Fast and Unsupervised Action Boundary Detection for Action Segmentation

Zexing Du, Xue Wang, Guoqing Zhou, Qing Wang
School of Computer Science, Northwestern Polytechnical University, Xi'an 710072, China

Abstract

To deal with the great number of untrimmed videos produced every day, we propose an efficient unsupervised action segmentation method by detecting boundaries, named action boundary detection (ABD). In particular, the proposed method has the following advantages: no training stage and low-latency inference. To detect action boundaries, we estimate the similarities across smoothed frames, which inherently have the properties of internal consistency within actions and external discrepancy across actions. Under this circumstance, we successfully transfer the boundary detection task into the change point detection based on the similarity. Then, non-maximum suppression (NMS) is conducted in local windows to select the smallest points as candidate boundaries. In addition, a clustering algorithm is followed to refine the initial proposals. Moreover, we also extend ABD to the online setting, which enables real-time action segmentation in long untrimmed videos. By evaluating on four challenging datasets, our method achieves state-of-the-art performance. Moreover, thanks to the efficiency of ABD, we achieve the best trade-off between the accuracy and the inference time compared with existing unsupervised approaches.

1. Introduction

There are tera-bytes of videos uploaded to cloud or edge storage to be processed every day. Efficiently analyzing the contents of these untrimmed videos is a nontrivial step towards real-world deployment, which has a great range of applications from video retrieval to surveillance analysis [9, 27, 33, 46]. Tremendous progress has recently been made on supervised action segmentation due to the introduction of large-scale datasets and the development of deep neural networks [12, 21, 26, 43]. However, building fully supervised learning models requires manual data labeling, which is slow, expensive and error-prone. Thus unsupervised action segmentation has gained great popularity.

Due to the temporal coherence of videos, when segmenting actions from a long, untrimmed video, human beings would pay primary attention to detecting action boundaries,

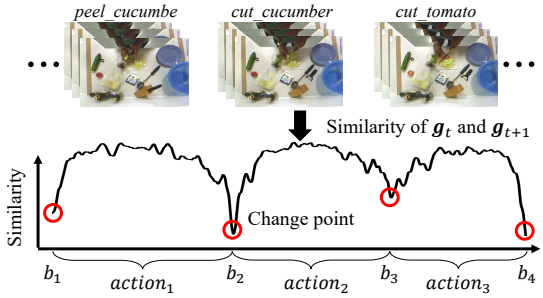


Figure 1. The overview of our method. g_t represents the smoothed feature for the t-th frame, and b_i the i-th detected boundary. We observe that the calculated frame-wise similarities based on the smoothed frame features for individual action instance take a "⊓"-shaped curve. Action boundaries can be directly pinpointed by detecting change points along the similarity curve.

and take the frames between adjacent temporal boundaries as an action instance. Building on this insight, we propose an efficient and effective unsupervised action segmentation method by detecting action boundaries, dubbed action boundary detection (ABD). The overview of the proposed method is illustrated in Figure 1.

Motivated by the Canny detector [6] in image processing and the segmentation method [18] in time series, we propose a bottom-up action segmentation method. For action segmentation, the ideal features should be consistent within an action and inconsistent across different actions. However, due to the occlusion, changing viewpoint or lighting, the features within an action may fail to maintain strictly consistent as expected. Therefore, we first utilize a smoothing filter on original features to dilute the effect of noise. Moreover, different from the Canny detector that finds intensity gradients of images, we pinpoint temporal action boundary proposals based on the similarity between adjacent frames. Then, we adopt non-maximum suppression (NMS) inside local windows and select smallest points as initial candidate action boundaries. In addition, we introduce a bottom-up method to refine initial segmentation results based on semantic segment-wise similarity.

There have been several attempts to detect action boundaries by stacking complicated neural networks or generating less-relevant pseudo labels [11, 17, 32, 48]. Directly depend-

ing on the similarity between frames is more reliable and interpretative. We hope our exploration will motivate communities to rethink the fundamental roles of feature similarity within and across actions for action segmentation.

Our method has the following advantages:

1) No training is needed. A powerful family of models for weakly- and unsupervised representations are collected under the umbrella of "self-supervised" learning, which concentrates on acquiring pseudo-labels that can be used for supervised training to obtain higher-level semantic features [11, 28, 30, 36, 38, 40, 42]. However, the less-related labels have limited contribution to the detection results, and the train-and-transfer strategy would sacrifice the generalizability of models. Instead, we directly exploit original features without any training, guaranteeing the robustness and transferability of our method.

2) Fast and state-of-the-art. The proposed method achieves the best trade-off between model accuracy and inference time compared with weakly supervised and unsupervised action segmentation methods [11, 39, 42]. Specifically, our method yields competitive performance compared with state-of-the-art approaches on four challenging datasets and significantly increases the inference speed. In addition, our ABD can be further extended into the online setting, which enables real-time action segmentation in untrimmed videos.

2. Related Work

This section reviews closely related work on weakly supervised and unsupervised action segmentation. We also discuss other approaches in sequence processing that are related to our work.

2.1. Weakly Supervised Action Segmentation

Weakly supervised methods can be classified into transcript- and set-constrained categories, where the former knows the label and order of actions [5, 11, 15, 22, 23, 28, 36, 42] and the latter just knows what action occurs [13, 29, 31, 37]. Kuehne *et al.* [22] iteratively generate and refine pseudo labels based on the hidden Markov model (HMM) and a Gaussian mixture model (GMM). Some methods [11, 23, 36] replace the GMM with a recurrent neural network (RNN), but keep to refine the pseudo labels iteratively. Richard *et al.* [38] propose a learning algorithm with a Viterbi-based loss that allows for online and incremental learning by directly leveraging transcripts. Souri *et al.* [42] recently present an end-to-end approach for weakly supervised action segmentation based on a two-branch neural network, and a novel mutual consistency loss is introduced to enforce the consistency of two redundant representations. For set-constrained supervision, Li and Todorovic [29] formulate a set-constrained Viterbi algorithm (SCV) to generate the MAP prediction, which is used as a frame-wise pseudo ground truth in the HMM training. SCT [13] divides the video into smaller temporal ranges and predicts for each region the action label and its length. An anchor constrained Viterbi algorithm (ACV) [31] is introduced to generate the pseudo labels, where anchors representing salient action parts are estimated for each action from a given ground-truth set.

2.2. Unsupervised Action Segmentation

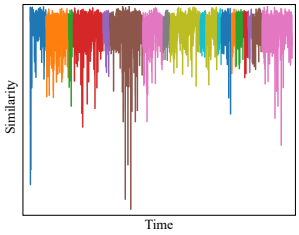
There are two groups of unsupervised action segmentation methods: the first group [24, 30, 38, 40, 45] targets solving the temporal action relations globally over a collection of videos from the same activity. The second group [1, 39] achieves action segmentation based on individual videos by detecting boundary changes. Kukleva *et al.* [24] present a continuous temporal embedding to enforce clusters of frames at similar temporal stages, which is next combined with a frame-to-cluster assignment based on Viterbi decoding. ASAL [30] trains an RNN to recognize positive and negative action sequences. An HMM is utilized to model the action lengths, and the Viterbi algorithm is used to infer a MAP action segmentation. LSTM+AL [1] fine-tunes a pre-trained model with LSTM, using future frame prediction as supervision to learn frame embeddings. TW-FINCH [39] introduces a temporally-weighted hierarchical clustering algorithm to group semantically consistent frames of the video. However, constructing all pairwise correlations brings in overwhelming computations.

2.3. Others

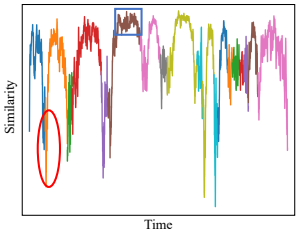
Our method is also related to change point detection algorithms [2, 14, 34, 44] and time series segmentation approaches [16, 18–20, 35], which mainly segment time series by regression with a high computational cost. In addition, shot boundary detection (SBD) [4, 41], which aims at automatically detecting the boundaries between shots in video, is also relevant to our approach. And the large body of anomaly detection algorithms [8] also conduct the similar procedure as our method. In the spatio-temporal dimensions, actions show the property of internal consistency within actions and external discrepancy across actions. Based on this insight, we propose a straightforward yet effective method to detect action boundaries based on the similarity between adjacent frames.

3. Method

Given a video with N frames $X = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_N\}$, the goal of action segmentation is to group these frames into K clusters, where K is the number of action classes. In this section, we first elaborate on the similarity between frames. Then, we show how to detect temporal boundaries based on the similarity. Furthermore, a clustering algorithm



(a) The similarity between adjacent frames, i.e., x_t and x_{t+1}, where x represents the original feature.



(b) The similarity between smoothed frames, i.e., g_t and g_{t+1}, where g represents the smoothed feature.

Figure 2. The similarity between frames on *rgb-19-1* in 50Salads [43]. Different colors represent different segments in ground truth. Best viewed in color.

is employed for refining the detected segments. Moreover, we further extend our method into the online setting.

3.1. Similarity between Frames

As mentioned in Section 1, the ideal features should be consistent within actions and inconsistent across boundaries. Unlike Canny detector [6] calculates the intensity gradient to detect boundaries in grayscale images, we directly calculate the cosine similarity between adjacent frames in the temporal dimension, which is formulated as

$$S_t = \frac{\boldsymbol{x}_t \cdot \boldsymbol{x}_{t+1}}{||\boldsymbol{x}_t||\ ||\boldsymbol{x}_{t+1}||}. \tag{1}$$

For better understanding this formulation, we visualize it in Figure 2a. We find it is pretty hard to detect action boundaries, as the similarity between $\boldsymbol{x}_t$ and $\boldsymbol{x}_{t+1}$ varies sharply along the whole temporal dimension. We conjecture it is because the hard frames caused by occlusion, viewpoint or lighting changing result in inconsistency within actions. Similar with Canny detector using Gaussian filters for smoothing images and preventing false detection caused by noise, here we conduct smoothing filters along the temporal dimension to mitigate the noise for action boundary detection. We could use average or Gaussian filters, which could be formulated as

$$\boldsymbol{g}_t = \frac{\sum_{i=t-k}^{t+k} W(\boldsymbol{x}_t, \boldsymbol{x}_i)\boldsymbol{x}_i}{\sum_{i=t-k}^{t+k} W(\boldsymbol{x}_t, \boldsymbol{x}_i)}, \tag{2}$$

where W is the filter kernel with size $(2k+1)$.

Therefore, the similarity between frames could be replaced by the similarity between $\boldsymbol{g}_t$ and $\boldsymbol{g}_{t+1}$. As shown in Figure 2b, the similarity curve of each segment has the shape of "⊓". Action boundaries can thus be found from the change points, where a significant change occurs locally. Moreover, from another perspective, the smoothing operation allows $\boldsymbol{g}_t$ to be expressed in terms of multiple neighbouring frames. Since the correlations are established between groups of frames instead of individual frames, more robust and smoother similarities could be established in this way.

3.2. Action Boundary Detection

Intuitively, action boundaries are most likely to coincide with the change points which indicate the dramatic change of actions. However, as shown in Figure 2b, it is still not easy to find boundaries in such a rough curve. Specifically, we need to avoid the pseudo boundaries within an action instance (blue rectangle) and suppress the ambiguous responses near the true boundaries (red circle).

To account for these spurious responses near boundaries, we utilize the NMS algorithm to reduce them i.e., the *non-minimum* values in the similarity. In detail, we conduct NMS in local windows along the temporal dimension and select the smallest points as candidate boundaries. The length of local window L will be discussed in experiments. Moreover, the pseudo boundaries in the blue rectangle Figure 2b are mainly caused by small changes within actions. To deal with these spurious responses, we propose a bottom-up refinement to cluster these segments.

3.3. Refinement

After detecting boundaries, we obtain candidate action boundaries $B = \{b_1, b_2, \ldots, b_M\}$, where $b_M = N$. The video is therefore divided into M segments. However, M is always greater than the number of action labels K, which is mainly caused by pseudo boundaries in actions. In addition, actions that happen multiple times can also lead to this result. Therefore, we present a bottom-up refinement to further cluster segments to K classes.

Firstly, we initialize the segment feature by averaging the features within each segment. The video can be denoted as $\mathcal{P}_i = \{\hat{\boldsymbol{x}}_1, \hat{\boldsymbol{x}}_2, \ldots, \hat{\boldsymbol{x}}_M\}$, where $\hat{\boldsymbol{x}}_m$ is the feature of the m-th segment. Then we calculate the similarity matrix by computing pairwise distances

$$S(i,j) = \frac{\hat{\boldsymbol{x}}_i \cdot \hat{\boldsymbol{x}}_j}{||\hat{\boldsymbol{x}}_i||\ ||\hat{\boldsymbol{x}}_j||}. \tag{3}$$

It is worth noting that $M << N$. Unlike TW-FINCH [39] which calculates the similarity between every frame pair, computing the segment-wise similarity merely imports negligible computational cost. The entry $S_{ij(i\neq j)}$ with the maximum value is selected and the corresponding $\hat{\boldsymbol{x}}_i$ and $\hat{\boldsymbol{x}}_j$ are considered as the most similar segments, which are merged to generate a new segment with the averaging of $\hat{\boldsymbol{x}}_i$ and $\hat{\boldsymbol{x}}_j$. We repeat the merging process until the number of action labels reduces to K. The main steps of the proposed algorithm are shown in Algorithm 1.

Discussion. Unlike [39] computes both semantic and temporal distances, our work pays no attention to the temporal distance. For frame-wise clustering used in [39], which

Algorithm 1 Refinement

Input: # of action labels K, Video $X = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_N\}$ and boundary candidates $B = \{b_1, b_2, \ldots, b_M\}$.
Output: Partition $\mathcal{P}_K$.

1: Initialize segment features $\mathcal{P}_i = \{\hat{\boldsymbol{x}}_1, \hat{\boldsymbol{x}}_2, \ldots, \hat{\boldsymbol{x}}_M\}$ with X and B.
2: **while** # of classes for segments $\mathcal{P}_i$ is greater than K **do**
3: Compute similarity matrix between segment-wise features via Equation (3).
4: Detect the most similar link (i, j).
5: Update labels in $\mathcal{P}_i$ and merge the corresponding features $\hat{\boldsymbol{x}}_i, \hat{\boldsymbol{x}}_j$.
6: **end while**

breaks the temporal consistency and results in severe over-segmentation errors, temporally modulated correlations are used to merge these over-segmented segments. However, this strategy focuses more on adjacent frames and cannot cluster distant segments which belong to the same action class. In contrast, our method segments videos by detecting boundary, and the usage of NMS successfully avoids over-segmentation. Performing the clustering algorithm on the semantic similarity matrix can effectively group similar segments even when they are temporal distant.

3.4. Online Action Boundary Detection

The online action segmentation is important for real-time vision applications, such as camera surveillance and self-driving. The local property of our method makes it easy to update adaptively in an online manner.

For offline action segmentation, like previous literature [24, 30, 39, 45], we take the number of action labels K as prior knowledge for refinement. However, it is infeasible to know the number of action labels under an online setting. Therefore, we propose an online ABD method which enables real-time action segmentation in long untrimmed videos.

Firstly, we smooth features only using previous frames, i.e., $\boldsymbol{g}_t$ is smoothed by $\{\boldsymbol{x}_{t-k}, \boldsymbol{x}_{t-k+1}, \ldots, \boldsymbol{x}_t\}$, and calculate similarities between smoothed frames following the statement in Section 3.1. Then we detect action boundaries based on the similarity. Like the offline ABD, we still perform NMS within a local window to suppress redundant responses near boundaries. Note that the window is now centered at frame $(t - L/2)$. Under this circumstance, we stack several frames into a group and process them online. For pseudo boundaries within actions, we no longer follow the procedure in Section 3.3 since the number of action labels K is unknown. Alternatively, we select a threshold to filter out edge frames with a high similarity value and preserve edge frames with a small one. As the similarity could vary greatly across videos and datasets, it is impossible to set a fixed threshold. To avoid struggling with the empirical threshold which needs sophisticated adjustments, we adaptively set it as the lower quartile of the similarity before time t. Considering only a fraction of frames are close to the boundaries (which would result in low similarity), this setting works well in filtering out weak boundaries.

4. Experiments

Datasets. To evaluate the performance of our method, we conduct experiments on four challenging dataset: Breakfast [21], YouTube Instructional Videos [3], Hollywood Extended [5] and 50Salads [43].

Breakfast contains 1,712 videos, of which the lengths vary from a few seconds to several minutes. The dataset contains 48 different actions for breakfast preparation where every video has 6.9 actions on average.

YouTube Instructional Videos contains 150 videos and the average length is around 2 minutes. There are 5 activities in this dataset, including *making coffee*, *cpr*, *jumping car*, *changing car tire* and *potting a plant*. The fraction of background is around 63.5%, which is the largest one in these four datasets.

Hollywood Extended has 937 video sequences with around 800,000 frames in total. The videos contain 16 different classes, and each video has about 2.5 action instances. There are roughly 61% background frames.

50Salads contains 4.5 hours of different people making salads and each video has about 10k frames on average. We evaluate two different action granularity levels, which include 19 and 12 action classes [43] respectively.

Features. For a fair comparison, we use the same features as [13, 24, 39, 40], which are frame-wise Fisher vectors of improved dense trajectory (IDT) [47] (Breakfast, Hollywood and 50Salads) and histogram of local optical flow (HOF) descriptors [25] (YouTube Instructional Videos).

Evaluation Metrics. Since our method outputs temporal boundaries without particular correspondences to the ground-truth labels, we require a one-to-one mapping between the outputs and the ground-truth labels. Following [1, 24, 39, 40], we utilize the Hungarian algorithm to generate this mapping based on the overlap between matched segments. Since our method does not concern the segment labels, we conduct this mapping on the video level as in [1, 39]. We also report the F1 score and mean over frames (MoF) for all datasets as used in previous works [24]. For evaluation on Hollywood Extended, we report Jaccard index as intersection over union (IoU) as an additional measurement.

Implementation Details. Unlike most existing methods [24, 39] evaluating the performance on the ground-truth number of classes for each activity, our method places less constraint on activities. Therefore, the parameter K used

Breakfast			
Weakly Supervised			MoF
NN-Vit. [38]			43.0
MuCon [42]			48.5
CDFL [28]			50.2
Unsupervised	Hungarian	F1	MoF
CTE [24]	Activity	26.4	41.8
VTE-UNET [45]	Activity	—	48.1
ASAL [30]	Activity	37.9	52.5
Equal Split [39]	Video	—	34.8
LSTM+AL [1]	Video	—	42.9
TW-FINCH [39]	Video	49.8	62.7
Our ABD	Video	52.3	64.0

Table 1. Comparison to the state-of-the-art on Breakfast. The dash indicates "not reported".

in this paper is the average number of action classes for a specific dataset. And we would also discuss the setting of K later.

4.1. Comparison to State-of-the-art

Breakfast. In Table 1, we report the performance comparison to the state-of-the-art methods on Breakfast. Besides unsupervised methods, we also compare with several supervised and weakly supervised methods, which can be severed as the upper bounds of our method. ABD outperforms all unsupervised methods. Significant improvements are achieved when compared with Equal Split [39], which demonstrates our ABD does work in pinpointing action boundaries. When compared with TW-FINCH [39], our ABD still shows better performance. Similarly, our method outperforms weakly supervised methods by a significant margin.

Moreover, to make a fair comparison with activity-level Hungarian methods, we assume the fixed ordering of actions. Under this situation, we achieve 37.7%/51.3% on F1/MoF metrics, which demonstrates that our method still achieves competitive results compared with the strong ASAL [30]. Exploiting complicated correlations can improve the performance, as reflected on ASAL. Nevertheless, the competitive performance achieved by our method on both datasets indicates detecting boundaries also plays an important role in action segmentation.

YouTube Instructional Videos. We summarize the performance of our method on YouTube Instructional Videos in Table 2. To make a fair comparison, we remove background frames from videos as previous approaches [24, 39, 40]. ABD significantly outperforms all unsupervised methods, with absolute gains of 10.5% on MoF over TW-FINCH [39] and 9.5% improvements on F1 over LSTM+AL [1].

YouTube Instructional Videos			
Unsupervised	Hungarian	F1	MoF
Mallow [40]	Activity	27.0	27.8
CTE† [24]	Activity	28.3	39.0
ASAL [30]	Activity	32.1	44.9
LSTM+AL [1]	Video	39.7	—
Equal Split [39]	Video	27.8	30.2
TW-FINCH [39]	Video	48.2	56.7
Our ABD	Video	49.2	67.2

Table 2. Comparison to the state-of-the-art under unsupervised learning on YouTube Instructions Videos.† CTE reports results for a background ratio of 75%.

Hollywood Extended				
Weakly Supervised		IoU		MoF
SCT [13]		17.7		—
CDFL [28]		19.5		40.6
MuCon [42]		—		41.6
Unsupervised	Hungarian	IoU	F1	MoF
Equal Split [39]	Video	24.6	—	39.6
TW-FINCH [39]	Video	35.0	45.7	55.0
Our ABD	Video	36.0	57.1	60.7

Table 3. Comparison to the state-of-the-art on Hollywood Extended.

Since most frames are background in this dataset, a higher MoF can be achieved by labeling most frames as background. However, this strong data bias has no significant effects on our method. When taking background frames into consideration, our method still gets 44.2% on MoF metric. The relatively slight drop in performance indicates our approach can deal with widely varying degrees of background videos.

Hollywood Extended. As shown in Table 3, our method achieves a significant leap compared with existing approaches. In particular, our ABD obtains improvements of 5.7% on MoF compared with TW-FINCH [39]. Similarly, our method outperforms the weakly supervised method MuCon [42] with 19.1% gains on the MoF metric.

50Salads. The performances by different methods in terms of MoF are demonstrated in Table 4. We evaluate the performance with respect to two action granularity levels as previous methods. The *eval* action granularity has 12 action classes such as *add oil*, *cut*, *place* and *mix_dressing*. We also evaluate the performance on the *mid* action granularity with 19 action classes, which differentiates actions such as *cut_tomato* and *cut_cheese*.

Experimental results show that our method achieves

50Salads			
Weakly supervised		*eval*	*mid*
RNN-FC [36]		—	45.5
NN-Vit. [38]		—	49.4
CDFL [28]		—	54.7
Unsupervised	Hungarian		
VTE-UNET [45]	Activity	30.6	24.2
CTE [24]	Activity	35.5	30.2
ASAL [30]	Activity	39.2	34.4
Equal Split [39]	Video	47.4	33.1
LSTM+AL [1]	Video	60.6	—
TW-FINCH [39]	Video	71.1	66.5
Our ABD	Video	71.4	71.8

Table 4. Comparison to the state-of-the-art on 50Salads. We report MoF in this table.

71.8% on MoF on the *mid* granularity, 5.3% higher than the best unsupervised approach TW-FINCH [39] and 17.1% higher than the weakly supervised method CDFL [28]. The F1 scores of the ABD on the *eval* and *mid* granularities are 64.2% and 59.2% respectively.

Generally, better performance is achieved on the *eval* granularity (that has fewer action classes) for other methods [24, 39, 45], which is different from ours. We argue that it is because our method mainly concerns the changing of action boundaries. For example, *cut_tomato*, *place_tomato_into_bowl* and *cut_cheese* are three continuous actions, and our method can successfully segment these three actions based on the boundaries detected. However, for the *eval* granularity, action *cut_tomato* and action *cut_cheese* are considered as the same class *cut*. Although it can be realized in the refinement process to some extent, the inherent distinctive properties with finer-granularity actions make it hard to group such actions.

Pairwise matching is conducted by clustering algorithms like TW-FINCH [39], which have a superior property for clustering these actions. However, it is difficult for them to pinpoint action boundaries since they focus on global correlations, which can be reflected on the *mid* granularity level experiment.

4.2. Effect of K and Features

Our method outputs results with K action classes, which considers the repetition of actions. Therefore, the setting of K used in our paper is (a) the average number of action classes per video (*e.g.,* 5 used for Breakfast). We also notice that there are several other settings on K, including (b) the average number of actions per video (7 for Breakfast); (c) total action classes for an activity (ASAL [30], CTE [24], etc.); (d) the maximum number of ground truth action classes for an activity, which has been discussed in [10]; (e) the average number of action classes per video for an activity (TW-FINCH [39]). We perform experiments on different settings as shown in Table 5. For activity-based settings, we have $K_{(c)} \geq K_{(d)} \geq K_{(e)}$. Our method still has good performance when setting K as (c), which is the most general one. It shows that our method has wide tolerance intervals for K.

	$K_{(a)}$	$K_{(b)}$	$K_{(c)}$	$K_{(d)}$	$K_{(e)}$	FV	I3D
F1	52.3	48.1	51.3	51.8	54.7	52.3	54.4
MoF	64.0	61.4	63.2	63.5	64.2	64.0	65.4

Table 5. Effect of K and features on Breakfast.

Moreover, besides Fisher Vectors (FV), we also perform experiments on CNN features (i.e., I3D [7]), and results are shown in the last two columns of Table 5. It demonstrates that better performance can be achieved when using a stronger feature extractor.

4.3. Gaussian Filter vs. Average Filter

There are multiple choices for the filters used for feature smoothing (Equation (2)). In this section, we compare the performance between Gaussian and average filters. As the video length and the number of actions vary enormously across datasets, we set the kernel size $(2k+1) = \alpha \cdot (N/K)$ and we will discuss α later. Note that K is fixed across all videos in the same dataset. Our method without smoothing is used as the baseline. Experimental results are demonstrated in the first three rows of Table 6.

We observe that the methods with either of the two smoothing filters outperform the baseline by a significant margin, which indicates the effectiveness of feature smoothing. Considering both Gaussian filter and average filter have consistent performance in our experiments, we utilize the average smoothing in all experiments due to its efficiency.

4.4. Effect of NMS

As declared in Section 3.2, the NMS algorithm is used to suppress redundant boundaries. We compare the performance by our method with and without NMS, which is listed in Table 6 (the 3th and 4th rows). Specifically, we conduct NMS inside local windows and select the smallest points as candidate boundaries. The length of local windows L is also equal to $\alpha \cdot (N/K)$.

As shown in Table 6, better performance can always be achieved when applied with NMS. For example, our method with NMS achieves great improvements on the *mid* granularity of 50Salads, i.e., up to 8.4% F1 improvements and 11.4% MoF improvements respectively. Experimental results indicate that NMS can well reduce redundant boundaries, resulting in fewer over-segmentation errors and

	Filter	NMS	Refinement	Breakfast		YTI		HE		FS (*eval*)		FS (*mid*)	
				F1	MoF	F1	MoF	F1	MoF	F1	MoF	F1	MoF
1	✗	✓	✓	42.4	58.3	46.0	63.7	56.1	59.5	60.3	64.0	55.3	59.0
2	Gaussian	✓	✓	51.6	63.5	49.1	66.6	56.7	60.2	63.4	70.8	**59.8**	**72.4**
3	Average	✓	✓	**52.3**	64.0	**49.2**	**67.2**	**57.1**	**60.7**	**64.2**	**71.4**	59.2	71.8
4	Average	✗	✓	41.7	57.3	42.8	60.1	55.6	56.3	53.2	55.0	50.8	60.4
5	Average	✓	✗	49.2	57.8	43.2	60.8	55.4	57.4	50.7	54.7	59.1	71.2
6	Average	✓	temporal	51.8	**64.2**	47.8	66.4	57.0	60.5	57.4	60.1	59.6	72.2

Table 6. The ablation experiments of our method on four datasets. YTI, HE and FS are the abbreviations for YouTube Instructional Videos, Hollywood Extended and 50Salads respectively. The metrics MoF and F1 are reported.

higher segmentation accuracy. In addition, NMS can significantly reduce the inference time as there are fewer segments for refinement.

4.5. Effect of Refinement

In Section 3.3, we introduce a clustering algorithm to refine initial candidates. The performance by our method with and without refinement is also compared on four datasets. Refer to the 3rd and 5th rows in Table 6 for quantitative comparison. In addition, as the temporal information plays an important role in the frame-wise clustering approach [39], we also discuss the case concerning this information (the last row in Table 6).

As discussed before, our ABD segments actions based on detecting boundaries, and the refinement is used to cluster segments that belong to the same action class. It is observed that when applied with the clustering algorithm, performance improvements can be achieved in all datasets, which indicates that split segments can be correctly clustered by this refinement algorithm. For example, a considerable boost is achieved by the refinement on the *eval* level of 50Salads, which agrees with our analysis that this clustering is capable of merging fine-granularity actions.

When considering the temporal information, negligible improvements (no more than 1%) are achieved on Breakfast and 50Salads (*mid*), nevertheless, a slight decline in performance occurs on other datasets. To better understand this problem, we summarise how many times each action occurs on average in Table 7. We find that the temporal information would contribute when activities occur roughly once in videos. Under this circumstance, there are fewer situations that actions belonging to the same class are temporally distant. However, when an action occurs multiple times, temporal information would have a negative effect as it prioritizes adjacent segments.

4.6. Effect of α

The filter size $(2k+1)$ and local window length of NMS L are all set to $\alpha \cdot (N/K)$, where $\alpha \in (0, 1)$. We compare

Dataset	BF	YTI	HE	FS (*eval*)	FS (*mid*)
Number	1.3	2.1	2.0	1.6	1.1

Table 7. The average number of each action occurs.

α	0.1	0.2	0.3	0.4	0.6	0.8	1
F1	43.4	50.2	53.2	52.3	52.5	51.5	49.1
MoF	59.1	63.4	63.7	64.0	61.1	57.0	55.9
Time	0.2	0.05	0.03	0.02	0.02	0.03	0.04

Table 8. The effect of α. The time is measured as the average inference time (seconds) for one video.

	YTI		HE	
Method	F1	MoF	F1	MoF
TW-FINCH [39]	48.2	56.7	—	55.0
Online ABD	48.6	60.3	52.3	59.2
Offline ABD	49.2	67.2	57.1	60.7

Table 9. Performance comparison of online and offline ABD on different datasets.

segmentation accuracy (F1 and MoF) and running time by varying values of α on Breakfast and the results are shown in Table 8. Inferior accuracy is achieved when α is too small or large, arising from redundant or overly-reduced boundaries. Experimental results exhibit promising performance for a broad wide range $\alpha \in (0.2, 0.8)$, indicating that the proposed ABD is not sensitive to well-adjusted hyper-parameters. Moreover, fast inference speed can be achieved by keeping a balance between feature smoothing and refinement.

4.7. Performance of Online ABD

Without knowing prior knowledge K, the online ABD method could segment actions based on the video stream-

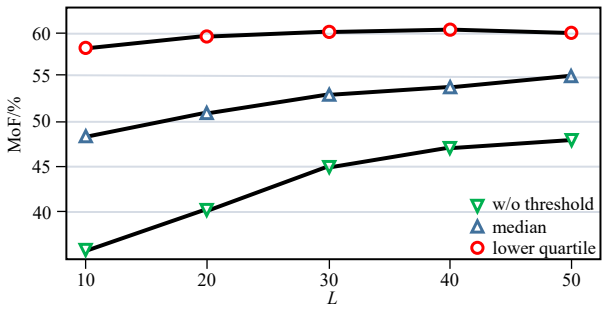


Figure 3. Online action segmentation performance in terms of MoF with varying L on YouTube Instructional Videos. These three curves represent different settings on the threshold.

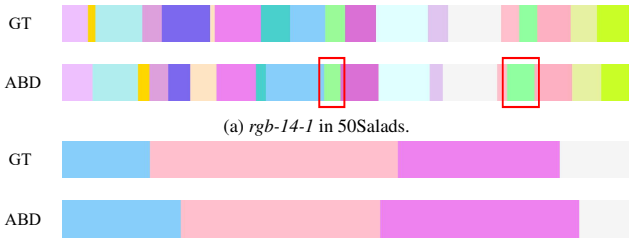
ing before the current timestamp in real-time. We compare the performance of online ABD and offline ABD on Hollywood Extended and YouTube Instructional Videos datasets. For evaluation, we use the prediction averaged over all the frames. The results are illustrated in Table 9.

We observe that online ABD achieves comparable performance compared to offline ABD, and outperforms the offline model TW-FINCH [39]. In addition, we also explore the effect of different values of window size L for NMS. Note that the filter size $(k+1)$ is set equal to L for similarity. Experimental results shown in Figure 3 are in accord with our analysis in Section 4.6, which demonstrates the robustness of the proposed method since the hyper-parameter values do not have a significant influence on online ABD. Moreover, we show three different strategies for threshold, including no threshold, median, and lower quartile. The gaps between three curves indicate that pseudo boundaries within actions can be greatly reduced by setting the threshold as the lower quartile of the similarity before time t.

4.8. Run-Time Comparison and Qualitative Results

The comparison of run-time between our ABD and other approaches is demonstrated in Table 10. All experiments are conducted on Breakfast split 1, and the inference time is reported for a single video. Every video has around 2,000 frames. The time used for feature extraction is not included. We can see that our method avoids hours of model training on GPUs. Compared with TW-FINCH [39] for which training is either not required, our approach gets a $8\times$ faster inference. Thanks to the fast inference speed and no training required, our method can be applied to practical applications when plugged with an off-the-shelf feature extractor.

Qualitative results by our method are illustrated in Figure 4. One can observe that the offline ABD successfully pinpoints action boundaries, yielding accurate segmentation results. Especially, as circled in the red rectangles in Figure 4a, our method can accurately cluster actions when they are temporally distant.



(a) *rgb-14-1* in 50Salads.

(b) *P12_webcam01_P12_cereals* in Breakfast.

Figure 4. Qualitative results of offline ABD on 50Salads and Breakfast. Each row demonstrates the segmentation result on the entire video and different colors represent different actions.

Supervision	Method	Training (hours)	Inference (seconds)
Weakly Sup.	CDFL‡ [28]	66.73	62.37
	MuCon-full‡ [42]	4.57	3.03
Unsup.	CTE [24]	—	217.94
	TW-FINCH [39]	✗	0.16
	Our ABD (offline)	✗	0.02
	Our ABD (online)	✗	0.008

Table 10. Comparison of training and inference time. The training time is measured for training on split 1 on Breakfast and the inference time is measured as the average inference time for a single video. ✗ indicates no training is needed. (‡ obtained from [42]).

5. Conclusion

We have proposed an efficient unsupervised action segmentation method by detecting action boundaries. Considering the properties of internal consistency within actions and external discrepancy across actions, we try to generate temporal action boundary proposals using frame-wise similarity in a bottom-up fashion. We first calculate the similarity between smoothed features. Initial boundary candidates could be obtained by detecting the change points along the similarity curve. Then a clustering algorithm is executed to refine candidates. Moreover, we further extend our method to the online setting, which enables real-time action segmentation. Evaluation on several challenging datasets demonstrates the effectiveness of our method.

Limitation. In this paper, to keep the overall approach as simple as possible, we set the kernel size of the smooth filter to a fixed value or proportional to the average action length, which would dilute the features of short actions. Also, applying the NMS technique could cause errors when two action boundaries are too close. Although experimental results show that our method has wide tolerance intervals for these parameters, more advanced strategies which enable variable lengths could be explored in the future work.

Acknowledgement. This work was supported by NSFC under Grant 62031023 and Grant 61801396.

References

[1] Sathyanarayanan N. Aakur and Sudeep Sarkar. A perceptual prediction framework for self supervised event segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019. 2, 4, 5, 6

[2] Ryan Prescott Adams and David JC MacKay. Bayesian online changepoint detection. *arXiv preprint arXiv:0710.3742*, 2007. 2

[3] Jean-Baptiste Alayrac, Piotr Bojanowski, Nishant Agrawal, Josef Sivic, Ivan Laptev, and Simon Lacoste-Julien. Unsupervised learning from narrated instruction videos. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4575–4583, 2016. 4

[4] Lorenzo Baraldi, Costantino Grana, and Rita Cucchiara. A deep siamese network for scene detection in broadcast videos. In *Proceedings of the 23rd ACM international conference on Multimedia*, pages 1199–1202, 2015. 2

[5] Piotr Bojanowski, Rémi Lajugie, Francis Bach, Ivan Laptev, Jean Ponce, Cordelia Schmid, and Josef Sivic. Weakly supervised action labeling in videos under ordering constraints. In *European Conference on Computer Vision*, pages 628–643. Springer, 2014. 2, 4

[6] John Canny. A computational approach to edge detection. *IEEE Transactions on pattern analysis and machine intelligence*, (6):679–698, 1986. 1, 3

[7] Joao Carreira and Andrew Zisserman. Quo vadis, action recognition? a new model and the kinetics dataset. In *proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6299–6308, 2017. 6

[8] Varun Chandola, Arindam Banerjee, and Vipin Kumar. Anomaly detection: A survey. *ACM computing surveys (CSUR)*, 41(3):1–58, 2009. 2

[9] R.T. Collins, A.J. Lipton, and T. Kanade. Introduction to the special section on video surveillance. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8):745–746, 2000. 1

[10] Guodong Ding and Angela Yao. Temporal action segmentation with high-level complex activity labels, 2021. 6

[11] Li Ding and Chenliang Xu. Weakly-supervised action segmentation with iterative soft boundary assignment. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6508–6516, 2018. 1, 2

[12] Yazan Abu Farha and Jurgen Gall. Ms-tcn: Multi-stage temporal convolutional network for action segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3575–3584, 2019. 1

[13] Mohsen Fayyaz and Jurgen Gall. Sct: Set constrained temporal transformer for set supervised action segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020. 2, 4, 5

[14] Kaylea Haynes, Paul Fearnhead, and Idris A Eckley. A computationally efficient nonparametric approach for changepoint detection. *Statistics and Computing*, 27(5):1293–1305, 2017. 2

[15] De-An Huang, Li Fei-Fei, and Juan Carlos Niebles. Connectionist temporal modeling for weakly supervised action labeling. In *European Conference on Computer Vision*, pages 137–153. Springer, 2016. 2

[16] Tariq Iqbal, Shen Li, Christopher Fourie, Bradley Hayes, and Julie A. Shah. Fast online segmentation of activities from partial trajectories. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 5019–5025, 2019. 2

[17] Yuchi Ishikawa, Seito Kasai, Yoshimitsu Aoki, and Hirokatsu Kataoka. Alleviating over-segmentation errors by detecting action boundaries. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 2322–2331, 2021. 1

[18] Eamonn Keogh, Selina Chu, David Hart, and Michael Pazzani. Segmenting time series: A survey and novel approach. In *Data mining in time series databases*, pages 1–21. World Scientific, 2004. 1, 2

[19] Eamonn J Keogh and Michael J Pazzani. Relevance feedback retrieval of time series data. In *Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*, pages 183–190, 1999. 2

[20] Eamonn J Keogh, Padhraic Smyth, et al. A probabilistic approach to fast pattern matching in time series databases. In *Kdd*, volume 1997, pages 24–30, 1997. 2

[21] Hilde Kuehne, Ali Arslan, and Thomas Serre. The language of actions: Recovering the syntax and semantics of goal-directed human activities. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 780–787, 2014. 1, 4

[22] Hilde Kuehne, Alexander Richard, and Juergen Gall. Weakly supervised learning of actions from transcripts. *Computer Vision and Image Understanding*, 163:78–89, 2017. 2

[23] Hilde Kuehne, Alexander Richard, and Juergen Gall. A hybrid rnn-hmm approach for weakly supervised temporal action segmentation. *IEEE transactions on pattern analysis and machine intelligence*, 42(4):765–779, 2018. 2

[24] Anna Kukleva, Hilde Kuehne, Fadime Sener, and Jurgen Gall. Unsupervised learning of action classes with continuous temporal embedding. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12066–12074, 2019. 2, 4, 5, 6, 8

[25] Ivan Laptev, Marcin Marszalek, Cordelia Schmid, and Benjamin Rozenfeld. Learning realistic human actions from movies. In *2008 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8, 2008. 4

[26] Colin Lea, Michael D Flynn, Rene Vidal, Austin Reiter, and Gregory D Hager. Temporal convolutional networks for action segmentation and detection. In *proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 156–165, 2017. 1

[27] Yong Jae Lee, Joydeep Ghosh, and Kristen Grauman. Discovering important people and objects for egocentric video summarization. In *2012 IEEE conference on computer vision and pattern recognition*, pages 1346–1353. IEEE, 2012. 1

[28] Jun Li, Peng Lei, and Sinisa Todorovic. Weakly supervised energy-based learning for action segmentation. In *Proceed-*

ings of the IEEE/CVF International Conference on Computer Vision, pages 6243–6251, 2019. 2, 5, 6, 8

[29] Jun Li and Sinisa Todorovic. Set-constrained viterbi for set-supervised action segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020. 2

[30] Jun Li and Sinisa Todorovic. Action shuffle alternating learning for unsupervised action segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12628–12636, 2021. 2, 4, 5, 6

[31] Jun Li and Sinisa Todorovic. Anchor-constrained viterbi for set-supervised action segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9806–9815, 2021. 2

[32] Tianwei Lin, Xu Zhao, Haisheng Su, Chongjing Wang, and Ming Yang. Bsn: Boundary sensitive network for temporal action proposal generation. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 3–19, 2018. 1

[33] Yu-Fei Ma, Xian-Sheng Hua, Lie Lu, and Hong-Jiang Zhang. A generic framework of user attention model and its application in video summarization. *IEEE transactions on multimedia*, 7(5):907–919, 2005. 1

[34] David S Matteson and Nicholas A James. A nonparametric approach for multiple change point analysis of multivariate data. *Journal of the American Statistical Association*, 109(505):334–345, 2014. 2

[35] Sanghyun Park, Sang-Wook Kim, and Wesley W Chu. Segment-based approach for subsequence searches in sequence databases. In *Proceedings of the 2001 ACM symposium on Applied computing*, pages 248–252, 2001. 2

[36] Alexander Richard, Hilde Kuehne, and Juergen Gall. Weakly supervised action learning with rnn based fine-to-coarse modeling. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017. 2, 6

[37] Alexander Richard, Hilde Kuehne, and Juergen Gall. Action sets: Weakly supervised action segmentation without ordering constraints. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 5987–5996, 2018. 2

[38] Alexander Richard, Hilde Kuehne, Ahsan Iqbal, and Juergen Gall. Neuralnetwork-viterbi: A framework for weakly supervised video learning. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 7386–7395, 2018. 2, 5, 6

[39] Saquib Sarfraz, Naila Murray, Vivek Sharma, Ali Diba, Luc Van Gool, and Rainer Stiefelhagen. Temporally-weighted hierarchical clustering for unsupervised action segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11225–11234, 2021. 2, 3, 4, 5, 6, 7, 8

[40] Fadime Sener and Angela Yao. Unsupervised learning and segmentation of complex activities from video. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018. 2, 4, 5

[41] Alan F Smeaton, Paul Over, and Aiden R Doherty. Video shot boundary detection: Seven years of trecvid activity. *Computer Vision and Image Understanding*, 114(4):411–418, 2010. 2

[42] Yaser Souri, Mohsen Fayyaz, Luca Minciullo, Gianpiero Francesca, and Juergen Gall. Fast weakly supervised action segmentation using mutual consistency. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2021. 2, 5, 8

[43] Sebastian Stein and Stephen J McKenna. Combining embedded accelerometers with computer vision for recognizing food preparation activities. In *Proceedings of the 2013 ACM international joint conference on Pervasive and ubiquitous computing*, pages 729–738, 2013. 1, 3, 4

[44] Gerrit JJ van den Burg and Christopher KI Williams. An evaluation of change point detection algorithms. *arXiv preprint arXiv:2003.06222*, 2020. 2

[45] Rosaura G VidalMata, Walter J Scheirer, Anna Kukleva, David Cox, and Hilde Kuehne. Joint visual-temporal embedding for unsupervised learning of actions in untrimmed sequences. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 1238–1247, 2021. 2, 4, 5, 6

[46] Sarvesh Vishwakarma and Anupam Agrawal. A survey on activity recognition and behavior understanding in video surveillance. *The Visual Computer*, 29(10):983–1009, 2013. 1

[47] Heng Wang and Cordelia Schmid. Action recognition with improved trajectories. In *Proceedings of the IEEE international conference on computer vision*, pages 3551–3558, 2013. 4

[48] Zhenzhi Wang, Ziteng Gao, Limin Wang, Zhifeng Li, and Gangshan Wu. Boundary-aware cascade networks for temporal action segmentation. In *European Conference on Computer Vision*, pages 34–51. Springer, 2020. 1