Unsupervised Tube Extraction using Transductive Learning and Dense Trajectories

Mihai Marian Puscas[1], Enver Sangineto[1], Dubravko Culibrk[1,2] and Nicu Sebe[1]
[1] Department of Information Engineering and Computer Science,
University of Trento, Italy
[2] Department of Industrial Engineering and Management,
University of Novi Sad, Serbia
mihaimarian.puscas,sangineto,Dubravko.Culibrk@unitn.it; sebe@disi.unitn.it

Abstract

We address the problem of automatic extraction of foreground objects from videos. The goal is to provide a method for unsupervised *collection of samples which can be further used for object detection training without any human intervention. We use the well known Selective Search approach to produce an initial still-image based segmentation of the video frames. This initial set of proposals is pruned and temporally extended using optical flow and transductive learning. Specifically, we propose to use Dense Trajectories in order to robustly match and track candidate boxes over different frames. The obtained box tracks are used to collect samples for* unsupervised *training of track-specific detectors. Finally, the detectors are run on the videos to extract the final tubes. The combination of appearance-based static "objectness" (Selective Search), motion information (Dense Trajectories) and transductive learning (detectors are forced to "overfit" on the unsupervised data used for training) makes the proposed approach extremely robust. We outperform state-of-the-art systems by a large margin on common benchmarks used for tube proposal evaluation.*

1. Introduction

The main motivation behind this work is to develop a method for automatic extraction of samples from videos which can then be used for unsupervised training of object detectors. In fact, the impressive progress recently obtained in object classification and object detection, e.g., using Convolutional Neural Networks [15, 11], heavily relies on huge datasets for training. While there exist huge supervised datasets for object classification (e.g., Imagenet), common datasets for object *detection* are much smaller, due to the additional effort which is necessary to produce bounding box (BB) annotations. However, there is a source of information which is not sufficiently taken into account, which can be used as a sort of surrogate of manual annotation, and this information is motion. In a given video, many objects of interest (persons, animals, vehicles, common tools, etc.) usually have a relative movement with respect to the camera viewpoint. This movement can be exploited to automatically segment such objects and provide BBs without human intervention.

Figure 1. Optical Flow between two consecutive frames can be used as a "voting" mechanism for matching Bonding Boxes. The blue lines are dense trajectories in common between the two boxes, while the red lines are trajectory starting from the first box but not included in the second.

One of the first attempts in this direction is the work of [19], in which Prest et al. extract *tubes* from a video clip which are used to train an object detector. A tube is defined as a sequence of BBs which (tightly) encloses the object of interest over different frames. This definition can be seen as a temporal extension of the "objectness", the property of an image window to likely contain an object instead of uninformative background. Objectness in still images has been studied in many works [24, 1, 5, 7]. For instance, Selective Search [24] is widely used in the object detection community in order to select a subset of windows in a still image which are then input to the classifier. However, it is worth noticing that the main goal of Selective Search and similar approaches is to speed up the *testing* phase, i.e., to replace the "old" sliding window approach with an initial selection

of boxes that most likely contain the objects. Hence, these techniques have as a primary goal to be conservative with respect to recall (BBs containing objects should not be discarded). Vice versa, our main goal in this paper is to achieve a sufficiently high precision, since the selected BBs need to contain few false positives to make the subsequent training of a detector feasible.

In this paper we do not directly deal with category-specific object detection, but we only focus on extending the objectness property from still images to videos (the outcome of our method being a set of BBs that can be used by common object detection methods for their training necessities). Other recent works which deal with automatic tube proposals address this extension of objectness to the temporal domain. However, most of the state-of-the-art approaches have the same limitation: they need a lot of tubes (usually hundreds or thousands per video clip) to achieve a sufficiently high recall [13, 12] which makes these methods reliable to speed up the testing phase but not sufficiently precise to allow for weakly supervised or unsupervised training. Using two common benchmarks (UCF Sports and YouTube Objects) we will show that we are able to achieve high recall with few tubes. For instance, in UCF Sports we achieve more than 30% relative improvement with respect to the state-of-the-art when using only one tube (Fig. 5).

These results have been achieved by combining different ideas. First, we use Selective Search in order to produce an initial set of candidate BBs. Then we propose to use Dense Trajectories [25, 26] in order to match BBs in different frames and to discard static BBs. This method allows us to collect initial tubes that we call *optical flow tubes* as they are based on the optical flow computed with Dense Trajectories. In order to avoid drifting (a common problem in all tracking algorithms), optical flow tubes are usually quite short and do not cover the whole video clip. For this reason we propose using the optical flow tubes in order to collect positive samples of the moving objects and train tube-specific detectors. We highlight that no class labels or other human-provided information is used for training. Conversely, tube-specific object detectors are learned in a *transductive* framework, i.e., we do not need these detectors to generalize to other videos, except the same video in which they have been trained. In fact, once trained, we run the detectors on the input videos in order to extract the final tubes (*detection tubes*). Using this strategy, we are able to extract BBs even in frames in which the object is static, while common tube-proposal approaches usually need movement in all the frames. To summarize, our contributions are:

- We use Dense Trajectories to robustly match BBs preselected by means of Selective Search.
- We use tube-specific, *class agnostic* detectors, trained in a transductive learning framework, to extract the final tubes.

The code for the proposed approach is available[1].

The rest of the paper is organized as follows. In Sec. 2 we briefly review the literature and in Sec. 3 we introduce some useful notation which will be used in the other sections. In Sec.s 4 and 5 we present our method. Experimental results are shown in Sec. 6 and we conclude in Sec. 7.

2. Related Work

In [19], Prest et al. extract tubes from a video clip exploiting homogeneous clusters of dense point tracks. The tubes are then used to learn a detector, together with video-level-based labels and based on the assumption that there is one dominant moving object per video. It is worth noticing that, in the approach we propose, the detectors are class-agnostic classifiers which are learned for every optical flow tube and then *used to extract the final tubes*. Conversely, in [19] the detectors are class-specific object detectors (fusing the segmentation phase with the final, unsupervised object classification phase). One drawback of this approach is that tubes are selected using inter-tube similarity, which is a fragile assumption when more than one moving object is present in the video clip and/or when a single object has a high variability of appearance.

Clustering dense tracks, obtained with optical flow, is a strategy adopted by many other authors. For instance in [4] point tracks are clustered using an affinity matrix based on the maximum translational difference between two tracks. Even if encouraging results can be obtained with this technique, articulated motion makes it hard to group tracks belonging to non-homogeneously moving objects. Optical flow is also used in [18], where objects are segmented using motion boundaries and then refined using a dynamic appearance model of the RGB foreground pixels. In [16] and in [2] optical flow and other appearance and saliency cues are used to extract coherent segments corresponding to moving objects.

In [13] the Selective Search [24] criteria for merging pixels in superpixels are extended into the time domain to obtain supervoxels. Supervoxels are used also in [12] with a hierarchical graph-based algorithm and in [17], where, instead of using heuristics, merging is performed using a classifier. In [10] motion boundaries are used in order to generate an initial set of moving object proposals, which is then ranked using a Convolutional Neural Network (CNN), trained using ground truth object BBs. It is worth noticing that both [17] and [10] are *supervised* methods, in which there is an important learning phase based on manually provided examples of ground truth objects and it is not clear what is the cross-dataset generalization capabilities of these

[1] *https://github.com/mihaipuscas/unsupervised-tube-extraction.git*

systems (when tested on datasets different from the ones used for training), while our approach is *completely unsupervised*. A similar limitation holds in [8, 23], where a CNN is trained in order to regress multiple boxes likely containing objects. The idea behind [8, 23] is that *static* objectness can be learned using ground truth BBs contained in large datasets (Pascal and ILSVRC 2012). However, a dataset bias does exist [14], since the cross-dataset experiments presented by the authors show a drastic drop of performance of the net when trained with Pascal and tested on ILSVRC 2012 and a minor drop vice-versa.

3. Static Objectness and Notation

Given a video with T frames, we apply Selective Search [24] to each frame F_t in order to extract the set of box candidates $B_t = \{b_1^t, ...b_n^t\}$ (we drop the superscript t when not necessary), and $b_i^t = (ymin_i, xmin_i, ymax_i, xmax_i)$.

Note that we rely on Selective Search to model *static* objectness. In other words, we do not manage pixel-level information, and we leverage on Selective Search for the pixel merging task in a single image. In fact this method is widely adopted and it has been proven to have a high *recall*: for instance, with $n = 2000$, the probability of an object to be highly overlapping with any $b_i^t \in B_t$ is around 0.9 [24]. All our efforts will be focused on pruning B_t ($1 \leq t \leq T$) using movement information in order to end up with a much smaller subset of boxes containing the moving objects of the video.

For simplicity, we also do not explicitly model the dynamics of the tracked boxes (which is difficult especially with "random" movements of biological "objects"). However, we use Intersection-over-Union (IoU) and Intersection-over-Min (IoM) in order to check spatial coherence between boxes of different frames and in the same frame:

$$IoU(b_1, b_2) = A(b_1 \cap b_2)/A(b_1 \cup b_2), \quad (1)$$

$$IoM(b_1, b_2) = A(b_1 \cap b_2)/\min\{A(b_1), A(b_2)\}, \quad (2)$$

where $A(b)$ is the area of b. Both IoU and IoM are widely adopted metrics in the object detection literature [6, 9, 11, 3] to assess spatial coherence (IoU) and/or to merge small BBs in a larger rectangle (e.g., see the Non-Maxima-Suppression algorithm, NMS, used in [6, 3] and based on IoM).

Finally, we use Dense Trajectories [26] to extract dense trajectories of moving points. In [26] the authors use optical flow in order to track points over different frames. They also improve over [25] by estimating the camera motion and deleting those trajectories whose movement is similar to the camera motion. The final trajectories cluster over the actual moving objects most of the times (but unfortunately camera motion compensation is not able to delete all the noisy trajectories in the background). Trajectories are continuously created and terminated over the video frames and are usually very short (max 15 frames [25]), thus there are no trajectories spanning the whole video. Given two consecutive frames F_t and F_{t+1}, we define the (camera motion compensated) optical flow between F_t and F_{t+1} as:

$$O(t, t+1) = \{o_1, ..., o_m\}, \quad (3)$$

where $o_j = (p_j, q_j)$ is a local translational offset belonging to one of the active trajectories between frames F_t and F_{t+1}, p_j is the starting point ($p_j \in F_t$) and q_j the ending point ($q_j \in F_{t+1}$).

For both Selective Search and Improved Trajectories we have used the publicly available code.

4. Optical Flow Tubes

The first step of our pipeline consists in matching boxes in F_t with boxes in F_{t+1} using optical flow information and spatial coherence. Given B_t and B_{t+1}, for each $b_i \in B_t$ and $b_j \in B_{t+1}$ we define:

$$OV(i, j) := IoU(b_i, bj) \geq 0.5, \quad (4)$$

where the threshold 0.5 is commonly adopted in object detection (e.g., in the Pascal and ImageNet detection tasks) to assess the spatial similarity of two BBs. Even if here the context is completely different (we use OV to prune BBs too far apart from each other in two different frames), we adopt the same threshold because it somehow guarantees that b_i and b_j can be matched only when the difference in scale and/or aspect ratio is not that large. This constrains a (possibly articulated) movement of the object between F_t and F_{t+1} to produce a small translational difference and a moderate deformation. If $n_1 = |B_t|$ and $n_2 = |B_{t+1}|$, then OV is an $n_1 \times n_2$ Boolean matrix.

For each b_i, b_j such that $OV(i, j) = true$, we compute the optical flow-based matching density between b_i and b_j, defined as:

$$D(i, j) := \frac{m_{ij}}{A(b_i) + A(b_j)}, \quad (5)$$

where m_{ij} is the number of optical flow offsets in $O(t, t+1)$ whose starting point is in b_i and ending point in b_j. The intuitive idea behind Eq. (5) is straightforward. The nominator represents the number of "votes" that can be accumulated in matching b_i and b_j, being each vote an element in $O(t, t+1)$. The denominator normalizes this number by the sum of the areas of the two BBs. This normalization is necessary because of noisy trajectories (e.g. trajectories laying on the background, despite camera motion compensation). In fact, maximizing m_{ij} without area normalization leads to

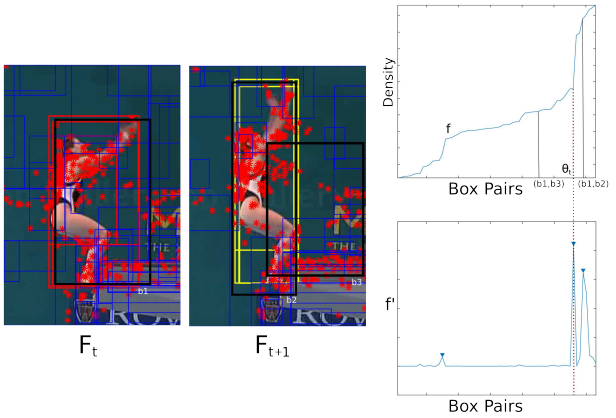

Figure 2. Adaptive threshold in matching density. Two consecutive frames with their initial set of BBs. Both optical flow and BBs cluster around the moving object (left). In turn, clusters correspond to plateaus in f, the sorted distribution of D (right-top). The smoothed gradient of f is used in order to detect peaks and to set the density threshold (right-bottom).

matching b_i with that b_j in B_{t+1} which is the *largest* possible, i.e., not a BB tight on the moving object but a BB usually including undesired background (e.g., see Fig. 1).

Using Eq. 5 we match b_i with b_j^* (and we write $M_t(b_i) = b_j^*$) such that:

$$b_j^* = \max_{b_j \in B_{t+1}} D(i,j), \tag{6}$$

subject to:

$$D(i,j) \geq \theta_t. \tag{7}$$

In Eq. (7) θ_t is a threshold which is used to reduce the risk of drifting in tracking a BB. Instead of using a fixed threshold, which is difficult to set, we adaptively compute θ_t for every pair of frames F_t and F_{t+1}, based on the observation that BBs produced by Selective Search usually cluster around true objects and dense trajectories tend to cluster around moving objects due to the camera motion compensation process. Looking at Fig. 2 (left), BBs b_1 and b_2, lying on the moving object, also belong to two corresponding clusters of BBs, respectively in frame F_t and F_{t+1} (depicted with red and yellow). The density value of those BB pairs belonging to these two clusters, computed using Eq. 5, is roughly constant for all the possible pairs. Conversely, matching b_1 with a background BB b_3, the corresponding density value is usually drastically different. In Fig. 2 (right-top) we plot the value of D, where the x-axis represents pairs of BBs sorted in ascending order with respect to D. Let $f()$ be the sorted distribution of D. Plateaus in $f()$ correspond to pairs of BBs belonging to clusters in F_t and F_{t+1}, and these clusters usually correspond to moving objects detected by Selective Search. We exploit this observation selecting θ_t as one of the steepest slopes in f. In Fig. 2 (right-bottom) we show the (smoothed) gradient of f, where peaks correspond to high variations in f before a plateau. We set θ_t to be the value of f corresponding to the median peak. Preliminary experiments with θ_t equal to the last peak (higher density) gave slightly lower results.

Using Eq.s (4)-(7) we can compute single frame matchings $M_t()$ for all the BBs in B_t, where $M_t(b_i)$ is not defined ($M_t(b_i) = \varnothing$) when there is no $b_j \in B_{t+1}$ such that (b_i, b_j) satisfies both constraints in Eq.s (4) and (7). We can then concatenate BBs in different frames forming a set of *chains* $CH = \{ch_1, ch_2, ...\}$, where a chain ch is computed starting from a given BB b_0 in frame t ($b_o \in B_t$) and:

$$ch = (b_0, b_1, ..., b_i, b_{i+1}, ..., b_{n_c}), \tag{8}$$

where:

$$b_{i+1} = M_{t+i}(b_i), \tag{9}$$

$$M_{t+n_c}(b_{n_c}) = \varnothing. \tag{10}$$

Chains are, on average, quite short ($E(n_c) \approx 6$ in our experiments). For this reason we further merge chains in *optical flow tubes*. We deal with the elements in CH as nodes in a graph, where an edge between two chains $ch_1, ch_2 \in CH$ is added when there is at least one frame in common between ch_1 and ch_2 such that the corresponding BBs in the two chains, $b_1 \in ch_1$ and $b_2 \in ch_2$, satisfy: $IoM(b_1, b_2) \geq 0.5$. Using IoM for measuring overlapping (instead of IoU) has the advantage that small BBs lying on subparts of the object of interest are clustered (e.g., [6, 3]) Hence, connected components of this graph correspond to chains with a sufficient spatial overlap in at least one frame. We compute an optical flow tube (ot) for each of these connected components:

$$ot = (r_0, r_1, ..., r_{n_o}), \tag{11}$$

where each $r_i \in ot$ is obtained by simply averaging the coordinates of those BBs $b_1, b_2, ...$ corresponding to the same frame F_t (i.e., $b_1, b_2, ... \in B_t$) and respectively belonging to the merged chains $ch_1, ch_2, ...$ (i.e., $b_1 \in ch_1$, $b_2 \in ch_2$, etc. ...).

The final optical flow tubes are relatively accurate. Still they only rely on two elements: the initial set of BBs provided by Selective Search and the matching pipeline described in this section, which is purely based on optical flow information. What is missing is a statistical model of the appearance of the tracked BBs, which can improve the result. We show in the next section how this model is computed.

5. Transductive learning

Let $OT = \{ot_1, ot_2, ...\}$ be the set of optical flow tubes computed as described in the previous section. For every

$ot \in OT$ we build a specialized classifier. We extract positive samples from the BBs in ot and negative samples from other BBs in the video frames in which ot is defined and we train a linear SVM. The classifier obtained is then run on the whole video to obtain a new tube, that we call a *detection tube*.

This is a special case of *transductive learning*, since the training samples are extracted, in an *unsupervised* manner, from the same video in which the classifier is tested. In other words, the aim of each classifier is to model the appearance of a tube and then use this model to refine the tube. We do not need that the classifier is able to generalize to other videos because it is only used for our tube extraction task.

The idea we propose is similar to *tracking by detection* approaches, and it is exploited, for instance, in [22]. The main difference of our approach with respect to [22] and other tracking by detection approaches is that our method is completely unsupervised, while in [22] a few positive BBs on the initial video frames need to be provided.

In more detail, given an optical flow tube $ot = (r_0, r_1, ..., r_{n_o})$, we include all of its BBs in the positive set P. Moreover, if $(F_{t_0}, ..., F_{t_{n_o}})$ is the sequence of frames in which ot is defined, we also include in P all those BBs which sufficiently overlap with one of the rectangles $r \in ot$ in one of these frames, using the IoU criterion in Eq. (4). The negative set starts with an initial set N_0 which is built including BBs b randomly extracted in the first frame F_{t_0} and such that $IoU(b, r_0) \leq 0.3$. The threshold 0.3 is widely adopted in the object detection literature for collecting negatives (e.g., see [11]). The negative set is iteratively pruned of the "easy negatives" and increased including new "hard negatives" by iteratively testing the current detector on the other frames while learning, following the well known hard negative mining approach proposed in [9]. Specifically, in a given frame $F_t \in (F_{t_0}, ..., F_{t_{n_o}})$, given P (which never changes) and N_t, we train a classifier $\mathbf{c}_t = [\mathbf{w}_t, a_t]$ by minimizing:

$$\mathbf{w}_t, a_t = \arg\min_{\mathbf{w},a} \sum_{r \in P} max(0, 1 - \mathbf{w}\phi(r) - a) + \sum_{r \in N_t} max(0, 1 + \mathbf{w}\phi(r) + a) + \lambda ||\mathbf{w}||_2^2, \quad (12)$$

where $\phi(r)$ is a feature representing the BB r. Different kinds of features can be used. For instance, HOG features are quite fast to be extracted from a rectangular patch of an image. In our experiments we used CNN features: $\phi(r)$ is the 4096-dimensional feature vector extracted from the last fully-connected layer (FC_7) of the ImageNet trained net described in [15]. Note that we do *not* perform fine tuning of the net's parameters. In principle we could use all the sets of positives P, extracted using all the optical flow tubes, in order to fine-tune the network before extracting our features. However, since the number of these tubes is small (on average, about 3 per video) and they are short, fine-tuning a network with millions of parameters [15] would probably lead to overfitting phenomena. Hence, we just use the net as a feature extractor, relying on the widely proven high discriminative skills of these features [21]. Following [11] we also add some padding around each r to include context. Finally, the value of λ, which controls the influence of the regularization term, is chosen according to [11]: $\lambda = 10^{-4}$ and the feature values are normalized as suggested in [11]. Following a consolidated object detection pipeline and adopting the parameters suggested in [9, 11] allows us to avoid the necessity of tuning the parameters of our classifiers. We believe that this is of primary importance for the success of an unsupervised method because it does not force one to collect data to tune the parameters when the method is applied to a new domain.

Once trained, $\mathbf{c}_t$ is tested on the BBs of the subsequent frames in which ot is defined, new hard negatives are added and training is repeated (Eq. (12)). We refer to [9] for details on the hard negative mining procedure. The final classifier is given by the parameters computed in the last frame of the tube: $\mathbf{c} = \mathbf{c}_{t_{n_o}}$.

5.1. Detection Tubes

Once collected a set of classifiers $C = \{\mathbf{c}^1, ..., \mathbf{c}^k\}$ from a given video, the final part of our pipeline concerns the extraction of detection tubes using these classifiers. Given a frame F_t and a classifier $\mathbf{c}^i = [\mathbf{w}^i, a^i] \in C$, the highest scoring detection BB d_t^i of $\mathbf{c}^i$ in F_t is obtained maximizing:

$$d_t^i = \arg\max_{b \in B_t} \mathbf{w}^i \phi(b) + a^i. \quad (13)$$

Note that we use all the BBs in B_t when "testing" the classifier. We then build a detection tube dt_i for each classifier $\mathbf{c}^i$ linking d_t^i over all the T frames of the video:

$$dt_i = (d_1^i, ..., d_t^i, ...d_T^i). \quad (14)$$

In this way we collect a set k detection tubes, one per classifier. Note that the cardinality of C, k, is *not* fixed a priori, and it depends on the number of optical flow tubes constructed in the previous phase (see Sec. 4). In our experiments, k is usually very small ($E(k) \approx 3$).

When many tubes are desired (e.g., to increase recall), we repeat training. More specifically, we *split* a detection tube dt in dt^1, dt^2 using the criteria of the first stage (Sec. 4). Given two consecutive detections d_t and d_{t+1} in dt, we split dt in $dt^1 = (d_1, ..., d_t)$ and $dt^2 = (d_{t+1}, ..., d_T)$ when: $IoU(d_t, d_{t+1}) < 0.5$ or $D(d_t, d_{t+1}) \geq \theta_t$, where, with a slight abuse of notation, $D(d_t, d_{t+1})$ is the match density defined in Eq. (5) and θ_t the adaptive threshold precomputed in the optical flow tube construction phase. After

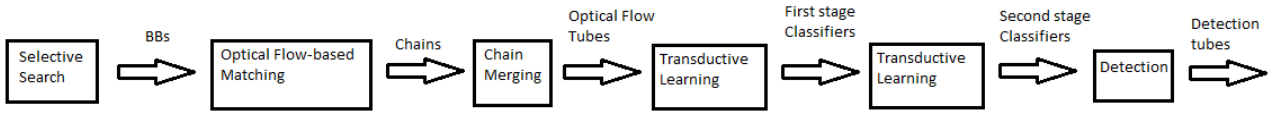

Figure 3. Flow chart of the proposed approach.

splitting the optical flow tubes, we use each tube to train a second set of classifiers C' repeating the procedure described in Sec. 5.

The final set of tubes for a given video is the set of the detection tubes obtained using all the detectors in C and C'. For a given video v, let $DT_v = \{dt_1, dt_2, ...\}$ be the set of all the detection tubes obtained using all the classifiers in C and C'. In Fig 3 we show the flow chart of the whole procedure.

6. Experiments

We evaluate our method using two common benchmarks and evaluation metrics for tube-proposal algorithms and we compare with the state-of-the-art approaches in this field.

6.1. Experimental Setup

Datasets We use for evaluation the UCF Sports dataset [20] and the YouTube Objects dataset [19]. UCF Sports is composed of 150 videos of 10 sports (e.g., diving, running, golf, kicking, etc.). For evaluation we used the ground truth annotation provided in [17]. Moreover, in order to allow a comparison with the results reported in [17], we adopted the same train/test split proposed in that article, where 100 videos are used for testing[2]. Note that in [17] the train split is used to train the proposed *supervised* method, while in our *unsupervised* approach we only used this "train" subset of 50 videos in the development stage to do all our design choices. We also do not have dataset-dependent parameters which need to be set (since all our parameter values are set using a consolidated object detection pipeline, see Sec.s 4-5), thus there is no training or parameter tuning phase in our approach, which makes the comparison with other supervised methods such as [17] disadvantageous for us since we do not exploit any dataset-specific information.

YouTube Objects is a large dataset composed of 1400 short shots obtained from videos collected on YouTube. As in the case of UCF Sports, many videos have large camera movement, illumination changes and cluttered backgrounds. However, the moving objects in this dataset usually occupy a larger portion of the frame, thus they are easier to detect. Differently from UCF Sports, in YouTube Objects there is only one annotated frame per shot but some frames are annotated with multiple objects. The dataset is split in a "train' and a "test" subset. We used the "test" shots to test our system (346 shots). Note that the "train" shots are usually easier, thus testing on the whole dataset would probably get higher accuracy results.

Metrics Following [17] we use two metrics: mBAO and CorLoc. Both metrics are based on the Best Average Overlap (BAO) of a set of tube proposals with ground truth objects. In UCF Sports dataset there is only one moving object annotated per video (but the dataset contains some videos with more than one moving object, being only the predominant object provided of ground truth annotations). Using this assumption, for a given video v and a set DT_v of tube proposals for v, BAO is defined as follows [17]:

$$BAO(v) = \max_{dt \in DT_v} \frac{1}{|T_v|} \sum_{t \in T_v} IoU(d_t, g_t), \qquad (15)$$

where $|T_v|$ is the set of frames of video v with ground truth annotation, d_t is the BB in tube proposal dt at frame t, and g_t is the ground truth at frame t. Note that in case of multiple annotated objects per video (YouTube Objects dataset), Eq. (15) is applied separately to each object using the same set of proposals T_v [17]. mBAO is the mean BAO across all the videos, while CorLoc is the fraction of videos for which the BAO is greater or equal to 0.5.

6.2. Comparison with State of the Art

UCF Sports. In Fig.s 4-5 we show the experimental results obtained on the "test" part of UCF Sports dataset (100 videos). The methods we compare to are: (1) the Spatio-Temporal Object Detection Proposals (STODP) proposed in [17], (2) The Graph-Based Hierarchical segmentation proposed in [12] and its variant (2) GBH-Flow presented in [17]. All the plotted results, except ours, have been obtained from [17].

Fig. 4 shows the mBAO plotted with respect to the number of average tube proposals per video and, similarly, Fig. 5 shows the CorLoc-based evaluation. In case of one tube per video, we obtain 0.374 mBAO and 0.37 CorLoc versus 0.3 and less than 0.3, respectively, of the state-of-the-art system on UCF Sports [17], with a relative CorLoc improvement of more than 30%. Once more we highlight that [17] is a supervised method, trained on the "train" split of UCF Sports, hence, most likely positively affected by a dataset

[2]The train/test split of the dataset and the annotations are provided at: *http://lear.inrialpes.fr/oneata/3Dproposals*

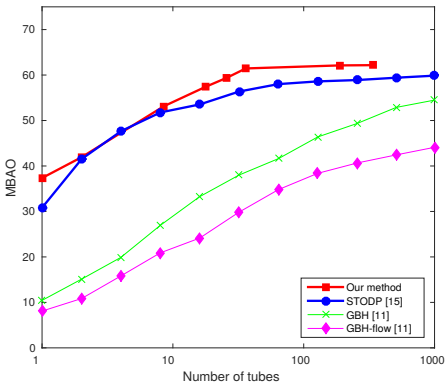

Figure 4. mBAO computed over the UCF Sport dataset.

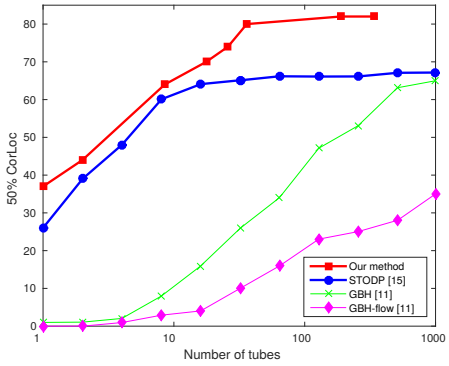

Figure 5. CorLoc computed over the UCF Sport dataset.

bias, while our method is completely unsupervised. We achieve the highest reported value of CorLoc (0.82) on UCF Sport with 188 tube proposals (quite close to 0.8, obtained with only 36 tubes). Moreover, our system achieves 0.7 CorLoc with only 18 tubes: a value of recall which is not achieved by the other methods even when using 1000 proposals.

YouTube Objects. Fig. 6 shows the results obtained with the YouTube Objects dataset. In this case we compare with two different parameter settings of STODP (we refer the reader to [17] for details), the unsupervised method proposed by Papazoglou et al. [18], the weakly supervised method of Prest et al. [19] and the result for the best tube among the proposals of the unsupervised method proposed by Brox and Malik [4]. All the plotted results, except ours, have been obtained from [17] (mBAO is not provided by the other authors).

As Fig. 6 clearly shows, we outperform all the competitors, both the supervised and the unsupervised methods. The only approach which achieves a CorLoc value better than our system is [18], which only outputs a single proposal per shot. However, we obtain a CorLoc higher than [18] with only 4 proposals. Compared with Oneata et al. [17], which obtained 0.461 when using 10 proposals, with the same number of tubes we obtain a CorLoc of 0.596, a relative improvement of 29%. Our largest value of CorLoc on this dataset is 0.927, obtained with 258 tubes, a recall much higher than any other published result.

6.3. Qualitative Results

In Fig. 7 we show some example results of our detection tubes using UCF Sports and Youtube Objects images. Most of the times our system is able to accurately detect the moving object even when it stops for a while (e.g., the dog, which is still with respect to the background, despite there is camera movement), unlike most of the state-of-the-art methods which require movement in all the frames.

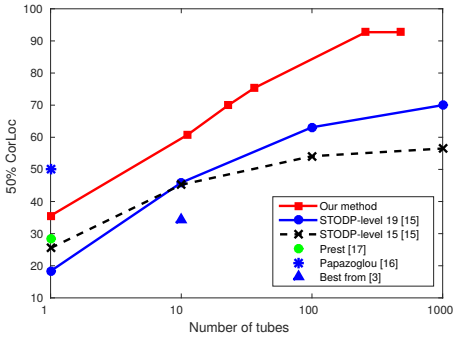

Figure 6. CorLoc computed over the YouTube Objects dataset.

In Fig. 8 we show some incorrectly detected tubes. In the middle row the misalignment between the ground truth and the detections is probably due to the difference in speed of the upper part and the lower part of the person, which produced detectors only for the fastest part (the upper body). In the first row our system is actually able to accurately track most of the moving persons but, unfortunately, the UCF Sport dataset contains annotation for only one object (person) per video, penalizing the extraction of multiple-objects.

7. Conclusions

We proposed a method for the extraction of tubes from videos which is based on a first pipeline in which optical flow obtained with Dense Trajectories is used for matching BBs and a second pipeline in which the initial tubes are used to collect positive training samples for training tube-specific detectors. The final tubes are given by the detections of the trained classifiers, used in a transductive framework. The method was evaluated on UCF Sports and YouTube Objects, showing state-of-the-art results.

Our approach is completely unsupervised and all the critical parameters and thresholds have been set by adopting the values commonly used in a consolidated object detection

Figure 7. Some examples of detection tubes. In each row we show a tube taken from a different video. 1-st and 2-nd row: UCF Sports dataset, 3-rd and 4-th row: YouTube Objects dataset. Red rectangles are BBs of the tube, while blue rectangles are ground truth annotations. Note that in YouTube Objects, only one frame is provided with ground truth (3-rd column).

Figure 8. Some examples of errors of the proposed method. 1-st and 2-nd row: UCF Sports dataset, 3-rd row: YouTube Objects dataset.

pipeline [9, 11, 6, 3] which makes the final system independent of specific datasets. Other important characteristics of the proposed technique are the possibility to detect the object in frames in which there is no movement (thanks to the detection-based approach) and the fact that we do not need to assume that only one moving object is present in a video clip.

Acknowledgements

We want to thank NVIDIA for their donation of a K40 GPU which was used to extract the CNN features. This work has been supported by the EU project xLiMe.

References

[1] B. Alexe, T. Deselaers, and V. Ferrari. Measuring the objectness of image windows. *IEEE Trans. on PAMI*, 34(11):2189–

2202, 2012. 1

[2] D. Banica, A. Agape, A. Ion, and C. Sminchisescu. Video object segmentation by salient segment chain composition. In *Computer Vision Workshops (ICCVW), IEEE International Conference on*, pages 283–290, 2013. 2

[3] L. Bourdev, S. Maji, T. Brox, and J. Malik. Detecting people using mutually consistent poselet activations. In *ECCV*, 2010. 3, 4, 8

[4] T. Brox and J. Malik. Object segmentation by long term analysis of point trajectories. In *ECCV*, pages 282–295, 2010. 2, 7

[5] J. Carreira and C. Sminchisescu. Constrained parametric min-cuts for automatic object segmentation. In *CVPR*, pages 3241–3248, 2010. 1

[6] P. Dollár, Z. Tu, P. Perona, and S. Belongie. Integral channel features. *BMVC*, 2009. 3, 4, 8

[7] I. Endres and D. Hoiem. Category independent object proposals. In *ECCV*, pages 575–588, 2010. 1

[8] D. Erhan, C. Szegedy, A. Toshev, and D. Anguelov. Scalable object detection using deep neural networks. In *CVPR*, pages 2155–2162, 2014. 3

[9] P. F. Felzenszwalb, R. B. Girshick, D. A. McAllester, and D. Ramanan. Object detection with discriminatively trained part-based models. *IEEE Trans. on PAMI*, 32(9):1627–1645, 2010. 3, 5, 8

[10] K. Fragkiadaki, P. A. Arbeláez, P. Felsen, and J. Malik. Spatio-temporal moving object proposals. *CoRR*, abs/1412.6504, 2014. 2

[11] R. B. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *CVPR*, pages 580–587, 2014. 1, 3, 5, 8

[12] M. Grundmann, V. Kwatra, M. Han, and I. Essa. Efficient hierarchical graph based video segmentation. In *CVPR*, 2010. 2, 6

[13] M. Jain, J. van Gemert, H. Jégou, P. Bouthemy, and C. G. M. Snoek. Action localization with tubelets from motion. In *CVPR*, pages 740–747, 2014. 2

[14] A. Khosla, T. Zhou, T. Malisiewicz, A. Efros, and A. Torralba. Undoing the damage of dataset bias. In *ECCV*, 2012. 3

[15] A. Krizhevsky, I. Sutskever, and G. E. Hinton. ImageNet classification with deep convolutional neural networks. In *NIPS*, pages 1106–1114, 2012. 1, 5

[16] Y. J. Lee, J. Kim, and K. Grauman. Key-segments for video object segmentation. In *ICCV*, pages 1995–2002, 2011. 2

[17] D. Oneata, J. Revaud, J. J. Verbeek, and C. Schmid. Spatio-temporal object detection proposals. In *ECCV*, 2014. 2, 6, 7

[18] A. Papazoglou and V. Ferrari. Fast object segmentation in unconstrained video. In *ICCV*, pages 1777–1784, 2013. 2, 7

[19] A. Prest, C. Leistner, J. Civera, C. Schmid, and V. Ferrari. Learning object class detectors from weakly annotated video. In *CVPR*, 2012. 1, 2, 6, 7

[20] M. D. Rodriguez, J. Ahmed, and M. Shah. Action MACH a spatio-temporal maximum average correlation height filter for action recognition. In *CVPR*, 2008. 6

[21] A. Sharif Razavian, H. Azizpour, J. Sullivan, and S. Carlsson. CNN features off-the-shelf: An astounding baseline for recognition. In *CVPR Workshops*, 2014. 5

[22] J. S. Supancic III and D. Ramanan. Self-paced learning for long-term tracking. In *CVPR*, pages 2379–2386, 2013. 5

[23] C. Szegedy, S. Reed, D. Erhan, and D. Anguelov. Scalable, high-quality object detection. *CoRR*, abs/1412.1441, 2014. 3

[24] J. R. R. Uijlings, K. E. A. van de Sande, T. Gevers, and A. W. M. Smeulders. Selective search for object recognition. *International Journal of Computer Vision*, 104(2):154–171, 2013. 1, 2, 3

[25] H. Wang, A. Kläser, C. Schmid, and C. Liu. Action recognition by dense trajectories. In *CVPR*, pages 3169–3176, 2011. 2, 3

[26] H. Wang and C. Schmid. Action recognition with improved trajectories. In *ICCV*, pages 3551–3558, 2013. 2, 3