Action Tubelet Detector for Spatio-Temporal Action Localization

Vicky Kalogeiton[1,2] Philippe Weinzaepfel[3] Vittorio Ferrari[2] Cordelia Schmid[1]

Abstract

Current state-of-the-art approaches for spatio-temporal action localization rely on detections at the frame level that are then linked or tracked across time. In this paper, we leverage the temporal continuity of videos instead of operating at the frame level. We propose the **AC***tion* **T***ubelet detector (ACT-detector) that takes as input a sequence of frames and outputs tubelets,* i.e.*, sequences of bounding boxes with associated scores. The same way state-of-the-art object detectors rely on anchor boxes, our ACT-detector is based on anchor cuboids. We build upon the SSD framework [19]. Convolutional features are extracted for each frame, while scores and regressions are based on the temporal stacking of these features, thus exploiting information from a sequence. Our experimental results show that leveraging sequences of frames significantly improves detection performance over using individual frames. The gain of our tubelet detector can be explained by both more accurate scores and more precise localization. Our ACT-detector outperforms the state-of-the-art methods for frame-mAP and video-mAP on the J-HMDB [12] and UCF-101 [31] datasets, in particular at high overlap thresholds.*

1. Introduction

Action localization is one of the key elements to video understanding. It has been an active research topic for the past years due to various applications, *e.g.* video surveillance [10, 21] or video captioning [33, 37]. Action localization focuses both on classifying the actions present in a video and on localizing them in space and time. Action localization task faces significant challenges, *e.g.* intra-class variability, cluttered background, low quality video data, occlusion, changes in viewpoint. Recently, Convolutional Neural Networks (CNNs) have proven well adapted for action localization, as they provide robust representations of video frames. Indeed, most state-of-the-art action localization approaches [9, 23, 27, 30, 36] are based on CNN object detectors [19, 25] that detect human actions at the frame level. Then, they either link frame-level detections or track them over time to create spatio-temporal tubes. Although these action localization methods have achieved remarkable results [23, 27], they do not exploit the temporal continuity of videos as they treat the video frames as a set of independent images on which a detector is applied independently. Processing frames individually is not optimal, as distinguishing actions from a single frame can be ambiguous, *e.g. person sitting down* or *standing up* (Figure 1).

Figure 1. Understanding an action from a single frame can be ambiguous, *e.g. sitting down* or *standing up*; the action becomes clear when looking at a sequence of frames.

In this paper, we propose to surpass this limitation and treat a video as a sequence of frames. Modern object detectors for images, such as Faster R-CNN [25] and Single Shot MultiBox Detector (SSD) [19], proceed by classifying and regressing a set of anchor boxes to the ground-truth bounding box of the object. In this paper, we introduce a spatio-temporal tubelet extension of this design. Our Action Tubelet detector (ACT-detector) takes as input a short sequence of a fixed number of frames and outputs *tubelets*, *i.e.*, sequences of bounding boxes over time (Figure 2). Our method considers densely sampled anchors of cuboid shape with various sizes and aspect ratios. At test time, we generate for each anchor cuboid a score for a given action and regressed coordinates transforming it into a tubelet. Importantly, the score and regression are based on convolutional feature maps from all frames in the sequence. While the anchor cuboids have fixed spatial extent across time, the tubelets change size, location and aspect ratio over time, following the actors. Here we build upon the SSD framework, but the proposed tubelet extension is applicable to other detectors based on anchor boxes, such as Faster R-CNN.

[1]Univ. Grenoble Alpes, Inria, CNRS, Grenoble INP, LJK, 38000 Grenoble, France.
[2]University of Edinburgh
[3]Naver Labs Europe

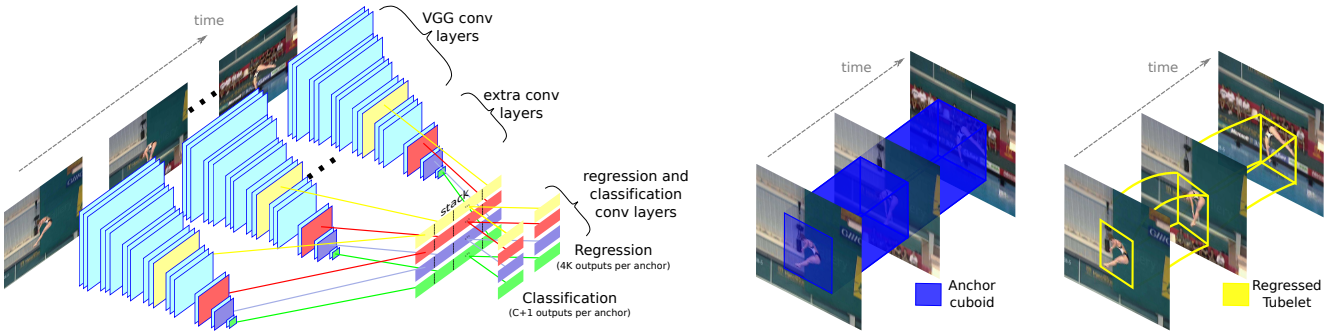

Figure 2. Overview of our ACT-detector. Given a sequence of frames, we extract convolutional features with weights shared between frames. We stack the features from subsequent frames to predict scores and regress coordinates for the anchor cuboids (middle figure, blue color). Depending on the size of the anchors, the features come from different convolutional layers (left figure, color coded: yellow, red, purple, green). As output, we obtain tubelets (right figure, yellow color).

Our experiments show that taking as input a sequence of frames improves: (a) action scoring, because the ambiguity between different actions reduces and (b) localization accuracy, because frames in a cuboid are regressed jointly and hence, they share information about the location of the actor in neighboring frames, see Figure 1. Our ACT-detector obtains state-of-the-art frame-mAP and video-mAP performance on the J-HMDB [12] and UCF-101 [31] datasets, in particular at high overlap thresholds.

In summary, we make the following contributions:

• We introduce the ACT-detector, an action tubelet detector that proceeds by scoring and regressing anchor cuboids.

• We demonstrate that anchor cuboids can handle moving actors for sequences up to around 10 frames.

• We provide an extensive analysis demonstrating the clear benefit of leveraging sequences of frames instead of operating at the frame level.

The code of our ACT-detector is available at http://thoth.inrialpes.fr/src/ACTdetector.

2. Related work

Almost all recent works [23, 27, 30, 36] for action localization build on CNN object detectors [19, 25]. In the following, we review recent CNN object detectors and then state-of-the-art action localization approaches.

Object detection with CNNs. Recent state-of-the-art object detectors [8, 19, 24, 25] are based on CNNs. R-CNN [8] casts the object detection task as a region-proposal classification problem. Faster R-CNN [25] extends this approach by generating bounding box proposals with a fully-convolutional Region Proposal Network (RPN). RPN considers a set of densely sampled anchor boxes, that are scored and regressed. Moreover, it shares convolutional features with proposal classification and regression branches. These branches operate on fixed-size features obtained using a Region-of-Interest (RoI) pooling layer. In a similar spirit, YOLO [24] and SSD [19] also use a set of anchor boxes, which are directly classified and regressed without a RoI pooling layer. In YOLO, all scores and regressions are computed from the last convolutional feature maps, whereas SSD adapts the features to the size of the boxes. Features for predicting small-sized boxes come from early layers, and features for big boxes come from the latter layers, with larger receptive fields. All these object detectors rely on anchor boxes. In our work, we extend them to anchor cuboids leading to significant improvement for action localization.

Action localization. Initial approaches for spatio-temporal action localization are extensions of the sliding window scheme [2, 16], requiring strong assumptions such as a cuboid shape, *i.e.*, a fixed spatial extent of the actor across frames. Other methods extend object proposals to videos. Hundreds of action proposals are extracted per video given low-level cues, such as super-voxels [11, 22] or dense trajectories [3, 7, 20]. They then cast action localization as a proposal classification problem. More recently, some approaches [17, 34, 38] rely on an actionness measure [4], *i.e.*, a pixel-wise probability of containing any action. To estimate actionness, they use low-level cues such as optical flow [38], CNNs with a two-stream fully-convolutional architecture [34] or recurrent neural networks [34]. They extract action tubes either by thresholding [17] the actionness score or by using a maximum set coverage formulation [38]. This, however, outputs only a rough localization of the action as it is based on noisy pixel-level maps.

Most recent approaches rely on object detectors trained to discriminate human action classes at the frame level. Gkioxari and Malik [9] extend the R-CNN framework to a two-stream variant [28], processing RGB and flow data separately. The resulting per-frame detections are then linked using dynamic programming with a cost function based on detection scores of the boxes and overlap between detections of consecutive frames. Weinzaepfel *et al.* [36] replace the linking algorithm by a tracking-by-detection method. More recently, two-stream Faster R-CNN was introduced by [23, 27]. Saha *et al.* [27] fuse the scores of both streams based on overlap between the appearance and the motion RPNs. Peng and Schmid [23] combine proposals

extracted from the two streams and then classify and regress them with fused RGB and multi-frame optical flow features. They also use multiple regions inside each action proposal and then link the detections across a video based on spatial overlap and classification score. Singh *et al*. [30] perform action localization in real-time using (a) the efficient SSD detector, (b) a fast method [14] to estimate the optical flow for the motion stream, and (c) an online linking algorithm. All these approaches rely on detections *at the frame level*. In contrast, we build our ACT-detector by taking as input sequences of frames and demonstrate improved action scores and location accuracy over frame-level detections.

3. ACtion Tubelet (ACT) detector

We introduce the **AC**tion **T**ubelet detector (ACT-detector), an action tubelet approach for action localization in videos. The ACT-detector takes as input a sequence of K frames $f_1, ..., f_K$ and outputs a list of spatio-temporal detections, each one being a *tubelet*, *i.e*., a sequence of bounding boxes, with one confidence score per action class. The idea of such an extension to videos could be applied on top of various state-of-the-art object detectors. Here, we apply our method on top of SSD, as it has lower runtime than other detectors, which makes it suitable for large video datasets. In this section, we first describe our fdafd4dfposed ACT-detector (Section 3.1), and then our full framework for video detection (Section 3.2). Finally, Section 3.3 describes our method for constructing action tubes.

3.1. ACT-detector

In this paper, we claim that action localization benefits from predicting tubelets taking as input a sequence of frames instead of operating at the frame level. Indeed, the appearance and even the motion may be ambiguous for a single frame. Considering more frames for predicting the scores reduces this ambiguity (Figure 1). Moreover, this allows to perform regression jointly over consecutive frames. Our ACT-detector builds upon SSD, see Figure 2 for an overview of the approach. In the following we review the SSD detector in details and then present our ACT-detector.

SSD detector. The SSD detector (Single Shot MultiBox Detector) [19] performs object detection by considering a set of anchor boxes at different positions, scales and aspect ratios. Each of them is (a) scored for each object class and for a background class, and (b) regressed to better fit the object extent. SSD uses a fully convolutional architecture, without any object proposal step, enabling fast computation. Classification and regression are performed using different convolutional layers depending on the scale of the anchor box. Note that the receptive field of a neuron used to predict the classification scores and the regression of a given anchor box remains significantly larger than the box.

ACT-detector. Given a sequence of K frames, the ACT-detector computes convolutional features for each one. The weights of these convolutional features are shared among all input frames. We extend the anchor boxes of SSD to anchor cuboids by assuming that the spatial extent is fixed over time along the K frames. We then stack the corresponding convolutional features from each of K frames (Figure 2). The stacked features are the input of two convolutional layers, one for scoring action classes and one for regressing the anchor cuboids. For instance, when considering an anchor cuboid for which the prediction is based on the 'red' feature maps of Figure 2, the classification and regression are performed with convolutional layers that take as input the 'red' stacked feature maps from the K frames. The classification layer outputs for each anchor cuboid $C+1$ scores: one per action class plus one for the background. This means that the tubelet classification is done based on the sequence of frames. The regression outputs $4 \times K$ coordinates (4 for each of the K frames) for each anchor cuboid. Note that although all boxes in a tubelet are regressed jointly, they result in a different regression for each frame.

The initial anchor cuboids have a fixed spatial extent over time. In Section 4.2 we show experimentally that such anchor cuboids can handle moving actors for short sequences of frames. Note that the receptive field of the neurons used to score and regress an anchor cuboid is larger than its spatial extent. This allows us to base the prediction also on the context around the cuboid, *i.e*., with knowledge for actors that may move outside the cuboid. Moreover, the regression significantly deforms the cuboid shape. Even though anchor cuboids have fixed spatial extent, the tubelets obtained after regressing the $4 \times K$ coordinates do not. We display two examples in Figure 3 with the anchor cuboid (cyan boxes) and the resulting regressed tubelet (yellow boxes). Note how the regression outputs an accurate localization despite the change in aspect ratio of the action boxes across time.

Training loss. For training, we consider only sequences of frames in which all frames contain the ground-truth action. As we want to learn action tubes, all positive and negative training data come from sequences in which actions occur. We exclude sequences in which the action starts or ends. Let $\mathcal{A}$ be the set of anchor cuboids. We denote by $\mathcal{P}$ the set of anchor cuboids for which at least one ground-truth tubelet has an overlap over 0.5, and by $\mathcal{N}$ the complementary set. Overlap between tubelets is measured by averaging the Intersection over Union (IoU) between boxes over K frames. Each anchor cuboid from $\mathcal{P}$ is assigned to ground-truth boxes with IoU over 0.5. More precisely, let $x_{ij}^y \in \{0,1\}$ be the binary variable whose value is 1 if and only if the anchor cuboid a_i is assigned to the ground-truth tubelet g_j of label y. The training loss $\mathcal{L}$ is defined as:

$$\mathcal{L} = \frac{1}{N}\left(\mathcal{L}_{\text{conf}} + \mathcal{L}_{\text{reg}}\right) , \tag{1}$$

with $N = \sum_{i,j,y} x_{ij}^y$ the number of positive assignments

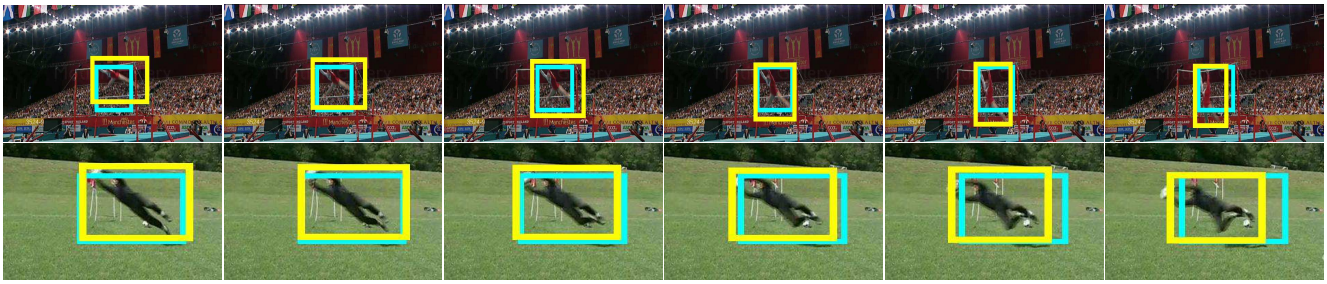

Figure 3. Examples of regressed tubelets (yellow) from cuboids (cyan) in our ACT-detector. Note the accurate localization of the tubelet, despite the fact that the aspect ratio of the cuboid is changing over time.

and $\mathcal{L}_{\text{conf}}$ (resp. $\mathcal{L}_{\text{reg}}$) the confidence (resp. regression) loss as defined below.

The confidence loss is defined using a softmax loss. Let $\hat{c}_i^y$ be the predicted confidence score (after softmax) of an anchor a_i for class y. The confidence loss is:

$$\mathcal{L}_{\text{conf}} = -\sum_{i\in\mathcal{P}} x_{ij}^y \log\left(\hat{c}_i^y\right) - \sum_{i\in\mathcal{N}} \log\left(\hat{c}_i^0\right) \quad . \tag{2}$$

The regression loss is defined using a Smooth-L1 loss between the predicted regression and the ground-truth target. We regress an offset for the center (x, y) of each box in the tubelet, as well as for the width w and the height h. The regression loss is averaged over K frames. More precisely, let $\hat{r}_i^{x_k}$ be the predicted regression for the x coordinate of anchor a_i at frame f_k and let g_j be the ground-truth target. The regression loss is defined as:

$$\mathcal{L}_{\text{reg}} = \frac{1}{K}\sum_{i\in\mathcal{P}}\sum_{c\in\{x,y,w,h\}} x_{ij}^y \sum_{k=1}^{K} \text{SmoothL1}\left(\hat{r}_i^{c_k} - \mathfrak{g}_{ij}^{c_k}\right) \quad ,$$

$$\text{with} \quad \mathfrak{g}_{ij}^{x_k} = \frac{g_j^{x_k} - a_i^{x_k}}{a_i^{w_k}} \qquad \mathfrak{g}_{ij}^{y_k} = \frac{g_j^{y_k} - a_i^{y_k}}{a_i^{h_k}} \quad , \tag{3}$$

$$\mathfrak{g}_{ij}^{w_k} = \log\left(\frac{g_j^{w_k}}{a_i^{w_k}}\right) \qquad \mathfrak{g}_{ij}^{h_k} = \log\left(\frac{g_j^{h_k}}{a_i^{h_k}}\right) \quad .$$

3.2. Two-stream ACT-detector

Following standard practice for action localization [23, 27, 36], we use a two-stream detector. We train an appearance detector, for which the input is a sequence of K consecutive RGB frames, and a motion detector, which takes as input the flow images [1] obtained following [9].

Each stream outputs a set of regressed tubelets with confidence scores, originating from the same set of anchor cuboids. For combining the two streams at test time we compare two approaches: union fusion and late fusion. For the union fusion [30], we consider the set union of the outputs from both streams: the tubelets from the RGB stream with their associated scores and the tubelets from the flow stream with their scores. For the late fusion [6], we average the scores from both streams for each anchor cuboid, as the set of anchors is the same for both streams. We keep the regressed tubelet from the RGB stream, as appearance is more relevant for regressing boxes, in particular for actions with limited motion. Our experiments show that late fusion outperforms the union fusion (Section 4.3).

3.3. From action tubelets to spatio-temporal tubes

For constructing action tubes, we build upon the frame linking algorithm of [30], as it is robust to missed detections and can generate tubes spanning different temporal extents of the video. We extend their algorithm from frame linking to *tubelet linking* and propose a *temporal smoothing* to build action tubes from the linked tubelets. The method is online and proceeds by iteratively adding tubelets to a set of links while processing the frames. In the following, t is a tubelet and L a link, *i.e.*, a sequence of tubelets.

Input tubelets. Given a video, we extract tubelets for each sequence of K frames. This means that consecutive tubelets overlap by $K-1$ frames. The computation of overlapping tubelets can be performed at an extremely low cost as the weights of the convolutional features are shared. We compute the convolutional features for each frame only once. For each sequence of frames, only the last layers that predict scores and regressions, given the stacked convolutional features (Figure 2), remain to be computed. For linking, we keep only the $N = 10$ highest scored tubelets for each class after non-maximum suppression (NMS) at a threshold 0.3 in each sequence of frames.

Overlap between a link and a tubelet. Our linking algorithm relies on an overlap measure $\text{ov}(L, t)$ between a link L and a tubelet t that temporally overlaps with the end of the link. We define the overlap between L and t as the overlap between the last tubelet of the link L and t. The overlap between two tubelets is defined as the average IoU between their boxes over overlapping frames.

Initialization. In the first frame, a new link is started for each of the N tubelets. At a given frame, new links start from tubelets that are not associated to any existing link.

Linking tubelets. Given a new frame f, we extend one by one in descending order of scores each of the existing links with one of the N tubelet candidates starting at this frame. The score of a link is defined as the average score of its tubelets. To extend a link L, we pick the tubelet candidate t

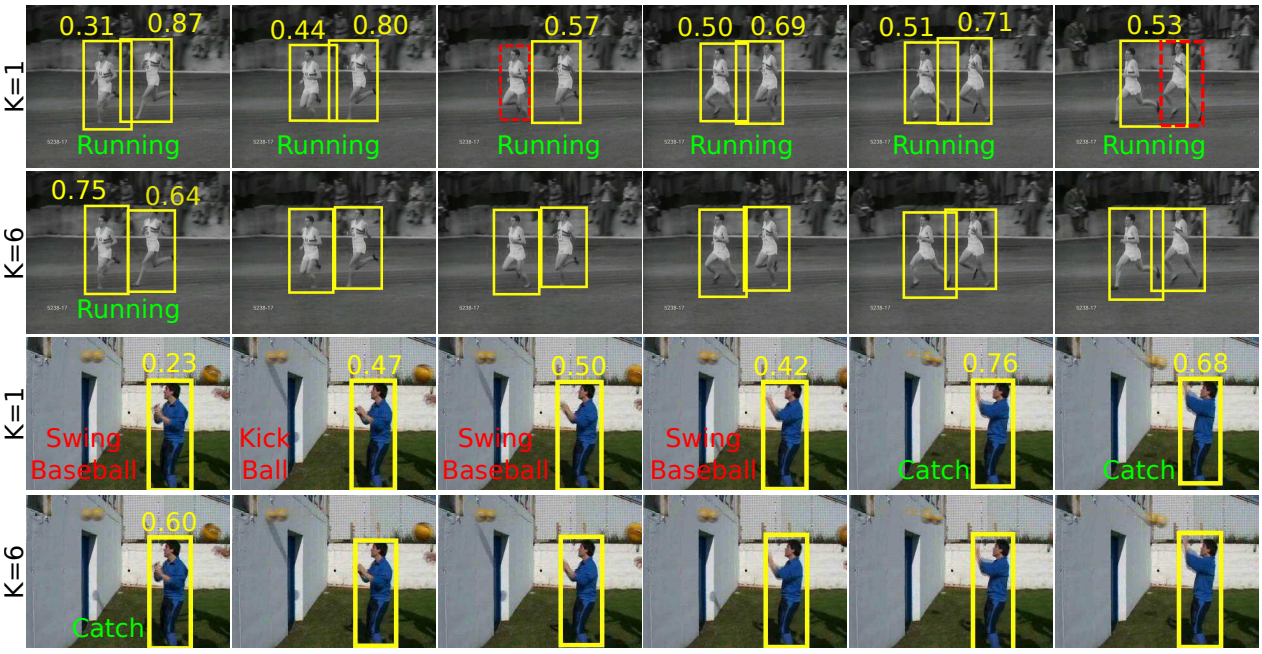

Figure 4. Examples when comparing per-frame ($K=1$) and tubelet detections ($K=6$). The yellow color represents the detections and their scores for the classes shown, the red color highlights errors due to missed detections (first row) or wrong labeling (third row) and the green color corresponds to correct labels. Our ACT-detector outputs one class label with one score per tubelet, we thus display it once.

that meets the following criteria: (i) is not already selected by another link, (ii) has the highest score, and (iii) verifies $\text{ov}(L,t) \geqslant \tau$, with τ a given threshold. In our experiments we use $\tau = 0.2$.

Termination. Links stop when these criteria are not met for more than $K-1$ consecutive frames.

Temporal smoothing: from tubelet links to action tubes. For each link L, we build an action tube, *i.e.*, a sequence of bounding boxes. The score of a tube is set to the score of the link, *i.e.*, the average score over the tubelets in the link. To set the bounding boxes, note that we have multiple box candidates per frame as the tubelets are overlapping. One can simply use the box of the highest scored tubelet. Instead, we propose a temporal smoothing strategy. For each frame, we average the box coordinates of tubelets that pass through that frame. This allows us to build smooth tubes.

Temporal detection. The initialization and termination steps result in tubes spanning different temporal extents of the video. Each tube determines, thus, the start and end in time of the action it covers. No further processing is required for temporal localization.

4. Experimental results

In this section we study the effectiveness of our ACT-detector. After presenting the datasets used in our experiments (Section 4.1), we provide an analysis of our ACT-detector: we validate anchor cuboids (Section 4.2), evaluate input modalities (RGB and flow) and their fusion (Section 4.3), and examine the impact of the length K of the sequence of frames (Section 4.4). We finally compare our method to the state of the art (Section 4.5). We also provide an error analysis in an extended version of this paper [13].

4.1. Datasets, metrics and implementation details

Datasets. The *UCF-Sports* dataset [26] contains 150 videos from 10 sports classes such as *diving* or *running*. The videos are trimmed to the action. We use the train/test split of [15].

The *J-HMDB* dataset [12] contains 928 videos with 21 actions, including *brush hair* and *climb stairs*. The videos are trimmed to the action. We report results averaged on the three splits defined in [12], unless stated otherwise.

The *UCF-101* dataset [31] contains spatio-temporal annotations for 24 sports classes in 3207 videos. The videos are not trimmed. Following [9, 23, 27, 36], we report results for the first split only.

Metrics. We use metrics at both frame and video level. Frame-level metrics allow us to compare the quality of the detections independently of the linking strategy. Metrics at the video level are the same as the ones at the frame level, replacing the Intersection-over-Union (IoU) between boxes by a spatio-temporal overlap between tubes, *i.e.*, an average across time of the per-frame IoU [27, 36]. To measure our performance at the frame level, we take into account the boxes originating from all tubelets that pass through the frame with their individual scores and perform NMS. In all cases, we only keep the detections with a score above 0.01.

We report *frame* and *video mean Average Precision (mAP)*. A detection is correct if its IoU with a ground-truth

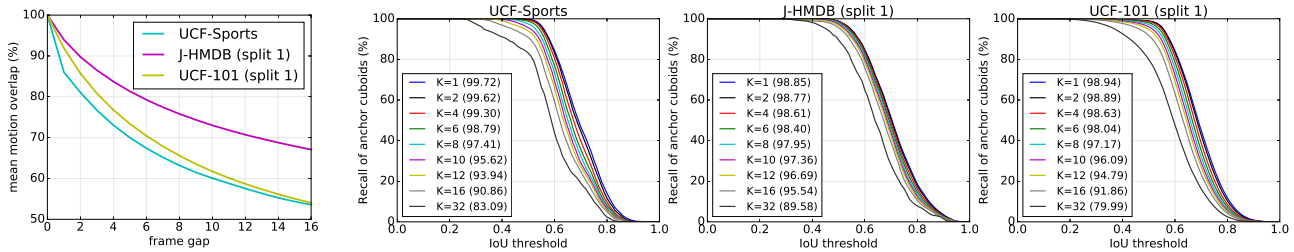

Figure 5. (a) *Motion overlap*: Mean motion overlap between a box in a ground-truth tube and its box n frames later for varying n. (b-d) Recall of the anchor cuboids for various IoU thresholds on the training set of three action localization datasets. The numbers in parenthesis indicate the recall at IoU$=0.5$.

box or tube is greater than 0.5 and its action label is correctly predicted [5]. For each class, we compute the average precision (AP) and report the average over all classes.

To evaluate the localization accuracy of the detections, we report *MABO* (Mean Average Best Overlap) [32]. We compute the IoU between each ground-truth box (or tube) and our detections. For each ground truth box (or tube), we keep the overlap of the best overlapping detection (BO) and, for each class, we average over all boxes (or tubes) (ABO). The mean is computed over all classes (MABO).

To evaluate the quality of the detections in terms of scoring, we also measure *classification accuracy*. In each frame, assuming that the ground-truth localization is known, we compute class scores for each ground-truth box by averaging the scores from the detected boxes or tubelets (after regression) whose overlap with the ground-truth box of this frame is greater than 0.7. We then assign the class having the highest score to each of these boxes and measure the ratio of boxes that are correctly classified.

Implementation details. We use VGG [29] with ImageNet pre-training for both appearance and motion streams [23, 35]. Our frames are resized to 300×300. We use the same hard negative mining strategy as SSD [19], *i.e.*, to avoid an unbalanced factor between positive and negative samples, only the hardest negatives up to a ratio of 3 negatives for 1 positive are kept in the loss. We perform data augmentation to the whole sequence of frames: photometric transformation, rescaling and cropping. Given the K parallel streams, the gradient of the shared convolutional layers is the sum over the K streams. We find that dividing the learning rate of the shared convolutional layers by K helps convergence, as it prevents large gradients.

4.2. Validation of anchor cuboids

This section demonstrates that an anchor cuboid *can* handle moving actions. We first measure how much the actors move in the training sets of the three action localization datasets by computing the *mean motion overlap*. For each box in a ground-truth tube, we measure its *motion overlap*: the overlap between this box and the ground-truth box n frames later for varying n. For each class, we compute the average motion overlap over all frames and we report the mean over all classes in Figure 5 (a). We observe that the motion overlap reduces as n increases, especially for UCF-Sports and UCF-101 for which the motion overlap for a gap of $n=10$ frames is around 60%. This implies that there is still overlap between the ground-truth boxes that are separated by $n=10$ frames. It also means that in many cases, this overlap is below 50% due to the motion of the actor.

In practice, we want to know if we have positive training anchor cuboids. Positive cuboids are the ones that have an overlap of at least 50% with a ground-truth tubelet; the overlap being the average IoU between boxes over the K frames in the sequence. Such cuboids are required for training the classifier and the regressor. Thus, we consider all possible training sequences and compute for each class the recall of the anchor cuboids with respect to the ground-truth tubelets, *i.e.*, the ratio of ground-truth tubelets for which at least one anchor cuboid has an overlap over 0.5. We report the mean recall over the classes for varying IoU thresholds for the three datasets in Figure 5 (b-d). For all datasets, the recall at IoU$=0.5$ remains $\geqslant 98\%$ up to $K=6$ and over 95% for $K=10$. This confirms that cuboid-shaped anchors can be used in case of moving actors. When increasing K, for instance to 32, the recall starts dropping significantly.

Given that sequences of up to $K=10$ frames result in high recall of the anchor cuboids, we examine in Sections 4.3 and 4.4 the performance of our tubelet detector for sequences of length ranging between $K=1$ and $K=10$.

4.3. Tubelet modality

In this section, we examine the impact of the RGB and flow modalities and their fusion on the performance of our ACT-detector. For all datasets, we examine the frame-mAP when using (i) only RGB data, (ii) only flow data, (iii) union of RGB + flow data [27], and (iv) late fusion of RGB + flow data for varying sequence length from 1 to 10 frames.

For all datasets and for all K, the RGB stream (blue line) outperforms the flow stream (red line), showing that appearance information is on average a more distinctive cue than motion (Figure 6). In all cases, using both modalities (green and black lines) improves the detection performance com-

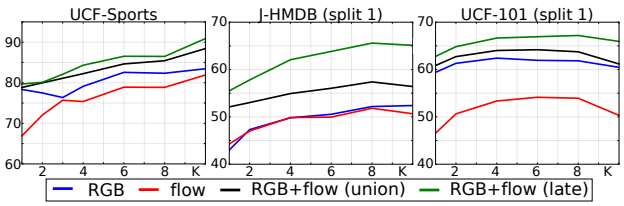

Figure 6. Frame-mAP of our ACT-detector on the three datasets when varying K for RGB data (blue line), flow (red line), union and late fusion of RGB + flow data (black and green lines, resp.).

pared to using only one. We observe that late fusion of the scores (green line) performs consistently better than union fusion (black line), with a gain between 1% and 4% in terms of frame-mAP. This can be explained by the fact that union fusion considers a bigger set of detections without taking into account the similarity between appearance and motion detections. Instead, the late fusion re-scores every detection by taking into account both RGB and flow scores. Given that late fusion delivers the best performance, we use it in the remainder of this paper.

4.4. Tubelet length

In this section, we examine the impact of K. We consider $K = 1$ as the baseline, and we report results for our method with $K = 2, 4, 6, 8, 10$. We quantify the impact of K by measuring (i) the localization accuracy (MABO), (ii) the classification accuracy, (iii) the detection performance (frame-mAP), and (iv) the motion speed of actors.

MABO. MABO allows us to examine the localization accuracy of the per-frame detections when varying K. Results are reported in Figure 7 (top). For all three datasets we observe that using sequences of frames ($K > 1$) leads to a significant improvement. In particular, MABO increases up to $K = 4$, and then remains almost constant up to $K = 8$ frames. For instance, MABO increases by 5% on UCF-Sports, 2% on J-HMDB and 5% on UCF-101 when using $K = 6$ instead of $K = 1$. This clearly demonstrates that performing detection at the sequence level results in more accurate localization, see Figure 3 for examples. Overall, we observe that $K = 6$ is one of the values for which MABO obtains excellent results for all datasets.

Classification accuracy. We report classification accuracy on the three action localization datasets in Figure 7 (bottom). Using sequences of frames ($K > 1$) improves the classification accuracy of the detections for all datasets. For UCF-Sports, the accuracy keeps increasing with K, while for J-HMDB it remains almost constant after $K = 6$. For UCF-101, the accuracy increases when moving from $K = 1$ to $K = 4$ and after $K = 8$ it starts decreasing. Overall, using up to $K = 10$ frames improves performance over $K = 1$. This shows that the tubelet scoring improves the classification accuracy of the detections. Again, $K = 6$ is one of the values which results in excellent results for all datasets.

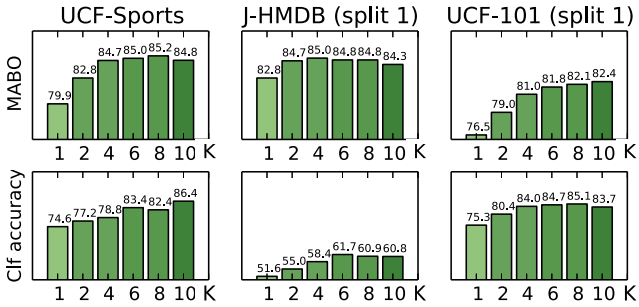

Figure 7. MABO (top) and classification accuracy (bottom) of our ACT-detector on the three datasets when varying K.

K	UCF-Sports			J-HMDB (split 1)			UCF-101 (split 1)		
	slow	medium	fast	slow	medium	fast	slow	medium	fast
$K=1$	79.5	84.0	68.1	61.2	55.5	49.0	69.6	73.5	67.3
$K=6$	85.5	89.7	76.8	69.8	66.9	58.0	75.4	78.5	70.7

Table 1. Frame-mAP for slow, medium and fast moving actors.

Frame-mAP. Figure 6 shows the frame-mAP when training the ACT-detector with varying K. On all three datasets, we observe a gain up to 10% when increasing the tubelet length up to $K = 6$ or 8 frames depending on the dataset, compared to the standard baseline of per-frame detection. This result highlights the benefit of performing detection at the sequence level. For J-HMDB and UCF-101, we also observe a performance drop for $K > 8$, because regressing from anchor cuboids is harder as (a) the required transformation is larger when the actor moves, and (b) there are more training parameters for less positive samples, given that the recall of anchor cuboids decreases (Section 4.2). The above results show that $K = 6$ gives overall good results. We use this value in the following sections.

Figure 4 shows some qualitative examples comparing the performance for $K = 1$ and $K = 6$. We observe that our tubelets lead to less missed detections and to more accurate localization compared to per-frame detection (first and second rows). Moreover, our ACT-detector reduces labeling mistakes when one frame is not enough to disambiguate between classes. For instance, in the last row we predict the correct label *catch*, whereas in the third row there is a big variance in the labels (*swing basketball, kick ball, catch*).

Handling moving actors. To validate that our ACT-detector can handle moving actors, we measure frame-mAP with respect to the speed of the actor. We group actors into three categories (slow, medium, fast) with 1/3 of the data in each category. Speed is computed using the IoU of an actor with its instances in ± 10 neighboring frames. Table 1 reports the frame-mAP at IoU = 0.5 for the three categories. For all datasets there is a clear gain between $K = 1$ and $K = 6$ for all speeds. In particular, for actors with fast motion the gain is +8% for UCF-Sports, +9% for J-HMDB, and +3% for UCF-101. This confirms that our tubelets can successfully handle large displacements. A potential explanation is the fact that the receptive fields are significantly larger than the the spatial extent of the anchor cuboid.

detector	method	UCF-Sports				J-HMDB (all splits)				UCF-101 (split 1)			
		0.2	0.5	0.75	0.5:0.95	0.2	0.5	0.75	0.5:0.95	0.2	0.5	0.75	0.5:0.95
actionness	[34]	-	-	-	-	-	56.4	-	-	-	-	-	-
R-CNN	[9]	-	75.8	-	-	-	53.3	-	-	-	-	-	-
	[36]	-	90.5	-	-	63.1	60.7	-	-	51.7	-	-	-
Faster R-CNN	[23] w/o MR	**94.8**	**94.8**	47.3	51.0	71.1	70.6	48.2	42.2	71.8	35.9	1.6	8.8
	[23] with MR	**94.8**	94.7	-	-	**74.3**	**73.1**	-	-	72.9	-	-	-
	[27]	-	-	-	-	72.6	71.5	43.3	40.0	66.7	35.9	7.9	14.4
SSD	[30]	-	-	-	-	73.8	72.0	44.5	41.6	73.5	46.3	15.0	20.4
	ours	92.7	92.7	**78.4**	**58.8**	**74.2**	**73.7**	**52.1**	**44.8**	**77.2**	**51.4**	**22.7**	**25.0**

Table 2. Comparison of video-mAP to the state of the art at various detection thresholds. The columns 0.5:0.95 correspond to the average video-mAP for thresholds with step 0.05 in this range. For [23], we report the results with and without their multi-region (+MR) approach.

detector	method	UCF-Sports	J-HMDB	UCF-101
actionness	[34]	-	39.9	-
R-CNN	[9]	68.1	36.2	-
	[36]	71.9	45.8	35.8
Faster R-CNN	[23] w/o MR	82.3	56.9	64.8
	[23] with MR	84.5	58.5	65.7
SSD	**ours**	**87.7**	**65.7**	**67.1**

Table 3. Comparison of frame-mAP to the state of the art. For [23], we report the results with and without their multi-region (+MR).

4.5. Comparison to the state of the art

We compare our ACT-detector to the state of the art. Note that our results reported in this section are obtained by stacking 5 consecutive flow images [23, 28] as input to the motion stream, instead of just 1 for each of the $K = 6$ input frames. This variant brings about $+1\%$ frame-mAP.

Frame-mAP. We report frame-mAP on the three datasets in Table 3. We compare our performance with late fusion of RGB+5flows when $K = 6$ to [9, 36], that use a two-stream R-CNN, and to [34], which is based on actionness. We also compare to Peng and Schmid [23] that build upon a two-stream Faster R-CNN with multiscale training and testing. We report results of [23] with and without their multi-region approach. The latter case can be seen as the baseline Faster R-CNN with multiscale training and testing for $K = 1$. Our ACT-detector (*i.e.*, with $K = 6$) brings a clear gain in frame-mAP, outperforming the state of the art on UCF-Sports, J-HMDB and UCF-101. We also observe that overall the performance of the baseline SSD ($K = 1$) is somewhat lower (by around 3 to 5%) than Faster R-CNN used by the state of the art [23], see Figure 6. SSD, however, is much faster than Faster R-CNN, and therefore more suitable for large video datasets.

Video-mAP. Table 2 reports the video-mAP results for our method and the state of the art at various IoU thresholds (0.2, 0.5, and 0.75). We also report results with the protocol 0.5:0.95 [18], which averages over multiple IoU thresholds, *i.e.*, over 10 IoU thresholds between 0.5 and 0.95 with a step of 0.05. At rather low IoU thresholds (0.2, 0.5) on UCF-Sports and J-HMDB the performance of our ACT-detector is comparable to the state-of-the-art methods that rely on Faster R-CNN [23, 27] or on SSD [30]. At higher overlap thresholds we significantly outperform them. For instance on UCF-Sports and J-HMDB at IoU $= 0.75$ we outperform [23] by 31% and 4%. In particular, our performance drops slower than the state of the art as the IoU threshold increases. This highlights the high localization accuracy of our tubelets and, therefore of our tubes. On UCF-101, we significantly outperform the state of the art at all overlap thresholds, with a larger gap at high thresholds. For instance, we outperform [30] by 5% at IoU$=0.5$, and by 7.5% at IoU $= 0.75$. To validate our tubelet linking strategy (Section 3.3), we experiment with an approach that transforms tubelets into individual boxes and links them with [23, 27]. We observe a consistent gain of 1% on all datasets. As a summary, our ACT-detector improves over the state of the art, especially at high thresholds.

Runtime. We compare our runtime using two streams (appearance and flow) to the frame-based SSD approach of Singh *et al*. [30] and to frame-based Faster R-CNN approaches [23, 27]. We report runtime on a single GPU without flow computation. Faster R-CNN based approaches [23, 27] run at 4fps and the SSD-based method [30] at 25-30fps. Our ACT-detector also runs at 25-30fps ($K = 6$). Computing tubelets has a low overhead, since the convolutional features are computed once per frame due to the parallel architecture with shared weights. The post-processing is extremely fast ($\sim$300fps) for all methods.

5. Conclusions

We introduced the ACT-detector, a tubelet detector that leverages the temporal continuity of video frames. It takes as input a sequence of frames and outputs *tubelets* instead of operating on single frames, as is the case with previous state-of-the-art methods [23, 27, 30]. Our method builds upon SSD and introduces anchor cuboids that are scored and regressed over sequences of frames. An extensive experimental analysis shows the benefits of our ACT-detector for both classification and localization. It achieves state-of-the-art results, in particular for high overlap thresholds.

Acknowledgments. This work was supported in part by the ERC grants ALLEGRO and VisCul, the MSR-Inria joint project, a Google research award, a Facebook gift, an Intel gift and an Amazon research award. We gratefully acknowledge the support of NVIDIA with the donation of GPUs.

References

[1] T. Brox, A. Bruhn, N. Papenberg, and J. Weickert. High accuracy optical flow estimation based on a theory for warping. In *ECCV*, 2004. 4

[2] L. Cao, Z. Liu, and T. S. Huang. Cross-dataset action detection. In *CVPR*, 2010. 2

[3] W. Chen and J. J. Corso. Action detection by implicit intentional motion clustering. In *ICCV*, 2015. 2

[4] W. Chen, C. Xiong, R. Xu, and J. Corso. Actionness ranking with lattice conditional ordinal random fields. In *CVPR*, 2014. 2

[5] M. Everingham, L. Van Gool, C. Williams, J. Winn, and A. Zisserman. The PASCAL Visual Object Classes Challenge 2007 (VOC2007) Results, 2007. 6

[6] C. Feichtenhofer, A. Pinz, and A. Zisserman. Convolutional two-stream network fusion for video action recognition. In *CVPR*, 2016. 4

[7] J. Gemert, M. Jain, E. Gati, C. G. Snoek, et al. APT: Action localization proposals from dense trajectories. In *BMVC*, 2015. 2

[8] R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *CVPR*, 2014. 2

[9] G. Gkioxari and J. Malik. Finding action tubes. In *CVPR*, 2015. 1, 2, 4, 5, 8

[10] W. Hu, T. Tan, L. Wang, and S. Maybank. A survey on visual surveillance of object motion and behaviors. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 2004. 1

[11] M. Jain, J. Van Gemert, H. Jégou, P. Bouthemy, and C. G. Snoek. Action localization with tubelets from motion. In *CVPR*, 2014. 2

[12] H. Jhuang, J. Gall, S. Zuffi, C. Schmid, and M. J. Black. Towards understanding action recognition. In *ICCV*, 2013. 1, 2, 5

[13] V. Kalogeiton, P. Weinzaepfel, V. Ferrari, and C. Schmid. Action tubelet detector for spatio-temporal action localization. *arXiv preprint arXiv:1705.01861*, 2017. 5

[14] T. Kroeger, R. Timofte, D. Dai, and L. Van Gool. Fast optical flow using dense inverse search. In *ECCV*, 2016. 3

[15] T. Lan, Y. Wang, and G. Mori. Discriminative figure-centric models for joint action localization and recognition. In *ICCV*, 2011. 5

[16] I. Laptev and P. Pérez. Retrieving actions in movies. In *ICCV*, 2007. 2

[17] Z. Li, E. Gavves, M. Jain, and C. G. Snoek. VideoLSTM convolves, attends and flows for action recognition. In *arXiv*, 2016. 2

[18] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick. Microsoft COCO: Common objects in context. In *ECCV*, 2014. 8

[19] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C. Y. Fu, and A. C. Berg. SSD: Single shot multibox detector. In *ECCV*, 2016. 1, 2, 3, 6

[20] M. Marian Puscas, E. Sangineto, D. Culibrk, and N. Sebe. Unsupervised tube extraction using transductive learning and dense trajectories. In *ICCV*, 2015. 2

[21] S. Oh, A. Hoogs, A. Perera, N. Cuntoor, C.-C. Chen, J. T. Lee, S. Mukherjee, J. Aggarwal, H. Lee, L. Davis, et al. A large-scale benchmark dataset for event recognition in surveillance video. In *CVPR*, 2011. 1

[22] D. Oneata, J. Revaud, J. Verbeek, and C. Schmid. Spatio-temporal object detection proposals. In *ECCV*, 2014. 2

[23] X. Peng and C. Schmid. Multi-region two-stream R-CNN for action detection. In *ECCV*, 2016. 1, 2, 4, 5, 6, 8

[24] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi. You only look once: Unified, real-time object detection. In *CVPR*, 2016. 2

[25] S. Ren, K. He, R. Girshick, and J. Sun. Faster R-CNN: Towards real-time object detection with region proposal networks. In *NIPS*, 2015. 1, 2

[26] M. D. Rodriguez, J. Ahmed, and M. Shah. Action MACH: A spatio-temporal maximum average correlation height filter for action recognition. In *CVPR*, 2008. 5

[27] S. Saha, G. Singh, M. Sapienza, P. H. Torr, and F. Cuzzolin. Deep learning for detecting multiple space-time action tubes in videos. In *BMVC*, 2016. 1, 2, 4, 5, 6, 8

[28] K. Simonyan and A. Zisserman. Two-stream convolutional networks for action recognition in videos. In *NIPS*, 2014. 2, 8

[29] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. In *ICLR*, 2015. 6

[30] G. Singh, S. Saha, M. Sapienza, P. Torr, and F. Cuzzolin. Online real time multiple spatiotemporal action localisation and prediction on a single platform. In *arXiv preprint arXiv:1611.08563*, 2017. 1, 2, 3, 4, 8

[31] K. Soomro, A. R. Zamir, and M. Shah. UCF101: A Dataset of 101 Human Actions Classes From Videos in The Wild. In *CRCV-TR-12-01*, 2012. 1, 2, 5

[32] J. R. Uijlings, K. E. van de Sande, T. Gevers, and A. W. Smeulders. Selective search for object recognition. *IJCV*, 2013. 6

[33] S. Venugopalan, M. Rohrbach, J. Donahue, R. Mooney, T. Darrell, and K. Saenko. Sequence to sequence—video to text. In *ICCV*, 2015. 1

[34] L. Wang, Y. Qiao, X. Tang, and L. Van Gool. Actionness estimation using hybrid fully convolutional networks. In *CVPR*, 2016. 2, 8

[35] L. Wang, Y. Xiong, Z. Wang, Y. Qiao, D. Lin, X. Tang, and L. Van Gool. Temporal segment networks: Towards good practices for deep action recognition. In *ECCV*, 2016. 6

[36] P. Weinzaepfel, Z. Harchaoui, and C. Schmid. Learning to track for spatio-temporal action localization. In *ICCV*, 2015. 1, 2, 4, 5, 8

[37] L. Yao, A. Torabi, K. Cho, N. Ballas, C. Pal, H. Larochelle, and A. Courville. Describing videos by exploiting temporal structure. In *ICCV*, 2015. 1

[38] G. Yu and J. Yuan. Fast action proposals for human action detection and search. In *CVPR*, 2015. 2