Hierarchical Dynamic Programming Module for Human Pose Refinement

Chunyang Xie, *Student Member, IEEE,* Dongheng Zhang, *Student Member, IEEE,* Yang Hu and Yan Chen, *Senior Member, IEEE*

***Abstract*—We observed that remarkable and impressive performance on image-based human pose estimation have been achieved by deep Convolutional Neural Networks (CNN). Nevertheless, directly applying these image-based models on videos is not only computionally intensive, but also may cause jitter and loss. The main reason is that the image-based models purely focus on the local features of individual frames and totally ignore the temporal information among adjacent frames. Some existing methods are proposed to address the temporal coherency issue. However, these methods need to be designed carefully and cannot be combined with existing image-based methods. In this paper, we propose a simple yet effective module to refine the estimated pose by exploting the temporal coherency among the heatmaps of adjacent frames, which can be easily inserted into image-based networks as a plug-in. We show that the temporal coherency issue among the heat map frames could be re-formulated as a graph path selection optimization problem. Moreover, to speed up the refinement process, we propose a hierarchical graph optimization to achieve the refinement from coarse to fine. Experimental results on two large-scale video pose estimation benchmarks show that that the proposed module can process up to 120 frames per second even when running on a single CPU thread, with better performance than all baseline methods.**

***Index Terms*—Human pose refinement, dynamic programming , pose estimation, deep learning**

I. Introduction

HUMAN pose estimation (HPE), a fundamental but important task, has been studied extensively in the literature of computer vision. HPE obtains the configuration of human skeleton from input visual data, which provides geometric and motion information to facilitate many high-level computer vision tasks, such as action recognition [1], [2], person re-identification [3], [4], etc. Therefore, HPE is a crucial task with extremely high application value.

With the development of deep Convolutional Neural Networks (CNN) in recent years, remarkable and impressive performance have been achieved by existing CNN-based mehods in HPE tasks. For example, Wei et al. [5] proposed the Convolutional Pose Machine (CPM) to refine joint locations via a sequence of predictors in the network. Around the same time, Newell et al. [6] proposed a structure named ”Stacked Hourglass Networks”, which realized accurate estimation of human joints by cascades of the hourglass structure. OpenPose [7] predicted joint locations based on [5] and then assembled joints via part affinity fields (PAFs). More recent methods [8]–[10] obtained human bounding boxes and then warped the high resolution feature maps with the boxes to predict human pose.

Chunyang Xie is with the School of Information and Communication Engineering, University of Electronic Science and Technology of China, Chengdu, China. Email: chunyangxie@std.uestc.edu.cn

Dongheng Zhang is with the School of Cyberspace Security, University of Science and Technology of China, Hefei, Anhui, China. Email: dongheng@ustc.edu.cn

Yang Hu is with the School of Information Science and Technology, University of Science and Technology of China, Hefei, Anhui, China. Email: eeyhu@ustc.edu.cn

Yan Chen is with the School of Cyberspace Security, University of Science and Technology of China, Hefei, Anhui, China. Email: eecyan@ustc.edu.cn

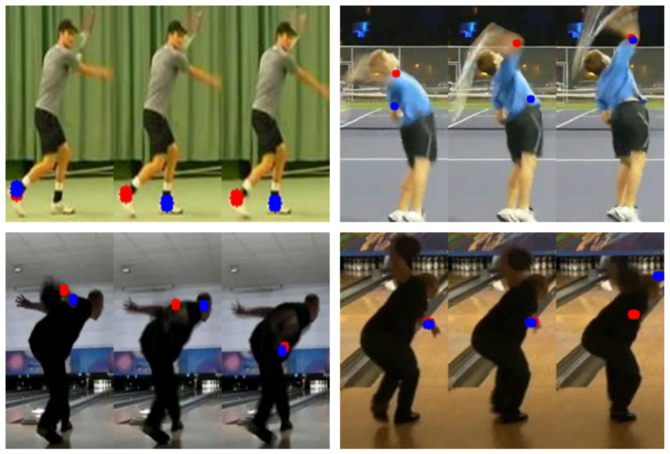

Fig. 1. Typical errors occur when applying the image-based pose estimation methods on videos under various scenarios, where the red points represent ground-truth annotations and the blue points are the results of OpenPose [7]: a) estimating symmetric joints like right and left elbows; b) human body moves rapidly; c) poor lighting conditions; d) body parts are occluded.

Nevertheless, the aforementioned image-based methods studied the HPE task on the still images, which cannot be directly applied on the videos. As shown in Fig.1, when applying the OpenPose [7] on the video sequences, various estimation errors may occur, e.g., the symmetric joints can be wrongly estimated, and fast motion, poor lighting condition, as well as occlusion can also lead to incorrect estimation of joint locations. The main reason is that the image-based methods estimate the joints purely based on the local appearance features of individual frames but totally ignore the temporal information among adjacent frames, which thus leads to degraded performance on video sequences.

To address the temporal coherency issue, many methods have been proposed. For instance, Ramakrishna et al. [11] and Zhang et al. [12] processed each joint separately and then got the initial hypotheses. Afterwards they obtained best pose hypotheses among adjacent video frames by utilizing graph optimization methods. However, these methods needs to be carried out on the whole video, which is computationally intensive. Ruder et al. [13] and Chen et al. [14] took optical flow from the video into account and propagated short-term coherence to long-term ones. Nevertheless, these methods need to be carefully designed and cannot be combined with existing image-based methods, which limits their applications.

We observe that the heatmaps output by the image-based pose estimation methods contain abundant information including the possible positions of joints and the corresponding confidence scores. Thus, in this paper, we propose a simple yet effective module to refine the estimated pose by exploiting the temporal coherency among the heatmaps of adjacent frames. Specifically, we formulate the refinement process as a graph path selection problem and utilize the graph optimization method to obtain the optimal pose hypotheses. Moreover, to reduce the computational complexity and make the proposed method more efficient, we propose a hierarchical graph optimization to refine the pose locations from coarse to fine. The main contributions of this paper can be summarized as follows.

1. We propose a simple yet effective module to refine the pose estimation results by utilizing the temporal consistency and dependency among the heatmaps of adjacent frames. The refinement process is formulated as a graph path selection optimization problem. It is worth pointing out that the proposed module does not need any training and thus can be combined with any existing image-based pose estimation methods.

2. To speed up the refinement process, we propose a hierarchical graph optimization method to achieve the refinement from coarse to fine. Experimental results on two large-scale public video pose estimation datasets show that the proposed module can process up to 120 frames per second even when running on a single CPU thread, with better performance than all baseline methods.

The rest of this paper is organized as follows. Section II introduces the related work of human pose estimation. Section III presents our module and optimization algorithm. Experiment settings and results are shown in section IV. Finally, conclusions are drawn in section V.

II. Related Works

HPE methods aim to estimate spatial human joint locations from images or videos. Traditional HPE methods utilize hand-crafted features to obtain global pose structures. Recently, with the development of deep learning, CNN-based solutions have improved the estimation performance significantly. In the following, we will review the existing HPE methods in images and videos, respectively.

A. Human Pose Estimation in Images

Early HPE methods [15]–[18] built the graphical structures to model geometry relations between human joints. However, those works rely heavily on hand-crafted features, which limit their generalization to real-world applications. With the development of deep learning, the performance of CNN based models [7], [9], [10], [19] have been impressively improved. Lifshitz et al. [20] proposed a CNN-based HPE network which determined locations of joints by incorporating the keypoints votes and joint probabilities. The Convolutional Pose Machine (CPM) [5] refined joint locations using a sequence of predictors in the network. OpenPose [7] predicted joint locations based on the CPM [5] and then assembled joints via part affinity fields (PAFs). Newell et al. [6] proposed an encoder-decoder structure named ”stacked hourglass networks”, which realized accurate estimation of human joints by cascades of the hourglass structure. The stacked hourglass (SHG) network repeated pooling and upsampling layers to capture multi-scale information of human pose. Since then, many variant structures of SHG were developed. For example, Yang et al. [21] proposed a Pyramid Residual Module (PRMs) to replace the residual unit in SHG, which can learn convolutional filters on various scales of input features and enhance the invariance in scales of deep convolutional neural networks. Chu et al. [22] designed a variation of SHG named Hourglass Residual Units (HRUs), which extended residual units with a side branch incorporating filters with larger receptive field, to learn and combine different scale features.

In addition to SHG and its variants, many other structures have also been designed. Mask R-CNN [8] is a well designed framework for object detection, instance segmentation and keypoint detection. It generates RoIs via a Region Proposal Network (RPN) first. Then ROIAlign layer is proposed to extract feature map for each ROI. Unlike RoIPool [23], RoIAlign avoids harsh quantization of the RoI boundaries and brings large improvements in many tasks. Finally, different head architectures are applied for different tasks. For pose estimation, the keypoint head architecture consists of a stack of conv layers, followed by a deconv layer and bilinear upscaling, producing heatmaps for each joint. The Cascaded Pyramid Network (CPN) [24] designed two novel networks to refine pose locations. GloabalNet is designed to locate ”simple” joint locations which is a pyramid feature extractor. The pyramid feature representation contains inevitabel information to inference ”hard” keypoints, such as occluded and invisible joints. And RefineNet handles the ”hard” keypoints by combining all levels of feature from GlobalNet. An online hard keypoint mining loss is applied. CPN follows a top-down pipeline, which first predicts the human bounding boxes and then estimates keypoint locations for each bounding box.

Besides these efforts in effective network design, some works explored the importance of high-resolution feature representation to improve the accuracy of HPE methods. Xiao et al. [9] proposed a simple network for human pose estimation and pose tracking, where ResNet [25] is applied as the backbone. Only a few deconvolutional layers are added to the last convolution stage in ResNet backbone to produce heatmaps for high-resolution representations, which achieves a balance between efficiency and accuracy. The High-Resolution Network (HRNet) [10], [26] consists of parallel high-low resolution sub-networks with multi-scale feature fusion, which can maintain high-resolution representations through the whole process. It starts from a high-resolution sub-network. Then several high-to-low resolution sub-networks are added one by one. The multi-resolution sub-networks are connected in parallel so that the resolutions for the sub-networks of a later stage consists of ones from the previous stage and an extra lower one. Besides, exchange units are proposed to fuse information

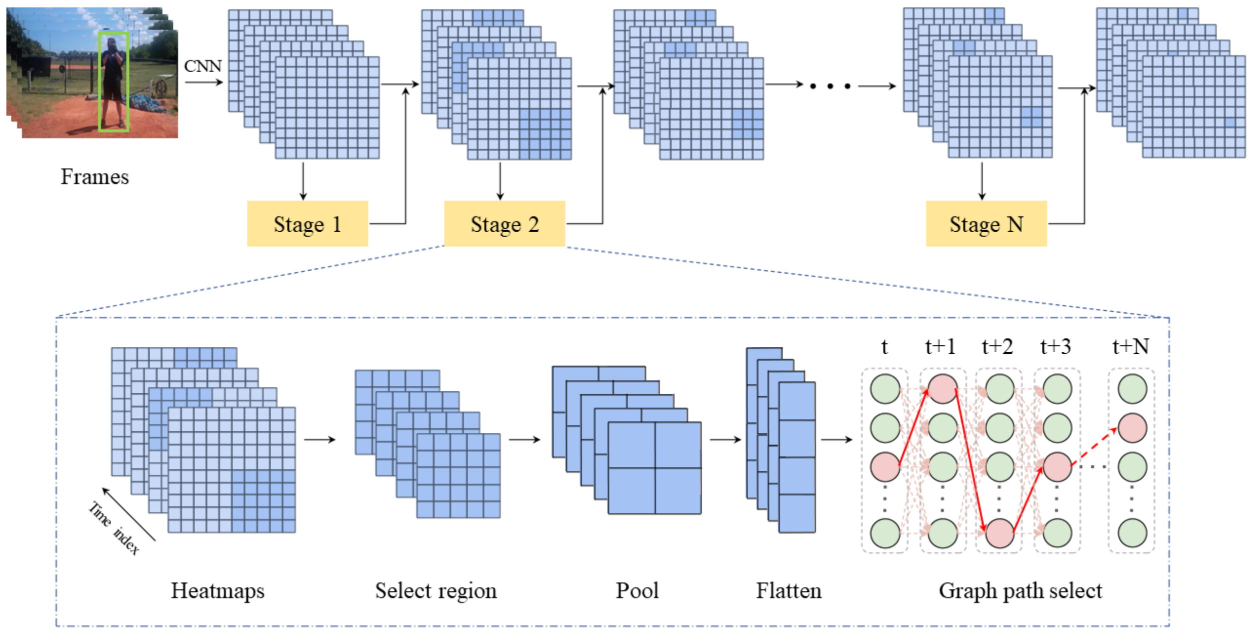

Fig. 2. The proposed hierarchical dynamic programming module for human pose refinement. Image-based human pose estimation network extracts features from video sequences to generate heatmaps, which corresponding to different key points of the human body. Our model refines these heat map sequences in a hierarchical and recursive manner. At each stage, a specific region of original heat map (the dark blue square in the figure) is first picked based on the results of the previous stage and pooled to a specific size, and then flattened into a one-dimensional vector. The optimal path is obtained by graph path selection algorithm, which will lead to a finer region or pixel of original heat map for the next stage.

across parallel sub-networks. The final joint predictions are obtained from the heatmaps output by the last exchange unit. Inspired by HRNet [10], [26], Cheng et al. [27] proposed a bottom-up pose estimation method for learning scale-aware representations, which is named HigherHRNet. Scale variation is the main challenge for bottom-up methods, especially when predicting the pose of small human bodies. HigherHRNet extracts high-resolution heatmaps via a new high-resolution feature pyramid module based on the 1/4 resolution path of HRNet [10], [26] . The proposed feature pyramid starts directly from $1/4$ resolution which is the highest resolution feature in the backbone. Then transposed convolution layers are applied to generate even higher-resolution feature maps. Besides, they propose a Multi-Resolution Supervision strategy to assign training target of different resolutions to the corresponding feature pyramid level. Finally, to generate scale-aware high-resolution heatmaps, a simple Multi-Resolution Heat map Aggregation strategy is introduced.

Although these CNN-based methods have achieved impressive results on image datasets, applying them directly to videos often produce unsatisfactory errors, such as jitter and loss. The reason can be attributed to the fact that these methods process the video frame by frame, ignoring the temporal coherency contained among adjacent frames.

B. Human Pose Estimation in Videos

Pose estimation in videos brings new challenge that how to utilize the abundant temporal information among adjacent frames to make joint predictions more robust and stable. A few previous studies [11], [12], [28], [29] tried to integrate temporal cues into human pose estimation. Ramakrishna et al. [11] and Zhang et al. [12] processed each joint separately and obtained the initial hypotheses. Afterwards the best pose hypotheses between adjacent video frames are obtained by optimization methods. These graph optimization methods need to be carried out on the whole video, which is computationally intensive. Jain et al. [29] proposed a two-branch CNN network named ”Modeep” to extract both motion and color features within frame pairs to model the spatio-temporal information in HPE. Pfister et al. [28] inserted frames at different color channels as inputs to merge the motion in the video. Afterwards, Pfister et al. [30] and Song et al. [31] both utilized dense optical flow [32] to adjust the joint locations to make the motion more stable. Thin-Slicing Network [31] obtained impressive results based on both adjustment from optical flow and a spatial-temporal model. Nevertheless, the intensive computational complexity makes the model slower than many image-based methods.

Some effective network designs have also been applied to video pose estimation tasks. Chained model [33] is a simple recurrent architecture to incorporate temporal component. LSTM Pose Machine [34] utilized LSTM to capture temporal dependency. RPSM [35] used LSTM to obtain better correspondence during the regression procedure from 2D to 3D. The winner [36] of ICCV’s 17 PoseTrack Challenge [15] designed a two-stage strategy to track multi-person poses: first estimated human pose via Mask R-CNN [8], then tracked the pose online using a greedy bipartite matching algorithm frame by frame. However, they only use temporal coherency to assign person IDs, not to refine human pose results. Pose Flow [37] proposed a novel framework, which mainly leverages spatio-temporal information to boost pose tracking task. Pose Flow Builder (PF-Bulider) and Pose Flow NMS (PF-NMs) are proposed to associate the cross-frame poses and improve the NMS stabilization.

Liu et al. [38] proposed a new method to maintain temporal consistency for video pose estimation via the structured space learning and halfway temporal evaluation methods. Zecha et al. [39] proposed a graph partitioning problem that connects pose over time and solve the problem using integer linear programming (ILP). Nie et al. [40] proposed a Dynamic Kernel Distillation (DKD) model for improving efficiency of video pose estimation. In particular, DKD used a distillator to online distill pose kernels via temporal cues from the previous frame. Afterwards, the joint localization problem is simplified into a matching procedure between pose kernels and the current frame, which can be solved by simple convolution. Artacho et al. [41] proposed a unified framework named ”UniPose” for human pose estimation, which is based on the Waterfall Atrous Spatial Pooling (WASP) module [42].

While achieving promising results, the aforementioned methods still have some drawbacks. Firstly, the computational complexity is typically high due to the optimization over the whole video sequences. Secondly, almost all methods need to carefully design and refine over new data. Different from these methods, our method is computationally efficient without any training, and can be combined with existing image-based methods.

III. The Proposed Hierarchical Dynamic Programming Module for Pose Refinement

As a plug-in, our module can be easily inserted into the existing human pose estimation network. The overall pipeline is shown in Fig. 2. The backbone image-based network extracts features of video frames to output heatmaps of various joints. Taking heatmaps as input, our module refines human pose in a recursive manner, i.e., we refine the initial joint locations from coarse to fine. At each stage, we select a specific region of the heatmaps based on the results of the previous stage, pool and flatten it into vectors. Then we apply a graph path selection algorithm to get the shortest path and output the final estimated pose.

A. Problem Formulation

Without considering the temporal consistency, the pridicted pose will be jittered or lost. With the help of the information provided by adjacent frames, it is expected that the pose prediction errors of current frame can be mitigated or corrected, resulting in smoother motion trajectory of the pose. Considering such a motivation, we take the information of multiple video frames into consideration and view the pose refinement process as an optimization problem. Since the heatmaps output by the image-based pose estimation methods contain rich spatial location information, we refine pose predictions based on the heat map sequence, which are represented as

$$\mathbf{H_P} = (H_P^0, H_P^1, H_P^2, \dots H_P^T), \quad P \in \{0, 1, \dots, N\} \tag{1}$$

where H_P is the sequence for the P_{th} joint of all N points and T is sequence length. The size of each heat map H_P^T is (h^T, w^T). Note that the size of each frame's heat map may be different.

Due to noise and lack of temporal consistency, the locations of predicted keypoints by image-based models can be unstable. If we expand and arrange all the candidate keypoints in the heat map sequence, then the task of refining keypoints can be regarded as the shortest path problem of a lattice, where each column of the lattice map is the candidate points of a heat map frame. We can extract the optimal path of the P_{th} joint by minimizing the following total cost function

$$\mathbf{Y_P} = \underset{\boldsymbol{y_P}}{\arg\min}\, C = \sum_{t=1}^{T} C_P^t \tag{2}$$

where $\mathbf{Y_P}$ represents the set of optimal keypoint coordinates, and C_P^t is the cost matrix at time t.

Considering that the heat map sequence contains the confidence and the spatial location information of keypoints, we can use them to construct the constraint matrix in Eqn. (2). Therefore, we proposed a constraint function based on confidence and distance items as follow:

$$\begin{aligned}\mathbf{C_P}(Y_P^t \to Y_P^{t+1}) =& \lambda \cdot dist(Y_P^t, Y_P^{t+1}) + \\ & (1-\lambda) \cdot exp(-S_{Y_P^{t+1}})\end{aligned} \tag{3}$$

where Y_P^t is the coordinate point picked from the heat map of the P_{th} joint at time t, $dist$ is the euclidean distance function, and $S_{Y_P^{t+1}}$ is the confidence score of the coordinate point at time $t+1$. The $\lambda \in [0, 1]$ is the weight parameter to balance the two items.

From Eqn. (2) and Eqn. (3), the optimization problem can be efficiently solved by dynamic programming with value iteration algorithm. The state transition equation can be obtained :

$$D(t+1) = \min(D(t) + C_P(Y_P^t, Y_P^{t+1})) \tag{4}$$

where $D(t+1)$ denotes the distance between the first and the $(t+1)_{th}$ frame.

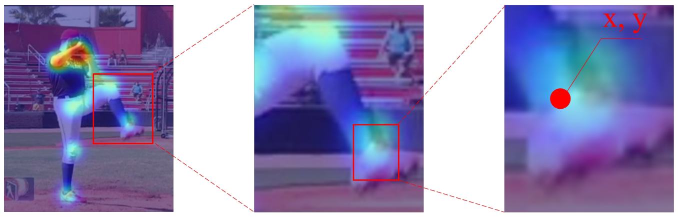

Fig. 3. The mechanism by which the human visual system identifies body parts. The red boxes represent the locations noticed by the human eye at each stage. Through several iterations, the human eye eventually focuses its attention on the most likely area and obtains the coordinate position (the red dot).

B. Hierarchical Dynamic Programming

In the above discussions, we have formulated the pose refinement process using temporal consistency as a graph path selection problem. Considering the large dimensions of the heatmaps, directly applying the dynamic programming would lead to large computational complexity. Assume that the length of the lattice graph in Fig. 2 is N and the width is D. The dynamic programming algorithm compares $D * D$ times between every two layers, keeps the optimal path to the next layer, and recursively, so the algorithm complexity is $O(N * D^2)$, which is relatively high. In this subsection, we will introduce hierarchical dynamic programming (HDP), an optimization method that can effectively reduce the complexity of the original algorithm while maintaining accuracy.

The proposed hierarchical dynamic programming is motivated by the human visual system, which scans the entire scene element, finds and focuses on the area of interest, and looks closely to obtain external information, as shown in Fig. 3. Following the principle of finding the region of interest before determining the target location, we improve the original dynamic programming algorithm into a hierarchical iterative form. As shown in Fig. 2, an image-based CNN backbone network extracts features of video frames to output heatmaps of various joints. Suppose the size of original heat map H_{ori} is (W, H). At each stage, we pool a specific region of H_{ori} based on the results of the previous stage to obtain heat map H_{pool} with the size of (M, N). Note the whole heat map H_{ori} will be picked when in the first stage. Afterwards, we flatten H_{pool} into vector V with the length of $M \times N$, where the index of each element maps a region of H_{ori}. Therefore, we would obtain the optimal region (or pixel) of the original heat map H_{ori} if the optimal path (i.e. the optimal element index of vector V) is obtained by graph path selection algorithm, which will lead to a finer region or pixel of original heat map for the next stage.

A key question is how we calculate the cost matrix of the pooled heat map sequence at each stage. Only after obtaining the exact cost matrix, we can use Eqn. (4) to construct the state transfer matrix and obtain the optimal path at the current stage by the dynamic programming algorithm. According to Eqn. (3), the cost matrix consists of two main items: the distance term $dist(Y_P^t, Y_P^{t+1})$ and the confidence term $exp(-S_{Y_P^{t+1}})$. Considering that each pooled heat map in the sequence may come from a different region of the original image, we cannot directly calculate the Euclidean distance between the locations in adjacent maps. Therefore, in each iteration we use affine transformation to map each position back to the original image coordinate system and then calculate the distance matrix. This way, no matter which region on the heat map is selected, we get the distance matrix in a uniform coordinate system. As for the confidence term, at each stage we use the value in the heat map after maximum pooling as the confidence score for the current position. We choose maximum pooling because it is better at retaining the regions with larger values in the heatmap, i.e. the regions where the backbone network is more confident. Average pooling, on the other hand, dilutes the influence of the larger value regions as the pooling kernel increases, which can have a detrimental effect on our module.

At each stage we simply use the downsampled heat map to accelerate the search for the optimal path, and then select the optimal region or pixel on the original heat map based on this path, so that we can gradually focus our attention on the most likely region instead of the global one, and finally get the optimal motion trajectory of the key points.

Furthermore, we can divide the whole video sequence into chunks and perform HDP method on the video chunks since a large motion generally does not last throughout the whole video. Obviously, a large length of video chuck would lead to large computational complexity. In the experiment, we divide the video into video chucks with length of 20, which achieves a good tradeoff between accuracy and efficiency.

Suppose we need to process a video sequence of length N. It is known that each heat map generated by the backbone network has D pixels after being flattened. If the length of the video chuck is $N/2$ and a two-stage hierarchical dynamic programming is adopted, then the overall time complexity is $O(N/2 * D)$, which is much smaller than the original time complexity $O(N * D^2)$. In the experiment, we use a heat map with a size of 64×48, i.e. the most common size for current human pose estimation networks. Suppose video length is 40 frames. If we use a video chuck with a length of 20 frames, theoretically the time complexity of our module will be about 3000 times smaller than the original algorithm.

C. Parameter estimation

According to Eqn. (3), λ is an important parameter which banlances distance term and confidence term. An intuitive attempt is to directly set λ to 0.5, which means that distance term and confidence term are equally important. But from our experiment results, such a simple setting does not bring about performance improvements. This is because that the motion of different human body parts is often different. For parts that move more vigorously, such as elbows, wrists, etc., the joint locations generated by the existing image-based methods often jitter greatly, i.e., the confidence of the predicted points is unreliable. In this case, the weight of distance should be greater than confidence to make joint motion more smooth and thus correct prediction errors.

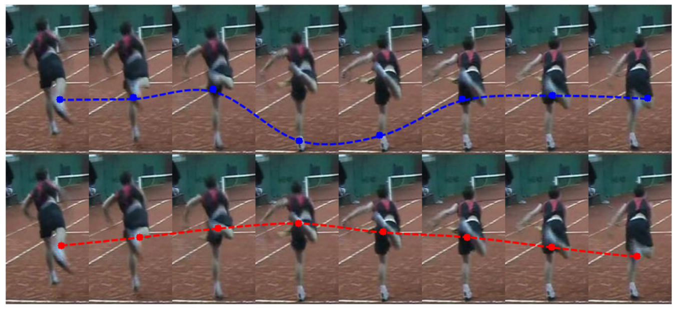

Fig. 4. The motion curves of predicted results and ground-truth labels. The blue dashed line is based on the prediction results. It is visually more jittery than the motion curve based on the annotations (red dashed line), which means a larger value of parameter λ is needed in such a case.

According to the animation principles [43], [44], the curve representation can avoid unnecessary flickers and make actions smooth, which conforms to the kinematics characteristics. Inspired by these facts, we fit the motion curve according to the movement of the key points. As depicted in Fig. 4, we found that the problematic prediction results often lead to an unsmooth curve, which means that these results do not conform to the characteristics of human motion. In such cases, we need to increase the value of parameter λ so that the refined joint locations consider more about the temporal consistency. Therefore, we propose a method to adaptively estimate the parameter λ according to the motion situation.

To avoid high computational complexity, we only focus on the problematic video clips. We observe through experiments that video clips with large gap between adjacent frames are often caused by jitter, which is exactly what we need to refine. Therefore, we propose a strategy toto find the problematic video clips as follows

$$Clip(x,y)=\begin{cases}1 & if \quad (\Delta_x \quad or \quad \Delta_y) > Thresh \\ 0 & otherwise\end{cases} \tag{5}$$

where Δ_x and Δ_y represent the difference in x and y coordinates between adjacent frames. For the sake of simplicity, we adopt the PCK metric introduced in [16] as the $Thresh$, which is defined as follows

$$Thresh = \alpha * \max(W_{bbox}, H_{bbox}) \tag{6}$$

where α is the weight parameter, W_{bbox} and H_{bbox} are the width and height of the bounding box, respectively. According to the general practice, we set α to 0.2.

Intuitively, if the keypoints predicted by the image-based methods are connected into a polyline, the greater jitter of the polyline, the larger λ we should use to make the refined results more smooth. Therefore, we can first fit the predicted keypoints with a curve and then evaluate the goodness of the curve fitting to choose a proper λ.

In order to reflect the motion characteristics of the joint points as accurately as possible, we apply a hampel filter on all point coordinates to reduce the influence of abnormal points on the motion curve. Afterwards, ridge regression is used to fit the motion characteristic curve

$$\begin{aligned}&\min_{w} \sum_{i=1}^{m}\left(y_i - x_i^T w\right)^2 \\ &\text{s.t. } \sum_{i=1}^{n} w_j^2 \le t\end{aligned} \tag{7}$$

The optimal ridge regression estimator can be obtained as

$$\hat{w} = \left(X^T X + kI\right)^{-1} X^T y \tag{8}$$

where k is the penalty term. Then we estimate the goodness of curve fitting, E, using

$$E = \frac{1}{Z^{\frac{1}{2}} + 1} \tag{9}$$

with Z being defined as follows

$$Z = \frac{\sum_{i=1}^{n}\left(|y_i - Y_i|^p\right)}{\sum_{i=1}^{n}\left(\left|Y_i - \bar{Y}\right|^p\right) + \varepsilon} \tag{10}$$

where ε is a very small number in order to avoid the denominator being 0. y_i is the fitted value, Y_i represents the original value, and $\bar{Y}$ represents the mean value of the original data.

Finally, we get the estimated value of λ according to the goodness of curve fitting using

$$\lambda = 1 - \min\left(E_x, E_y\right) \tag{11}$$

where E_x and E_y represent the goodness of curve fitting in the horizontal and vertical directions , respectively.

D. Algorithm

The detailed algorithm of the proposed method is shown in Algorithm 1. Table I summarizes the notations used in the algorithm. We first split the input heat map sequences into several video chucks according to the threshold. Afterwards, the module judges whether refinement is required according to the motion characteristics of each clip. For the video chucks to be refined, hierarchical dynamic programming method is applied to find the optimal graph path between heat map sequences. Finally, the optimal path will be transformed into the refined joint locations of the video chuck.

TABLE I
NOTATIONS IN ALGORITHM 1

I	video frames
H	heat map sequence
$bboxes$	human bounding boxes
$center$,$scale$	affine transformation parameters
L	slide window length
α	threshold parameter
Δ_x, Δ_y	the difference in x and y coordinates of each clip
λ	weight parameter of constraint function
$kernel$	kernel size of pooling operation
$pred$	local predictions of each refinement stage
$CropHm$	heat maps cropped from original sequence
$PooledHm$	pooled heat map sequence
$Cost$	the cost matrix of each clip
$OptimalPath$	optimal path generated by graph optimization
Y	refined joint locations
$\mathcal{N}_{\text{Pose}}$	baseline human pose estimation network
$\mathcal{F}_{\text{Diff}}$	function for calculating difference of each video clip
$\mathcal{F}_{\text{Iter}}$	function for calculating number of HDP iterations
$\mathcal{F}_{\text{GetHM}}$	function for cropping heatmaps
$\mathcal{F}_{\text{Pool}}$	function for pooling cropped heatmaps
$\mathcal{F}_{\text{GetCost}}$	function for calculating cost matrix
$\mathcal{F}_{\text{HDP}}$	function for hierarchical dynamic programming
$\mathcal{F}_{\text{RefinePred}}$	function for obtaining local pose predictions
$\mathcal{F}_{\text{GetFinalPreds}}$	function for obtaining final pose predictions

IV. EXPERIMENTS AND EVALUATIONS

A. Implementation details

To fairly compare the effect of our module in the existing state-of-the-art image-based methods, we use the released pre-trained models without any finetune. Specifically, we use [7], [9], [10], [45]–[47] as our baseline models, which have never been trained on the datasets we used to evaluate. The FastPose [45] is the new designed network from the team of AlphaPose [19]. We let OpenPose [7] generate a heat map with the same size as the input image in order to improve its accuracy, and other models generate a heat map with a size of 64×48 as default. In order to eliminate the impact of hardware as much

Algorithm 1: Hierarchical DP Module for Human Pose Refinement

```
Input  : frames $I$; bounding boxes $bboxes$; affine transformation parameters $center$ and $scale$; slide window length $L$;
         video filter threshold $\alpha$;
Output: optimal pose predictions of video $Y$
1  $H = \mathcal{N}_{Pose}(I, bboxes)$ ;                // obtain initial heat maps of each joints
2  Split original sequence $H$ into $C$ chunks, each with a length of $L$;
3  for $c = 1$ to $C$ do
4      $Thresh = \alpha * \max(W_{bboxes}, H_{bboxes})$ ; // calculate the threshold value for filtering this
         video clip
5      $\Delta_x, \Delta_y = \mathcal{F}_{\text{Diff}}(H, c)$ ;       // calculate the difference in motion of key points in
         adjacent frames
6      if $\max(\Delta_x, \Delta_y) > Thresh$ then // refine chunks with jitter above the filtering
         threshold
7          Fit motion curve using ridge regression;
8          Caculate goodness of curve fitting $E_x$ and $E_y$;
9          $\lambda = 1 - \min(E_x, E_y)$ ;                  // calculate the HDP parameter $\lambda$
10         $N = \mathcal{F}_{\text{Iter}}(H)$ ; // calculate the optimal number of iterations of HDP based on the
             heat map size
11         $pred = []$, $OptimalPath = []$;
12         for $i = 1$ to $N$ do
13             Compute pooling kernel size $kernel$;
14             $CropHm = \mathcal{F}_{\text{GetHM}}(H, c, OptimalPath)$; // crop specific region based on the path of
                 the previous iteration
15             $PooledHm = \mathcal{F}_{\text{Pool}}(CropHm, kernel)$ ;          // pool selected heat map region
16             $Cost = \mathcal{F}_{\text{GetCost}}(PooledHm, kernel, center, scale, \lambda)$ ;    // calculate the cost matrix
17             $OptimalPath = \mathcal{F}_{\text{HDP}}(Cost)$ ;       // obtain the optimal path of current stage
18             $pred = \mathcal{F}_{\text{RefinePred}}(pred, OptimalPath)$ ;    // obtain coarse pose estimation results
19         end
20         $Y = \mathcal{F}_{\text{GetFinalPreds}}(pred, center, scale)$ ;   // obtain final pose estimation results using
             affine transformation
21     end
22 end
```

as possible, all baseline models are run on a GTX1080Ti GPU. Our HDP module post-processes the heat maps in a single thread on a Xeon E5-2680 CPU. We first use the ground-truth bounding boxes as the detection results of the baseline models, crop the input frames and send it to network. Afterwards, we get and store the coordinates of the joint points, confidence scores and the heat map sequences. Finally, our Hierarchical DP Module is applied to refine the predictions from baseline models. It is worth noting that our module does not require additional training data and can work in online or offline manner, which can greatly expand the applications of in video pose estimation.

B. Datasets

Penn Action Dataset. Penn Action Dataset [49] is a large dataset containing in total 2326 video clips of 15 different actions, with 1258 clips for training and 1068 clips for testing. Note that we only use the 1068 video clips because our module does not require training. All frames are in RGB. The resolution of each frame is within the size of 640×480. Each frames has joint annotations of 13 joints including head, shoulders, elbows, wrists, hips, knees and ankles. An additional label indicates whether a joint is visible or not is also provided. Following previous works, we will only consider those visible joints during evaluation.

Sub-JHMDB Dataset. JHMDB [50] is another video-based dataset for pose estimation. To maintain the consistency with previous works, we only evaluate our module on a subset of JHMDB called sub-JHMDB dataset, which contains 316 clips with all 11200 frames in the same size. This subset contains only complete bodies and no invisible joint is annotated. Sub-JHMDB has 3 different split schemes, so we evaluate our module separately and report the average accuracy over these three splits. Similar to the Penn Action Dataset, we only use the test set and ignore the training set.

C. Evaluation Metrics

We use the Percentage Correct Keypoints (PCK) [16] as the metric to evaluate the performance of pose estimation. A prediction is considered to be correct if the distance between it and the true position is smaller than $\alpha \cdot \max(w, h)$, where h and w are the height and width of the ground-truth bounding box. α is set to 0.2 on both datasets for fairly comparing with image-based models. Besides, we deduce the human bounding

TABLE II
THE PCK METRIC COMPARISONS ON PENNACTION DATASET AMONG STATE-OF-THE-ART METHODS AND OUR MODULE. BOLD FONTS REPRESENT THE BEST RESULTS.

Method		Head	Sho	Elb	Wri	Hip	Knee	Ank	Mean
Thin-Slicing [31]	baseline	98.0	97.3	95.1	94.7	97.1	97.1	96.9	96.5
CPM [5]	baseline	98.6	97.9	95.9	95.8	98.1	97.3	96.6	97.1
LPM [34]	baseline	98.9	98.6	96.6	96.6	98.2	98.2	97.5	97.7
UniPose [41]	baseline	-	-	-	-	-	-	-	**99.3**
DCPose [48]	baseline	-	98.6	96.2	96.0	98.7	98.8	98.7	97.9
OpenPose [7]	baseline	-	97.1	93.6	92.7	96.0	97.1	96.8	95.6
	greedy	-	97.4	94.2	93.6	96.2	97.2	97.1	96.0
	HDP	-	**97.7**	**94.9**	**94.6**	**96.5**	**97.7**	**97.5**	**96.5**
SimpleBaseline [9]	baseline	-	98.2	95.8	95.7	99.0	98.8	98.3	97.7
	greedy	-	98.3	95.8	95.6	99	98.8	98.4	97.7
	HDP	-	**98.6**	**96.9**	**97.0**	**99.1**	**99.2**	**98.9**	**98.3**
HRNet [10]	baseline	-	98.4	96.1	96.3	98.5	98.6	97.9	97.6
	greedy	-	98.3	96.2	962	98.6	0 98.6	97.8	97.7
	HDP	-	**98.9**	**97.3**	**97.6**	**98.9**	**99.1**	**98.5**	**98.4**
FastPose [45]	baseline	-	98.2	95.8	95.7	99.0	98.8	98.3	97.7
	greedy	-	98.3	95.8	95.6	99.0	98.8	98.4	97.7
	HDP	-	**98.6**	**96.9**	**97.0**	**99.1**	**99.2**	**98.9**	**98.3**
TransPose [46]	baseline	-	98.3	95.4	94.6	98.5	98.4	97.7	97.2
	greedy	-	98.2	95.5	94.4	98.5	98.3	97.6	97.1
	HDP	-	**98.7**	**96.8**	**96.4**	**98.7**	**98.8**	**98.3**	**98.0**
LightHRNet [47]	baseline	-	97.8	94.2	94.4	98.6	98.0	97.7	96.9
	greedy	-	98.0	94.3	94.5	98.6	98.1	97.8	97.0
	HDP	-	**98.1**	**96.4**	**95.8**	**98.9**	**98.4**	**98.3**	**97.8**

TABLE III
THE PCK METRIC COMPARISONS ON SUB-JHMDB DATASET AMONG STATE-OF-THE-ART METHODS AND OUR MODULE. THE BEST RESULTS ARE IN BOLD.

Method		Head	Sho	Elb	Wri	Hip	Knee	Ank	Mean
Thin-Slicing [31]	baseline	97.1	95.7	87.5	81.6	98.0	92.7	89.8	92.1
CPM [5]	baseline	**98.4**	94.7	85.5	81.7	97.9	94.9	90.3	91.9
LPM [34]	baseline	98.2	96.5	89.6	86.0	98.7	95.6	90.9	93.6
DCPose [48]	baseline	-	**97.9**	**93.2**	**92.1**	**98.4**	**97.8**	**95.9**	**95.8**
OpenPose [7]	baseline	-	96.1	90.6	86.4	95.5	93.0	90.8	92.1
	greedy	-	96.3	90.9	86.8	95.7	93.3	91.2	92.4
	HDP	-	**97.3**	**92.5**	**89.2**	**96.7**	**94.6**	**93.2**	**93.9**
SimpleBaseline [9]	baseline	-	97.6	93.2	90.2	98.3	96.8	942	95.0
	greedy	-	97.7	93.4	90.3	98.3	96.8	94.3	95.1
	HDP	-	**98.2**	**95.3**	**93.6**	**98.8**	**97.7**	**95.5**	**96.5**
HRNet [10]	baseline	-	97.9	94.0	90.8	98.2	96.9	94.8	95.5
	greedy	-	97.9	94.1	91.0	98.3	97.0	94.9	95.5
	HDP	-	**98.5**	**96.2**	**93.2**	**98.7**	**98.0**	**96.2**	**96.8**
FastPose [45]	baseline	-	97.6	92.9	90.6	98.0	97.1	93.9	95.0
	greedy	-	97.6	93.0	90.8	98.1	97.1	94.2	95.1
	HDP	-	**98.4**	**94.9**	**93.2**	**98.5**	**98.0**	**95.6**	**96.5**
TransPose [46]	baseline	-	96.4	91.1	87.2	97.2	95.7	93.6	93.5
	greedy	-	96.4	91.3	87.4	97.3	95.8	93.8	93.7
	HDP	-	**97.4**	**94.3**	**90.8**	**98.0**	**97.1**	**95.5**	**95.5**
LightHRNet [47]	baseline	-	97.3	91.7	89.3	98.1	96.9	94.2	94.6
	greedy	-	97.4	91.8	89.8	98.2	96.8	94.5	94.8
	HDP	-	**97.8**	**93.6**	**91.9**	**98.4**	**97.6**	**95.8**	**95.9**

boxes from the puppet masks used for segmentation for sub-JHMDB datasets as done in [34]. It should be noted that since the models we use are not trained on these two datasets, they can only predict the position of the *nose* instead of the *head*. So we will not report the prediction results of the *head* during the experiments.

D. Objective Results and Quantitative Analysis

To validate the effectiveness of our Hierarchical DP Module, we first compare it with the state-of-the-art methods on Penn Action Dataset. Table II shows the performance of OpenPose [7], SimpleBaseline [9], HRNet [10], FastPose [45], TransPose [46] and LightHRNet [47]. The best results are shown in bold. The *baseline* means no additional refinement is applied to the backbone network. The *greedy* means we use greedy search algorithm to refine the predictions. And the *HDP* means the proposed Hierarchical DP Module is applied. Note that we do not report the results of *Head* because all of the pretrained models predict *Nose* instead of *Head*. Since the prediction accuracy of *Head* is usually the highest, the true overall accuracy of our module should be slightly higher than the ones we report here.

Several important observations can be obtained from Table II. First of all, compared with the greedy algorithm, our module has much better performance improvement on all

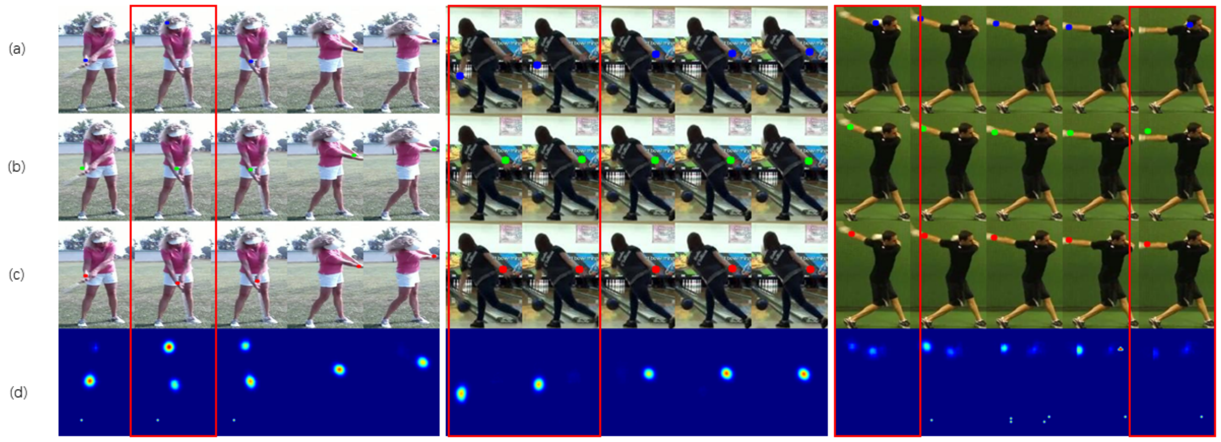

Fig. 5. Some qualitative examples on the Penn Action dataset: (a) original joint predictions (blue markers), (b)refined joint locations by our HDP module (green markers), (c) the groundtruth labels (red markers), (d) heat maps generated by baseline models, respectively. The problematic frames are marked with red bounding boxes.

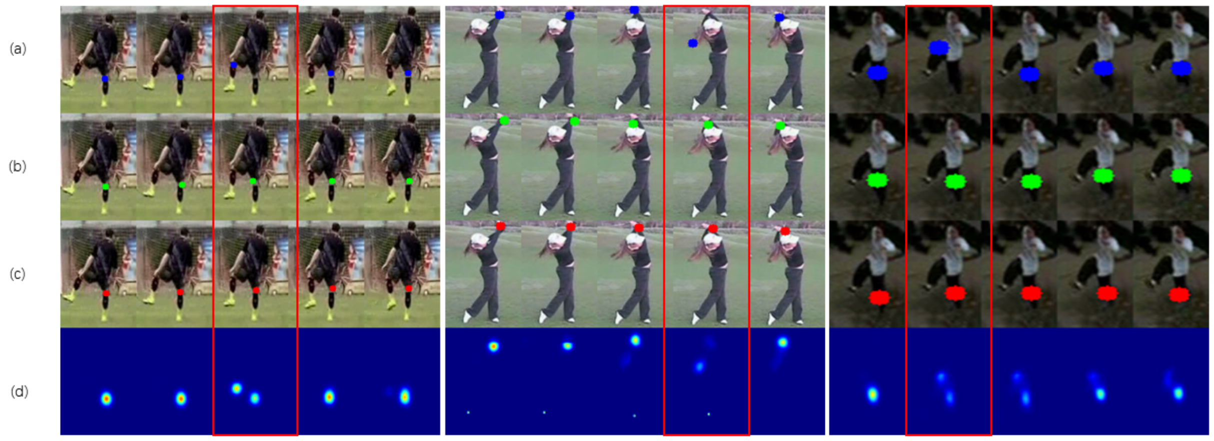

Fig. 6. Some qualitative examples on the Sub-JHMDB dataset: (a) original joint predictions (blue markers), (b)refined joint locations by our HDP module (green markers), (c) the groundtruth labels (red markers), (d) heat maps generated by baseline models, respectively. The problematic frames are marked with red bounding boxes.

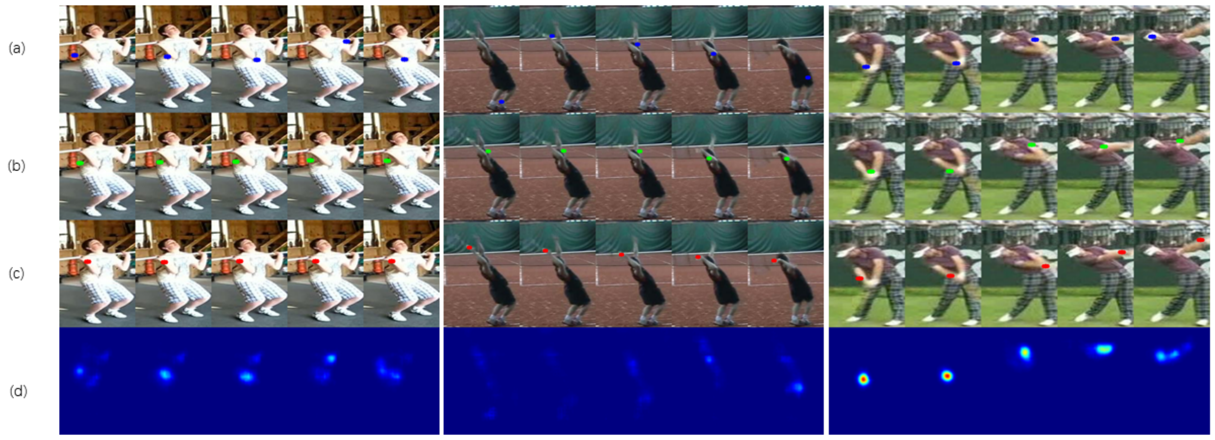

Fig. 7. Failure cases of our module: (a) original joint predictions (blue markers), (b)refined joint locations by our HDP module (green markers), (c) the groundtruth labels (red markers), (d) heat maps generated by baseline models, respectively.

baseline models. This may be because that greedy algorithm tends to fall into the local optimum, while the DP algorithm can obtain the global optimal results. Secondly, the gains mainly come from ”hard” joints. For example, the accuracy of *elbow* increases 1.3% and the accuracy of *wrist* increases 1.9% when our module is combined with OpenPose [7], which means our module can improve the accuracy of the joints that are subject to drastic movements or occlusion by utilizing temporal information. Thirdly, we also compare our module with some video-based pose estimation methods. Due to the good performance of the baseline models we use, the accuracy can be comparable to many video-based methods even without using our HDP module. For example, LightHRNet [47] has higher accuracy than Thin-Slicing [31], achieves 96.9%, while FastPose has the same overall accuracy of 97.7% with LPM [34]. After combining with HDP module, all baseline methods have performance gain. Three baseline methods [9], [10], [45] have improved by more than 0.6% to an accuracy of around 98.3%, second only to the UniPose method [41]. Probably because the distribution of the new dataset is different from that of the training dataset, the two state-of-the-art methods [46], [47] don’t outperform other methods on Penn Action dataset, but still gained more than 0.8% with the help of the HDP module, achieving 98.0% and 97.8% accuracy, respectively.

Similarly, we achieved improvements in every joints on Sub-JHMDB dataset as shown in Table III, especially those ”hard“ joints like *elbow*, *wrist* and *ankle*. Our module has achieved better performance improvement, which may be due to the poorer performance of the baseline models on these data. We observe that TransPose [46] has the largest improvement, reaching 2%, and the performance of the remaining three models is similar.

In order to better understand the effect of our proposed Hierarchical DP Module, we show some qualitative results on Penn Action and Sub-JHMDB Datasets in Fig.5 and 6 respectively. As illustrated in Fig.5 and 6, we visualize (a) original prediction results from image-based networks, (b) prediction results refined by our Hierarchical DP Module, (c) the ground truth labels, (d) heat maps generated by image-based networks. It can be observed that some jitter occurs when only image-based method is applied. For example, the first sample in Fig. 6 shows that baseline models can make errors when estimating symmetric joints. Two peaks appear in the heat map. The third sample in Fig. 6 shows that poor lighting condition can also leads to wrong predictions. Nevertheless, our module helps to improve the accuracy of those joints by better utilizing their historical locations. Therefore, the motion trajectory of the joint points is smoother and more stable.

E. Ablation Study

Analysis on all components. Our proposed method consists of three main components: Hierarchical DP module, sliding window acceleration, and adaptive DP parameter estimation. To validate the contribution of each component, we conduct ablation study on the Sub-JHMDB dataset [50].

In the ablation experiments, we focus on the accuracy of the results and the speed of inference. According to TABLE IV, the hierarchical dynamic programming module runs more efficiently than the original DP algorithm. The overall running speed is further increased after the combination of the sliding window method. However, the overall accuracy slips due to the lack of accurate estimation of the DP parameter λ. Therefore, with the addition of adaptive parameter estimation, the module can further improve the accuracy of pose estimation while meeting the real-time requirements.

TABLE IV
THE ABLATION EXPERIMENT ON SUB-JHMDB DATASET. BOLD FONTS REPRESENT THE BEST RESULTS. NOTE THAT BASELINE REFERS TO HRNET [10].

	FPS	Acc
Baseline	**35.2**	95.6
Baseline + DP	<0.2	95.6
Baseline + HDP	5.6	95.5
Baseline + HDP + slide window	25.4	95.9
Baseline + HDP + slide window + Adaptive Parameter	22	**96.8**

Analysis on temporal length. As shown in TABLE V, we investigate the effect of different time window lengths on performance. We observe that the accuracy of pose estimation does not always improve with increasing time length. Specifically, our module performs better with video clips of about 20 frames, when the module can estimate good motion characteristics and is not distracted by excess video frames.

TABLE V
THE PERFORMANCE COMPARISONS ON SUB-JHMDB DATASET WITH DIFFERENT LENGTH OF SLIDE WINDOW. BOLD FONTS REPRESENT THE BEST RESULTS. NOTE THAT FULL MODEL REFER TO THE HRNET [10] COMBINED WITH OUR PROPOSED HDP MODULE

	5-Frames	10-Frames	20-Frames	30-Frames	40-Frames
Full model	96.0	96.3	**96.8**	96.6	96.4

Analysis on problematic video filter. In the HDP algorithm, we added a problematic video detection step to further improve the overall speed. We also report the performance data of the module under different detection thresholds. As shown in Fig.8, when α becomes larger, the number of filtered problem videos decreases and the module runs faster. But accordingly, the accuracy rate also appears to decrease. However, α should not be too small which will lead to a large number of clips that do not need to be processed cannot be ignored, thus seriously slowing down the processing speed of the module. After weighing the speed and accuracy, we choose to set α to 0.2.

TABLE VI
THE PERFORMANCE COMPARISONS ON SUB-JHMDB DATASET WITH DIFFERENT POOLING KERNELS. BOLD FONTS REPRESENT THE BEST RESULTS.

Input heat map	heat map (stage 1)	heat map (stage 2)	FPS	Acc
64×48	16×16	4×3	11	**97.0**
	8×8	8×8	**127**	96.8
	4×4	16×12	19	96.4

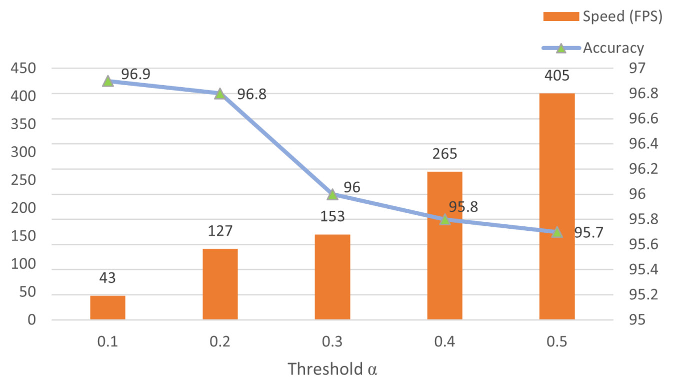

Fig. 8. Analysis on problematic video filter. The inference speed(FPS) and accuracy of the module under different thresholds α are presented.

Analysis on pooled heat map size. One remaining but critical problem is: how do different sizes of pooled heat maps at each stage affect the performance of the module? As shown in Table VI, we show the results of some experiments. We choose HRNet [10] as the backbone and combined it with our HDP module with different pooling kernels. It can be seen that at the first stage, the bigger the pooled heat map is, the smaller the quantization error is, which will improve the final accuracy. However, the bigger the pooled heat map is, the higher the computation complexity of the graph path search algorithm will be, which will slow down the inference speed. In practice, we ensure that the size of the pooled heat map of each stage is appropriate, e.g. for an original heat map of size 64×48, we use a pooling kernel of size 8×6 to produce a map of size 8×8 at each stage, which achieves a balance of speed and accuracy.

F. Inference Speed

Inference speed is crucial for video-based applications. Since our module is designed as a plug-in which can be used as a component of existing image-based pose estimation networks to improve their performance on video data, its running speed is very important. We conduct the experiment on the test set of Penn Action Dataset, using the heat map sequences produced by SimpleBaseline [9]. The results are illustrated in Table VII. Note that the original dynamic programming method has too high computational complexity and takes up a lot of memory, so it is impossible to run a complete test. Therefore, we randomly pick a few short video clip for test.

TABLE VII
INFERENCE SPEED TEST AMONG DIFFERENT GRAGH PATH SELECT ALGORITHMS ON PENN ACTION DATASET.

Optimization Method	Inference Speed (FPS)
Greedy Search	270
Original DP	0.2*
Hierarchical DP	127

We can see that original dynamic programming method is extremely slow, which needs more than 5s per-frame to refine the joint predictions. Our Hierarchical DP Module is much faster, only needs 7.8ms per-frame, about 641x faster than original DP. Greedy search runs the fastest, reaching 270 fps, but its accuracy is far lower than our Hierarchical DP module according to Table II and Table III.

To more fully compare the performance benefits of our modules, we perform speed tests on the Sub-JHMDB dataset for state-of-the-art methods. As can be seen from the Table VIII, although the video-based pose estimation method can achieve high accuracy, it cannot meet the need for real-time performance. For example, DCPose [48] requires three consecutive frames to optimize the result of the middle frame, so the running speed is greatly reduced. In contrast, with the addition of our efficient HDP module, the image-based baseline methods have a small loss in speed but a significant gain in accuracy.

TABLE VIII
THE RESPONSE TIME COMPARISONS ON SUB-JHMDB DATASET AMONG STATE-OF-THE-ART METHODS AND OUR MODULE. THE BEST RESULTS ARE IN BOLD.

Method	Speed (FPS)	Accuracy
LPM [34]	7	93.6
DCPose [48]	8	95.8
OpenPose [7] + HDP	20	93.9
SimpleBaseline [9] + HDP	**31**	96.5
HRNet [10] + HDP	22	**96.8**
FastPose [45] + HDP	27	96.5
TransPose [46] + HDP	24	95.5
LightHRNet [47] + HDP	15	95.9

G. Failure Case

We also present some failure cases in Fig.7. We can observe that our module can not refine the key points when the percentage of wrong points in the clip is too high. When encountering rare actions or pictures with poor quality, the baseline model will be confused and produce ambiguous heat maps. These heat maps contain a lot of misleading information so that our module cannot correct the wrong prediction results well.

V. CONCLUSION

In this paper, we present a lightweight but effective module named Hierarchical DP Module for video pose estimation. It can be used as a component of existing image-based pose estimation networks to improve their performance on video data. Experiments on two widely-used datasets show that our module significantly improves the performance of state-of-the-art image-based methods, especially for those joints that are subject to drastic movements or occlusion. Our module can process hundreds of frames per second in the experiment, even in single-threaded mode.

REFERENCES

[1] X. Jiang, K. Xu, and T. Sun, "Action recognition scheme based on skeleton representation with ds-lstm network," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 30, no. 7, pp. 2129–2140, 2020.

[2] Z. Yang, Y. Li, J. Yang, and J. Luo, "Action recognition with spatio–temporal visual attention on skeleton image sequences," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 29, no. 8, pp. 2405–2415, 2019.

[3] J. Lei, L. Niu, H. Fu, B. Peng, Q. Huang, and C. Hou, "Person re-identification by semantic region representation and topology constraint," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 29, no. 8, pp. 2453–2466, 2018.

[4] Z. Zheng, L. Zheng, and Y. Yang, "Pedestrian alignment network for large-scale person re-identification," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 29, no. 10, pp. 3037–3045, 2019.

[5] S.-E. Wei, V. Ramakrishna, T. Kanade, and Y. Sheikh, "Convolutional pose machines," in *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, 2016, pp. 4724–4732.

[6] A. Newell, K. Yang, and J. Deng, "Stacked hourglass networks for human pose estimation," in *European conference on computer vision*. Springer, 2016, pp. 483–499.

[7] Z. Cao, T. Simon, S.-E. Wei, and Y. Sheikh, "Realtime multi-person 2d pose estimation using part affinity fields," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 7291–7299.

[8] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask r-cnn," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 2961–2969.

[9] B. Xiao, H. Wu, and Y. Wei, "Simple baselines for human pose estimation and tracking," in *Proceedings of the European conference on computer vision (ECCV)*, 2018, pp. 466–481.

[10] K. Sun, B. Xiao, D. Liu, and J. Wang, "Deep high-resolution representation learning for human pose estimation," in *CVPR*, 2019.

[11] V. Ramakrishna, T. Kanade, and Y. Sheikh, "Tracking human pose by tracking symmetric parts," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2013, pp. 3728–3735.

[12] D. Zhang and M. Shah, "Human pose estimation in videos," in *Proceedings of the IEEE International Conference on Computer Vision*, 2015, pp. 2012–2020.

[13] M. Ruder, A. Dosovitskiy, and T. Brox, "Artistic style transfer for videos," in *German conference on pattern recognition*. Springer, 2016, pp. 26–36.

[14] D. Chen, J. Liao, L. Yuan, N. Yu, and G. Hua, "Coherent online video style transfer," in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 1105–1114.

[15] M. Andriluka, S. Roth, and B. Schiele, "Pictorial structures revisited: People detection and articulated pose estimation," in *2009 IEEE conference on computer vision and pattern recognition*. IEEE, 2009, pp. 1014–1021.

[16] Y. Yang and D. Ramanan, "Articulated human detection with flexible mixtures of parts," *IEEE transactions on pattern analysis and machine intelligence*, vol. 35, no. 12, pp. 2878–2890, 2012.

[17] L. Pishchulin, M. Andriluka, P. Gehler, and B. Schiele, "Poselet conditioned pictorial structures," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2013, pp. 588–595.

[18] Y. Tian, C. L. Zitnick, and S. G. Narasimhan, "Exploring the spatial hierarchy of mixture models for human pose estimation," in *European Conference on Computer Vision*. Springer, 2012, pp. 256–269.

[19] H.-S. Fang, S. Xie, Y.-W. Tai, and C. Lu, "RMPE: Regional multi-person pose estimation," in *ICCV*, 2017.

[20] I. Lifshitz, E. Fetaya, and S. Ullman, "Human pose estimation using deep consensus voting," in *European Conference on Computer Vision*. Springer, 2016, pp. 246–260.

[21] W. Yang, S. Li, W. Ouyang, H. Li, and X. Wang, "Learning feature pyramids for human pose estimation," in *proceedings of the IEEE international conference on computer vision*, 2017, pp. 1281–1290.

[22] X. Chu, W. Yang, W. Ouyang, C. Ma, A. L. Yuille, and X. Wang, "Multi-context attention for human pose estimation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 1831–1840.

[23] R. Girshick, "Fast r-cnn," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 1440–1448.

[24] Y. Chen, Z. Wang, Y. Peng, Z. Zhang, G. Yu, and J. Sun, "Cascaded pyramid network for multi-person pose estimation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 7103–7112.

[25] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[26] J. Wang, K. Sun, T. Cheng, B. Jiang, C. Deng, Y. Zhao, D. Liu, Y. Mu, M. Tan, X. Wang *et al.*, "Deep high-resolution representation learning for visual recognition," *IEEE transactions on pattern analysis and machine intelligence*, 2020.

[27] B. Cheng, B. Xiao, J. Wang, H. Shi, T. S. Huang, and L. Zhang, "Higherhrnet: Scale-aware representation learning for bottom-up human pose estimation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

[28] T. Pfister, K. Simonyan, J. Charles, and A. Zisserman, "Deep convolutional neural networks for efficient pose estimation in gesture videos," in *Asian Conference on Computer Vision*. Springer, 2014, pp. 538–552.

[29] A. Jain, J. Tompson, Y. LeCun, and C. Bregler, "Modeep: A deep learning framework using motion features for human pose estimation," in *Asian conference on computer vision*. Springer, 2014, pp. 302–315.

[30] T. Pfister, J. Charles, and A. Zisserman, "Flowing convnets for human pose estimation in videos," in *Proceedings of the IEEE International Conference on Computer Vision*, 2015, pp. 1913–1921.

[31] J. Song, L. Wang, L. Van Gool, and O. Hilliges, "Thin-slicing network: A deep structured model for pose estimation in videos," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 4220–4229.

[32] P. Weinzaepfel, J. Revaud, Z. Harchaoui, and C. Schmid, "Deepflow: Large displacement optical flow with deep matching," in *Proceedings of the IEEE international conference on computer vision*, 2013, pp. 1385–1392.

[33] G. Gkioxari, A. Toshev, and N. Jaitly, "Chained predictions using convolutional neural networks," in *European Conference on Computer Vision*. Springer, 2016, pp. 728–743.

[34] Y. Luo, J. Ren, Z. Wang, W. Sun, J. Pan, J. Liu, J. Pang, and L. Lin, "Lstm pose machines," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 5207–5215.

[35] M. Lin, L. Lin, X. Liang, K. Wang, and H. Cheng, "Recurrent 3d pose sequence machines," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 810–819.

[36] R. Girdhar, G. Gkioxari, L. Torresani, M. Paluri, and D. Tran, "Detect-and-track: Efficient pose estimation in videos," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 350–359.

[37] Y. Xiu, J. Li, H. Wang, Y. Fang, and C. Lu, "Pose flow: Efficient online pose tracking," *arXiv preprint arXiv:1802.00977*, 2018.

[38] S. Liu, Y. Li, and G. Hua, "Human pose estimation in video via structured space learning and halfway temporal evaluation," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 29, no. 7, pp. 2029–2038, 2018.

[39] D. Zecha, M. Einfalt, and R. Lienhart, "Refining joint locations for human pose tracking in sports videos," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, June 2019.

[40] X. Nie, Y. Li, L. Luo, N. Zhang, and J. Feng, "Dynamic kernel distillation for efficient pose estimation in videos," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2019.

[41] B. Artacho and A. Savakis, "Unipose: Unified human pose estimation in single images and videos," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 7035–7044.

[42] ——, "Waterfall atrous spatial pooling architecture for efficient semantic segmentation," *Sensors*, vol. 19, no. 24, p. 5361, 2019.

[43] J. Lasseter, "Principles of traditional animation applied to 3d computer animation," in *Proceedings of the 14th annual conference on Computer graphics and interactive techniques*, 1987, pp. 35–44.

[44] R. Williams, *The animator's survival kit: a manual of methods, principles and formulas for classical, computer, games, stop motion and internet animators*. Macmillan, 2012.

[45] Z. Chen, "Fastpose: A fast and small human pose estimator," https://github.com/ZexinChen/FastPose.

[46] S. Yang, Z. Quan, M. Nie, and W. Yang, "Transpose: Keypoint localization via transformer," in *IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021.

[47] C. Yu, B. Xiao, C. Gao, L. Yuan, L. Zhang, N. Sang, and J. Wang, "Lite-hrnet: A lightweight high-resolution network," in *CVPR*, 2021.

[48] Z. Liu, H. Chen, R. Feng, S. Wu, S. Ji, B. Yang, and X. Wang, "Deep dual consecutive network for human pose estimation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 525–534.

[49] W. Zhang, M. Zhu, and K. G. Derpanis, "From actemes to action: A strongly-supervised representation for detailed action understanding," in

Proceedings of the IEEE International Conference on Computer Vision, 2013, pp. 2248–2255.

[50] H. Jhuang, J. Gall, S. Zuffi, C. Schmid, and M. J. Black, "Towards understanding action recognition," in *Proceedings of the IEEE international conference on computer vision*, 2013, pp. 3192–3199.