queries, keys, and values as $\mathbf{Z}_\ell$, this process can be expressed as:

$$\begin{aligned} \overline{\mathbf{Z}}_\ell &= \mathrm{MSA}(\mathrm{LN}(\mathbf{Z}_\ell)) + \mathbf{Z}_\ell, \\ \hat{\mathbf{Z}}_\ell &= \mathrm{MLP}(\mathrm{LN}(\overline{\mathbf{Z}}_\ell)) + \overline{\mathbf{Z}}_\ell, \end{aligned} \tag{5}$$

where $\hat{\mathbf{Z}}_\ell$ is the encoded multi-scale spatio-temporal features. Then, we project the encoded multi-scale features to offset embeddings that coincide with the dimension of input poses $K \cdot D$, using 1D convolution $\mathbf{Conv1D}_\ell$. We denote the projected positional offsets as $\boldsymbol{x}_\ell$, as illustrated in Fig. 3. A weighted sum of $\boldsymbol{x}_\ell$ is added to the input pose $\mathbf{P}$ and can be defined as follows:

$$\begin{aligned} \boldsymbol{x}_\ell &= \mathbf{Conv1D}_\ell(\hat{\mathbf{Z}}_\ell), \\ \mathbf{P}' &= \mathbf{P} + \frac{1}{N_L}\sum_{\ell=0}^{N_L} \boldsymbol{x}_\ell, \end{aligned} \tag{6}$$

where $\mathbf{P}'$ denotes the refined input poses.

3.3. Decoder

Given the weighted sum of offsets $\mathbf{X}$, the decoder processes them with $\boldsymbol{v}_t$ and $\boldsymbol{a}_t$ as keys and queries with $\mathbf{P}'$ as values, to estimate the final poses $\mathbf{P}_s$. First, we use $\boldsymbol{v}_t$ and $\boldsymbol{a}_t$ as derivative features with 1D convolutional layers, which can be defined as,

$$\begin{aligned} \mathbf{S}_v &= w_v \boldsymbol{v}_t + b_v, \\ \mathbf{S}_a &= w_a \boldsymbol{a}_t^2 + b_a, \end{aligned} \tag{7}$$

to represent the derivative positional features $\mathbf{S}_v$ and $\mathbf{S}_a$, where $w_v, w_a \in \mathbb{R}^{C \times C}$ and $b_v, b_a \in \mathbb{R}^C$ are weights and biases. Then, we stack them along keypoint channels and

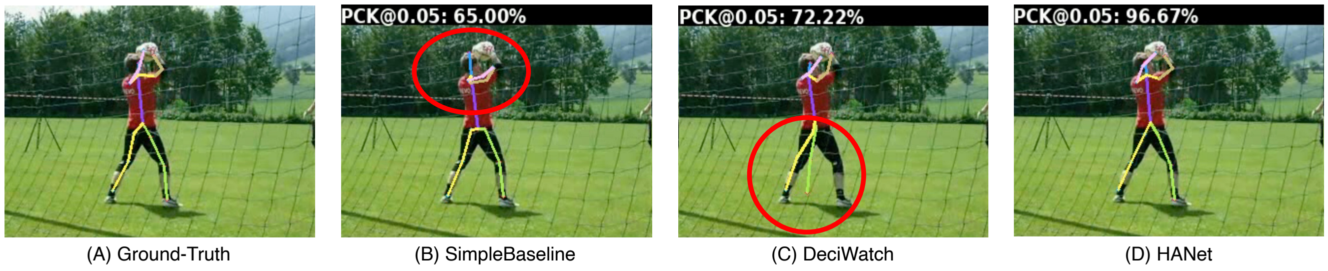


Figure 4. Qualitative comparison on Sub-JHMDB [17] dataset. From left to right, A, B, C, and D are ground truth, output of SimpleBaseline [51], DeciWatch [55], and HANet. We report PCK@0.05 on the video and visualize that our framework outperforms existing methods.

leverage the transformer decoder (**Decoder**) to estimate final poses $\mathbf{P}_s$ as,

$$\begin{aligned}\mathbf{V} &= \mathbf{Conv1D}(\mathbf{P}'\mathbf{W}_P) + \mathbf{E}_{pos,0} \\ \mathbf{P}_s &= \mathbf{Decoder}(\mathbf{V}, (\mathbf{S}_v; \mathbf{S}_a; \mathbf{X})\mathbf{W}_M)\mathbf{W}_D,\end{aligned} \quad (8)$$

where $\mathbf{V}$ is our decoder's values embedded via a 1D convolutional layer, $\mathbf{W}_P \in \mathbb{R}^{T/N \times T}$ is an interpolation matrix, and $\mathbf{W}_M \in \mathbb{R}^{(3KD+C)\times C}$ and $\mathbf{W}_D \in \mathbb{R}^{C\times(K\cdot D)}$ are linear projection matrices.

3.4. Online Mutual Learning

We further propose an online mutual learning that provides cross-supervision between refined input poses $\mathbf{P}'$ and final predictions $\mathbf{P}_s$. HANet mutually optimize them by choosing online learning targets along their training losses.

Weighted Loss. The objective of our loss function is minimizing the weighted L_1 norm between prediction and ground truth joint positions. Weighted loss tracks top-k keypoints that are hard to predict based on the training loss by extending [7]. We refer to this weighted loss as $\mathcal{L}_w$, defined as follows.

$$\mathcal{L}_w = \frac{1}{N_j}\sum_{j=1}^{N_j} v_j \|\mathbf{G}_j - \mathbf{P}_j\| + \frac{\lambda}{N_k}\sum_{k=1}^{N_k} v_k \|\mathbf{G}_k - \mathbf{P}_k\|, \quad (9)$$

where N_k, $\mathbf{G}_j$, $\mathbf{P}_j$, and v_j denote the number of top-k keypoints, ground truth, prediction, and visibility for joint j, respectively. This distance between prediction and ground truth is valid when a joint j is visible. The first loss term penalizes all keypoint errors, while the second term only tracks the top-k keypoint errors.

Online Loss. Then, from the weighted loss $\mathcal{L}_w$, the online loss $\mathcal{L}_O$ is computed by,

$$\mathcal{L}_O^j = \begin{cases} \mathcal{L}_w^j(\mathbf{P}_s, \mathbf{P}') & \mathcal{L}_w^j(\mathbf{G}, \mathbf{P}_s) < \mathcal{L}_w^j(\mathbf{G}, \mathbf{P}'), \\ \mathcal{L}_w^j(\mathbf{P}', \mathbf{P}_s) & \text{otherwise}, \end{cases} \quad (10)$$

where the first parameter of $\mathcal{L}_w$ is a target and the second one learns from the target. If $\mathcal{L}_w(\mathbf{G}, \mathbf{P}')$ is greater than $\mathcal{L}_w(\mathbf{G}, \mathbf{P}_s)$ for j^{th} keypoint, we back propagate the refined input pose $\mathbf{P}'$ to $\mathbf{P}_s$, and vice versa, final prediction $\mathbf{P}_s$ is penalized by $\mathbf{P}'$. From these two components, we define our loss function as,

$$\mathcal{L} = \mathcal{L}_w(\mathbf{G}, \mathbf{P}') + \lambda_s \mathcal{L}_{\boldsymbol{w}}(\mathbf{G}, \mathbf{P}_{\boldsymbol{s}}) + \mathcal{L}_O(\mathbf{P}', \mathbf{P}_s), \quad (11)$$

where λ_s stands for weight of the final prediction error.

4. Experiments

In this section, we discuss our extensive experiments and demonstrate that our proposed method refines input poses well and can generally be applied to 2D pose estimation, 3D pose estimation, body mesh recovery, and sparsely-annotated 2D pose estimation tasks.

4.1. Datasets

We evaluate our framework on four tasks and report experimental results on various benchmark datasets. First, we use the dataset Sub-JHMDB [17] for 2D pose estimation. Second, we validate HANet on PoseTrack2017 [16] and PoseTrack2018 [2] for sparsely-annotated multi-human 2D pose estimation. For 3D pose estimation, we select the most generally used dataset Human3.6M [15]. Lastly, we verify our model on an in-the-wild dataset 3DPW [48], and a dance dataset AIST++ [27] with fast and diverse actions, for body mesh recovery.

4.2. Estimators

We trained our model using the estimated 2D coordinates, 3D coordinates, or SMPL parameters as input. Specifically, we use off-the-shelf estimators such as SimpleBaseline [51] for Sub-JHMDB, DCPose [31] for PoseTrack, FCN [37] and Mhformer [28] for Human3.6M, PARE [21] for 3DPW, and SPIN [22] for AIST++.

4.3. Evaluation Metrics

For 2D pose estimation, we adopt the Percentage of Correct Keypoints (**PCK**) following [40, 57, 55], where the matching thresholds are set as 20%, 10%, and 5% of the bounding box size under pixel level, and mean

Method	PCK@0.2								PCK@0.1	PCK@0.05
	Head	Sho.	Elb.	Wri.	Hip	Knee	Ank.	Avg.	Avg.	Avg.
DKD [40]	98.3	96.6	90.4	87.1	99.1	96.0	92.9	94.0	-	-
KFP (ResNet18) [57]	94.7	96.3	95.2	90.2	96.4	95.5	93.2	94.5	-	-
SimpleBaseline [51]	97.5	97.8	91.1	86.0	99.6	96.8	92.6	94.4	81.3	56.9
SimpleBaseline + DeciWatch [55]	99.8	99.5	**99.7**	**99.7**	98.7	99.4	96.5	98.8	94.1	79.4
SimpleBaseline + **HANet**	**99.9**	**99.7**	**99.7**	**99.7**	**99.2**	**99.9**	**98.8**	**99.6**	**98.3**	**91.9**

Table 1. Quantitative comparison on Sub-JHMDB dataset [17] with prior works [40, 57, 51, 55].

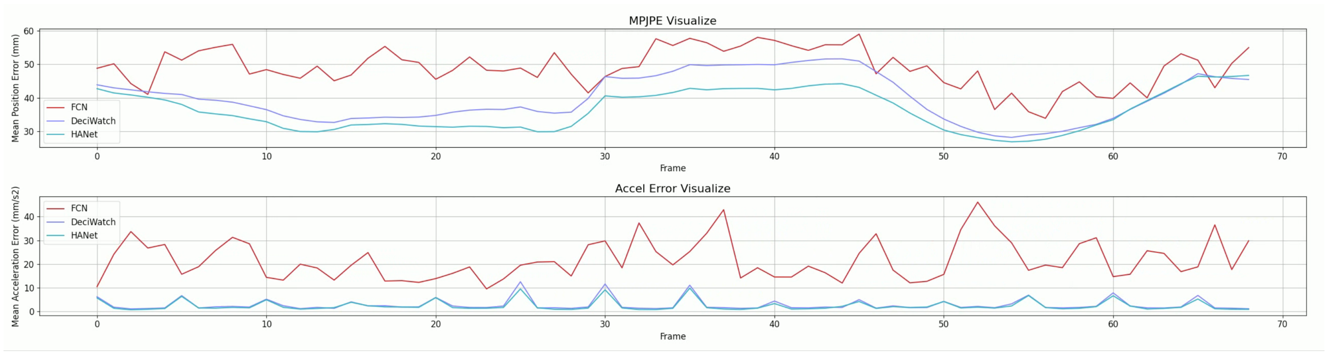


Figure 5. *MPJPE* and *Accel* error comparison with prior works [37, 55] on a video from Human3.6M [15] dataset.

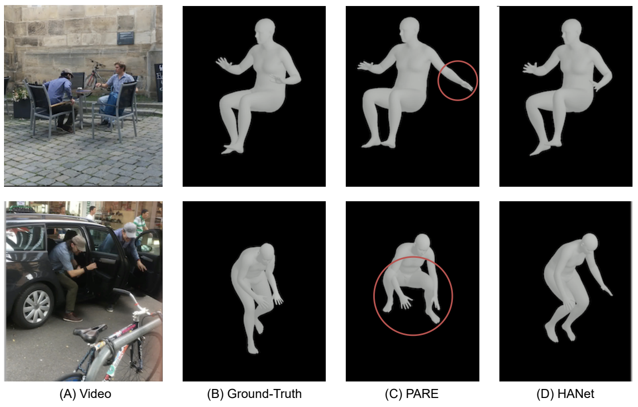


Figure 6. Qualitative results of body mesh recovery on 3DPW [48]. We visualize that HANet alleviates occlusion and jitter of input poses well. We recommend to watch our supplementary video for jitter comparison.

Average Precision (**mAP**) only for visible joints following [16, 52, 51, 44]. For 3D pose estimation and body mesh recovery, we adopt Mean Per Joint Position Error (*MPJPE*) and the mean Acceleration error (*Accel*) following [37, 22, 21].

4.4. Implementation Details

In our experiments, we consider the previous and next velocity of keypoints, $\boldsymbol{v}_p$ and $\boldsymbol{v}_n$. We set a different length T of the sliding window and an interval of frames N for each task. For sparsely-annotated multi-human 2D pose estimation, a long sliding window may not be useful because multi-human 2D pose estimation often involves more severe occlusion and motion blur than single-human pose estimation. So, we use $T = 5$ and $N = 1$ and fix the input image size to 384×288. Furthermore, PoseTrack datasets [16, 2] are sparsely annotated, we enlarge the bounding box by 25% from the annotated frame and use it to crop the supplement frames. Within a sliding window, we only trained with annotated frames. We use an AdamW optimizer [32], set an initial learning rate as 1×10^{-3}, warm it up at the first 5 epochs, and then decay in a cosine annealing manner. For weighted loss $\mathcal{L}_w$, we set λ as 0.5 to match the normalization degree of the first term because we set N_k as half of the number of joints.

4.5. Comparisons

2D Pose Estimation. As illustrated in Fig. 4, we visualize the estimated pose from [51], existing state-of-the-art method [55], and our method for Sub-JHMDB [17], which is a 2D single-human pose dataset. The input pose estimator SimpleBaseline [51] showed a performance of 65.00 on an intricate metric of PCK@0.05, and DeciWatch [55] showed a result of 72.22. Above all, we achieve the remarkable results of 96.67. In addition, we quantitatively validate our model against current state-of-the-art models [40, 57, 51, 55]. Although the performance for PCK@0.2 may look similar, Table 1 shows that the performance improvement of our model is noticeable as the threshold reduces from PCK@0.2 to PCK@0.05.

Dataset	Methods	*MPJPE* ↓	*Accel* ↓
Human3.6M [15]	FCN [37]	54.6	19.2
	FCN + DeciWatch [55] (T=101)	52.8	1.5
	FCN + **HANet** (T=51)	**51.8**	2.0
	FCN + **HANet** (T=101)	52.8	**1.4**
	Mhformer [28]	38.3	**0.8**
	Mhformer [28] + **HANet** (T=101)	**35.4**	**0.8**
3DPW [48]	PARE [21]	78.9	25.7
	PARE + DeciWatch [55] (T=101)	77.2	6.9
	PARE + **HANet** (T=51)	**74.6**	8.0
	PARE + **HANet** (T=101)	77.1	**6.8**
AIST++ [27]	SPIN [22]	107.7	33.8
	SPIN + DeciWatch [55] (T=101)	71.3	5.7
	SPIN + **HANet** (T=51)	**64.3**	6.4
	SPIN + **HANet** (T=101)	69.2	**5.4**

Table 2. Quantitative comparison on the 3D pose estimation (Human3.6M [15]) and body mesh recovery (3DPW [48], AIST++ [27]).

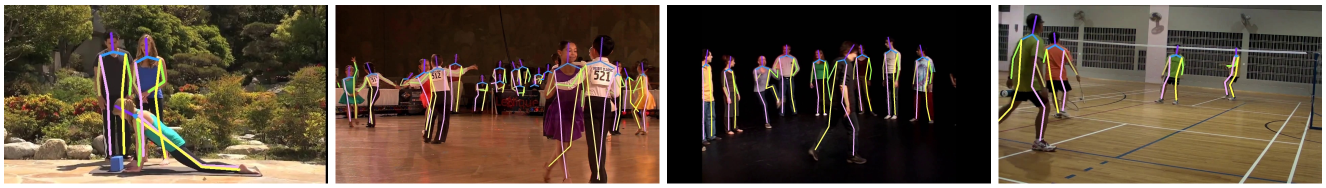


Figure 7. Qualitative results of PoseTrack 2018 test set. We demonstrate that HANet performs well at multi-human 2D pose estimation, even in occlusion, motion blur, and crowd environments.

3D Pose Estimation & Body Mesh Recovery. In addition, we demonstrate that our method can be applied to different tasks such as 3D pose estimation and body mesh recovery. We input 3d pose coordinates or SMPL parameters estimated from [21, 22, 37, 28] and refine their results. First, we validate HANet compared to state-of-the-art methods [21] for body mesh recovery on 3DPW [48] dataset. We visualize results of body mesh recovery for highly-occluded scenes. As illustrated in Fig. 6, [21] shows large positional difference. However, our model shows accurate estimation results because our model learns sufficient temporal information through keypoint kinematic features. We further visualize *MPJPE* and *Accel* error with FCN [37] and DeciWatch [55] on Human3.6M [15] dataset. In Fig. 5, we show that our framework reduces *MPJPE* more while maintaining a lower *Accel* error.

We also validate our framework and present the quantitative results on Table 2 for 3D pose estimation and body mesh recovery. We compare the *MPJPE* and *Accel* with input pose estimators [37, 28, 22, 21] and DeciWatch [55], which efficiently improved performance by sampling keyframes taking advantage of continuity of human motions. We report two version of our HANet that uses a different sliding window length $T = 51$ and $T = 101$. When $T = 101$, i.e., learn more temporal relationships between consecutive poses, we observe that *Accel* metric is improved more. Conversely, when we set $T = 51$, we demonstrate that our method reduces the need for a long sliding window and efficiently improved *MPJPE* and *Accel* errors.

Sparsely Annotated Pose Estimation. We also conduct experiments on PoseTrack2017 [16] and PoseTrack2018 [2], which is sparsely-annotated datasets. In Table 3, we compare our model with state-of-the-art methods [44, 5, 31] and the input pose estimator [31] on the validation set of PoseTrack2017 and PoseTrack2018. We achieve state-of-the-art results and demonstrate that our method improves performance (**mAP**) on sparsely-annotated multi-human 2D pose estimation. In addition, we visualize that our method shows remarkable results in occlusion, crowded scenes, fast motion, and challenging pose, as shown in Fig. 7.

4.6. Ablation Study

We validate the effect of our component and keypoint kinematic features with an extensive ablation study. We perform the experiments on Sub-JHMDB [17] and compare results in Table 4 - Table 5. We use **PCK** for the evaluation metric.

Dataset	Method	Head	Shoulder	Elbow	Wrist	Hip	Knee	Ankle	Mean
PoseTrack17 Validation [16]	HRNet [44]	82.1	83.6	80.4	73.3	75.5	75.3	68.5	77.3
	PoseWarper [5]	81.4	88.3	83.9	78.0	82.4	80.5	73.6	81.2
	DCPose [31]	88.0	88.7	84.1	78.4	83.0	81.4	74.2	82.8
	DCPose + **HANet**	**90.0**	**90.0**	**85.0**	**78.8**	**83.1**	**82.1**	**77.1**	**84.2**
PoseTrack18 Validation [2]	PoseWarper [5]	79.9	86.3	82.4	77.5	79.8	78.8	73.2	79.7
	DCPose [31]	84.0	86.6	82.7	78.0	**80.4**	79.3	73.8	80.9
	DCPose + **HANet**	**86.1**	**88.5**	**84.1**	**78.7**	79.0	**80.3**	**77.4**	**82.3**

Table 3. Quantitative comparison on the validation sets of PoseTrack2017 [16] and PoseTrack2018 [2].

Component				**PCK@0.2**	**PCK@0.1**	**PCK@0.05**
N_L	HE	OML L_1	OML L_2			
4				98.2	95.9	83.8
4	✓			99.2	97.0	86.1
4	✓		✓	99.1	96.3	85.8
4	✓	✓		99.3	97.2	86.7
4	✓	✓	✓	99.4	97.3	87.2
5	✓	✓	✓	**99.6**	**98.3**	**91.9**
6	✓	✓	✓	99.5	97.7	90.2

Table 4. Ablation study to observe the contribution of each component of our model on JHMDB [17].

Hierarchical Encoder. First, we compare and report the effect of our hierarchical encoder at rows 1-2 and scale of the encoder at rows 5-7 in Table 4. When we replace the hierarchical encoder to normal transformer encoder, performance significantly drops (2.3%) for PCK@0.05. In addition, we scale the number of encoder and decoder layers N_L to 4 and grew one by one. It gradually improves performance up to 5 layers (4.7%) and decreases (1.7%).

Online Mutual Learning. We verify the effectiveness of our proposed online mutual learning (OML) at rows 2-5. Rows 2 and 4 indicates that OML with L_1 norm has a significant impact on performance improvement (0.6%) for PCK@0.05. We also conduct experiments for OML with L_2 norm (row 3 and 5) and found that OML has improved performance by jointly using the L_1 norm and L_2 norm. We proceed with the rest of the ablation study using the row 6 version that showed the best performance.

Keypoint Kinematic Features. We also conduct ablation studies on keypoint kinematic features on Table 5. Here, except for the last row, the encoder does not take concatenated previous frame and subsequent frame features. The first row is the minimum version of HANet that removes all features associated with keypoint kinematic features, and the last row is the complete version of HANet considering all keypoint kinematic features: flows, velocity,

Component				**PCK@0.2**	**PCK@0.1**	**PCK@0.05**
Flow	Vel.	Accel.	WB			
				99.1	96.3	86.5
✓				99.4	97.4	88.4
✓	✓			99.4	97.6	90.6
✓	✓	✓		99.5	98.1	91.5
✓	✓	✓	✓	**99.6**	**98.3**	**91.9**

Table 5. Ablation study of keypoint kinematic features on JHMDB. WB stands for weight and bias of velocity and acceleration.

acceleration, and WB (weight and bias). From the top, the rows in turn indicate the addition of flow, velocity, acceleration, and WB. We could observe significant performance impact of each keypoint kinematic features in the order of velocity (2.2%), flow (1.9%), acceleration (0.9%), and WB (0.4%) for PCK@0.05.

5. Conclusion

In this work, we construct a hierarchical transformer encoder and exploit the keypoint kinematic features to address occlusion and mitigate jitter. We demonstrate that explicitly leveraging flow, velocity, and acceleration can capture the keypoint's temporal dependencies better. Furthermore, HANet effectively addresses the occlusion issue by learning spatio-temporal relationships between consecutive frames' poses through multi-scale feature maps and achieve state-of-the-art performance on a variety of pose estimation tasks. We are interested in extending our model to an end-to-end model, which can be combined with our approach to produce more accurate results.

Acknowledgement. This work was partially supported by Institute of Information & communications Technology Planning & Evaluation (IITP) grant funded by the Korea government(MSIT) (No. 2019-0-00079, Artificial Intelligence Graduate School Program (Korea University), No. 2022-0-00984, Development of Artificial Intelligence Technology for Personalized Plug-and-Play Explanation and Verification of Explanation).

References

[1] Mohiuddin Ahmad and Seong-Whan Lee. Human action recognition using multi-view image sequences. In *FG*, 2006.

[2] Mykhaylo Andriluka, Umar Iqbal, Eldar Insafutdinov, Leonid Pishchulin, Anton Milan, Juergen Gall, and Bernt Schiele. Posetrack: A benchmark for human pose estimation and tracking. In *CVPR*, 2018.

[3] Anurag Arnab, Mostafa Dehghani, Georg Heigold, Chen Sun, Mario Lučić, and Cordelia Schmid. Vivit: A video vision transformer. In *ICCV*, 2021.

[4] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

[5] Gedas Bertasius, Christoph Feichtenhofer, Du Tran, Jianbo Shi, and Lorenzo Torresani. Learning temporal pose estimation from sparsely-labeled videos. In *NeurIPS*, 2019.

[6] Zhe Cao, Tomas Simon, Shih-En Wei, and Yaser Sheikh. Realtime multi-person 2d pose estimation using part affinity fields. In *CVPR*, 2017.

[7] Yilun Chen, Zhicheng Wang, Yuxiang Peng, Zhiqiang Zhang, Gang Yu, and Jian Sun. Cascaded pyramid network for multi-person pose estimation. In *CVPR*, 2018.

[8] Chhavi Dhiman and Dinesh Kumar Vishwakarma. View-invariant deep architecture for human action recognition using two-stream motion and shape temporal dynamics. *TIP*, 2020.

[9] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. In *ICLR*, 2020.

[10] Hao-Shu Fang, Shuqin Xie, Yu-Wing Tai, and Cewu Lu. Rmpe: Regional multi-person pose estimation. In *ICCV*, 2017.

[11] Hiromichi Fujisawa, Hiroshi Sako, Yoshihiro Okada, and Seong-Whan Lee. Information capturing camera and developmental issues. In *ICDAR*, 1999.

[12] Hengkai Guo, Tang Tang, Guozhong Luo, Riwei Chen, Yongchen Lu, and Linfu Wen. Multi-domain pose network for multi-person pose estimation and tracking. In *ECCVW*, 2018.

[13] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016.

[14] Yihui He, Rui Yan, Katerina Fragkiadaki, and Shoou-I Yu. Epipolar transformers. In *CVPR*, 2020.

[15] Catalin Ionescu, Dragos Papava, Vlad Olaru, and Cristian Sminchisescu. Human3. 6m: Large scale datasets and predictive methods for 3d human sensing in natural environments. *TPAMI*, 2013.

[16] Umar Iqbal, Anton Milan, and Juergen Gall. Posetrack: Joint multi-person pose estimation and tracking. In *CVPR*, 2017.

[17] Hueihan Jhuang, Juergen Gall, Silvia Zuffi, Cordelia Schmid, and Michael J Black. Towards understanding action recognition. In *ICCV*, 2013.

[18] Tao Jiang, Necati Cihan Camgoz, and Richard Bowden. Skeletor: Skeletal transformers for robust body-pose estimation. In *CVPR*, 2021.

[19] Kyung-Min Jin, Gun-Hee Lee, and Seong-Whan Lee. OTPose: Occlusion-Aware Transformer for Pose Estimation in Sparsely-Labeled Videos. In *SMC*, 2022.

[20] Tae-Kyung Kang, Gun-Hee Lee, and Seong-Whan Lee. HTNet: Anchor-free Temporal Action Localization with Hierarchical Transformers. In *SMC*, 2022.

[21] Muhammed Kocabas, Chun-Hao P Huang, Otmar Hilliges, and Michael J Black. PARE: Part attention regressor for 3D human body estimation. In *ICCV*, 2021.

[22] 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*, 2019.

[23] Gun-Hee Lee and Seong-Whan Lee. Uncertainty-aware human mesh recovery from video by learning part-based 3d dynamics. In *ICCV*, 2021.

[24] Seong-Whan Lee, Jin H Kim, and Frans CA Groen. Translation-, rotation-and scale-invariant recognition of hand-drawn symbols in schematic diagrams. *IJPRAI*, 1990.

[25] Seong-Whan Lee and Sang-Yup Kim. Integrated segmentation and recognition of handwritten numerals with cascade neural network. *SMC, Part C*, 1999.

[26] Seong-Whan Lee and Alessandro Verri. *Pattern Recognition with Support Vector Machines: Proc. of First International Workshop, Niagara Falls, Canada*. Springer, 2003.

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

[28] Wenhao Li et al. Mhformer: Multi-hypothesis transformer for 3d human pose estimation. In *CVPR*, 2022.

[29] Yanjie Li, Shoukui Zhang, Zhicheng Wang, Sen Yang, Wankou Yang, Shu-Tao Xia, and Erjin Zhou. Tokenpose: Learning keypoint tokens for human pose estimation. In *ICCV*, 2021.

[30] 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 *ECCV*, 2014.

[31] Zhenguang Liu, Haoming Chen, Runyang Feng, Shuang Wu, Shouling Ji, Bailin Yang, and Xun Wang. Deep dual consecutive network for human pose estimation. In *CVPR*, 2021.

[32] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *ICLR*, 2019.

[33] Yue Luo, Jimmy Ren, Zhouxia Wang, Wenxiu Sun, Jinshan Pan, Jianbo Liu, Jiahao Pang, and Liang Lin. Lstm pose machines. In *CVPR*, 2018.

[34] 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. In *BMVC*, 2021.

[35] Andrew L Maas, Awni Y Hannun, Andrew Y Ng, et al. Rectifier nonlinearities improve neural network acoustic models. In *ICML*, 2013.

[36] Weian Mao, Zhi Tian, Xinlong Wang, and Chunhua Shen. FCPose: Fully Convolutional Multi-Person Pose Estima-

tion with Dynamic Instance-Aware Convolutions. In *CVPR*, 2021.

[37] Julieta Martinez, Rayat Hossain, Javier Romero, and James J Little. A simple yet effective baseline for 3d human pose estimation. In *ICCV*, 2017.

[38] Alejandro Newell, Kaiyu Yang, and Jia Deng. Stacked hourglass networks for human pose estimation. In *ECCV*, 2016.

[39] Aiden Nibali, Zhen He, Stuart Morgan, and Luke Prendergast. 3d human pose estimation with 2d marginal heatmaps. In *WACV*, 2019.

[40] Xuecheng Nie, Yuncheng Li, Linjie Luo, Ning Zhang, and Jiashi Feng. Dynamic kernel distillation for efficient pose estimation in videos. In *ICCV*, 2019.

[41] Leonid Pishchulin, Eldar Insafutdinov, Siyu Tang, Bjoern Andres, Mykhaylo Andriluka, Peter V Gehler, and Bernt Schiele. Deepcut: Joint subset partition and labeling for multi person pose estimation. In *CVPR*, 2016.

[42] Myung-Cheol Roh, Tae-Yong Kim, Jihun Park, and Seong-Whan Lee. Accurate object contour tracking based on boundary edge selection. *Pattern Recognition*, 2007.

[43] Myung-Cheol Roh, Ho-Keun Shin, and Seong-Whan Lee. View-independent human action recognition with volume motion template on single stereo camera. *Pattern Recognition Letters*, 2010.

[44] Ke Sun, Bin Xiao, Dong Liu, and Jingdong Wang. Deep high-resolution representation learning for human pose estimation. In *CVPR*, 2019.

[45] Amor Ben Tanfous, Aimen Zerroug, Drew Linsley, and Thomas Serre. How and What to Learn: Taxonomizing Self-Supervised Learning for 3D Action Recognition. In *WACV*, 2022.

[46] Alexander Toshev and Christian Szegedy. Deeppose: Human pose estimation via deep neural networks. In *CVPR*, 2014.

[47] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *NeurIPS*, 2017.

[48] 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*, 2018.

[49] Manchen Wang, Joseph Tighe, and Davide Modolo. Combining detection and tracking for human pose estimation in videos. In *CVPR*, 2020.

[50] Shih-En Wei, Varun Ramakrishna, Takeo Kanade, and Yaser Sheikh. Convolutional pose machines. In *CVPR*, 2016.

[51] Bin Xiao, Haiping Wu, and Yichen Wei. Simple baselines for human pose estimation and tracking. In *ECCV*, 2018.

[52] Yuliang Xiu, Jiefeng Li, Haoyu Wang, Yinghong Fang, and Cewu Lu. Pose Flow: Efficient online pose tracking. In *BMVC*, 2018.

[53] Hee-Deok Yang and Seong-Whan Lee. Reconstruction of 3D human body pose from stereo image sequences based on top-down learning. *Pattern Recognition*, 2007.

[54] Sen Yang, Zhibin Quan, Mu Nie, and Wankou Yang. Transpose: Keypoint localization via transformer. In *ICCV*, 2021.

[55] Ailing Zeng, Xuan Ju, Lei Yang, Ruiyuan Gao, Xizhou Zhu, Bo Dai, and Qiang Xu. Deciwatch: A simple baseline for 10x efficient 2d and 3d pose estimation. In *ECCV*, 2022.

[56] Jiabin Zhang, Zheng Zhu, Wei Zou, Peng Li, Yanwei Li, Hu Su, and Guan Huang. Fastpose: Towards real-time pose estimation and tracking via scale-normalized multi-task networks. *arXiv preprint arXiv:1908.05593*, 2019.

[57] Yuexi Zhang, Yin Wang, Octavia Camps, and Mario Sznaier. Key frame proposal network for efficient pose estimation in videos. In *ECCV*, 2020.

[58] Ce Zheng, Sijie Zhu, Matias Mendieta, Taojiannan Yang, Chen Chen, and Zhengming Ding. 3d human pose estimation with spatial and temporal transformers. In *ICCV*, 2021.