Track and Segment: An Iterative Unsupervised Approach for Video Object Proposals

Fanyi Xiao and Yong Jae Lee
University of California, Davis
{fanyix, yjlee}@cs.ucdavis.edu

Abstract

We present an unsupervised approach that generates a diverse, ranked set of bounding box and segmentation video object proposals—spatio-temporal tubes that localize the foreground objects—in an unannotated video. In contrast to previous unsupervised methods that either track regions initialized in an arbitrary frame or train a fixed model over a cluster of regions, we instead discover a set of easy-to-group instances of an object and then iteratively update its appearance model to gradually detect harder instances in temporally-adjacent frames. Our method first generates a set of spatio-temporal bounding box proposals, and then refines them to obtain pixel-wise segmentation proposals. We demonstrate state-of-the-art segmentation results on the SegTrack v2 dataset, and bounding box tracking results that perform competitively to state-of-the-art supervised tracking methods.

1. Introduction

Generating object proposals—a set of candidate object-like regions in an image that may contain the object-of-interest—from *static images* has been extensively studied in recent years [1, 50, 2, 60, 27, 41]. The success of these methods now serves as a keystone to many state-of-the-art object detection [49, 22, 18] and semantic segmentation [21, 10] algorithms. Object proposals are beneficial in two main aspects: (1) *Computation*: compared to sliding window detection, they greatly reduce the number of regions in an image that must be considered (from potentially millions to thousands); and (2) *Recognition accuracy*: they tend to reduce non-object regions that would otherwise result in false-positive detections [23].

Compared with static images, video provides rich spatio-temporal information that can greatly benefit learning algorithms. Object proposals are equally, if not more, needed in the video domain since the number of regions in a video is much larger than that of a single image. Furthermore, many video applications including summarization, activity recognition, and retrieval would benefit tremendously from a robust video object proposals method that can reduce the complexity of a video by focusing on its main objects. For example, by reducing a long video down to a small set of spatio-temporal tubes consisting of the main objects, more accurate video summaries could be produced.

Figure 1. Given an unannotated video, our algorithm produces a set of spatio-temporal bounding box proposals and segmentation proposals that localize the foreground objects. Here we show one proposal (out of multiple) for each type.

Existing approaches for video object proposals [32, 45, 40, 38, 15] (and more generally, video object segmentation [46, 6, 19, 51, 7, 11]) employ bottom-up or learned top-down appearance and motion cues to group pixels into spatio-temporal tubes that may belong to the same object. However, most methods either attempt to group pixels from all frames [19, 57, 11] or track regions that are initialized from arbitrary frames (e.g., the 1st frame) [6, 8, 33]. Consequently, these methods are susceptible to well-known challenges associated with clustering and tracking (model selection and computational complexity for the former, and initialization, drifting, and occlusion for the latter).

Inspired by the *key-segments* approach of [32], we instead sample a group of *easy* instances for each candidate object in the video to initialize an appearance model, and use that model to detect other "harder" instances of the object in the remaining frames. An object's easy instances are the regions that are object-like in appearance and have distinct motion against their immediate surrounding background. These regions are likely to span a single object, and therefore exhibit more appearance regularity, making them easier to group.

However, unlike [32], our key idea is to *iteratively* discover the harder instances in adjacent frames and update the object's appearance model in a self-paced manner, which allows the learned model to adapt and be more robust to (potentially) large appearance variations of the object. Furthermore, we work with bounding boxes, and only generate pixel-level segmentations conditioned on the detected boxes once all frames have been covered. Operating on bounding boxes substantially reduces computational complexity, since individual pixel predictions can be avoided. Finally, we show how to explicitly enforce the initial easy instances to be temporally spread-out across the video. This helps to reduce drifting, since any new frame in which the object needs to be detected is likely to be temporally-close to at least one of the initial easy instances. It also has the additional benefit to help focus on the main foreground objects that consistently appear throughout the video, and give less emphasis to background objects that appear over only a short period of time.

Contributions. Our main contribution is a novel unsupervised algorithm that generates a set of spatio-temporal video object proposal boxes and segmentations (see Fig. 1). By discovering an object's easy instances first, and gradually detecting harder instances in temporally-adjacent frames, our algorithm effectively adapts to the object's changing appearance over time. We conduct experiments to evaluate our spatio-temporal bounding box and segmentation proposals. To evaluate our bounding box proposals, we compare with existing tracking algorithms, which require human annotation in the first frame. We demonstrate competitive results on the Visual Tracker benchmark [55], even though we do not use any human supervision. To evaluate our segmentation proposals, we compare with existing video segmentation algorithms and show state-of-the-art results on the SegTrack-v2 [33] dataset.

2. Related Work

Video segmentation. Previous video segmentation algorithms can be roughly categorized into three types. The first includes methods that cluster pixels using appearance and optical flow-based motion information across all frames [19, 56, 11]. The second clusters long-range point trajectories [7, 8, 36, 43, 39, 37], which tend to handle rigid objects better. The main limitation of these approaches is their lack of an explicit notion of *object appearance*; i.e., with only low-level bottom-up information, they tend to oversegment objects. We instead discover a small but diverse set of easy instances that likely belong to the same object, and use them as top-down supervision to detect the harder instances in the remaining frames.

The third type of methods, which is closest to our approach, compute segments on each frame and link them together through bottom-up or learned appearance matching and optical flow [6, 32, 3, 40, 16, 38, 53, 15]. The key difference is that we first discover a set of easy instances of an object to build an initial model, and then we *iteratively* refine the model while simultaneously discovering harder instances in temporally-adjacent frames. For this, we leverage the fact that the object's appearance will be smoothly-varying in time. As more and more instances are discovered, the model becomes more robust to the object's changing appearance, and allows us to link together the object's instances that might otherwise be difficult to group due to large appearance variations of the object. While the method in [33] also iteratively refines a model to track an object over the video, the model is initialized with regions from the first frame, so can be more susceptible to drifting. Finally, unlike [15, 27, 41, 38], which *learn* to propose objects either in videos or static images, our approach does not require any labeled video/image to train and instead adapts to each unknown object on a per-video basis.

Tracking. Tracking algorithms (e.g., [20, 59, 55, 48, 17]) share our goal of localizing the same object over the video, but require human-annotation as initialization in the first-frame. In particular, the tracking approaches of [48, 24] use the framework of self-paced learning [4, 29] to carefully choose which frames to learn and update a tracking model. We also iteratively update our model and detections after initializing with the easy instances. However, unlike [48, 24], we do not require any human supervision. Moreover, in addition to bounding boxes, we output pixel-level segmentation masks.

3. Approach

We are given an unlabeled video $V=\{f_1, \ldots, f_N\}$ with N frames, and our goal is to discover the main objects in every frame that they appear, without knowing their categories a priori. To this end, we propose to generate a set of video object proposals – spatio-temporal tubes that track objects that have salient appearance and motion, and appear frequently throughout the video.

Our approach consists of three main steps: *initialization*, *iterative growing*, and *pixel-wise segmentation*. In the initialization step, we discover and rank a set of clusters; each cluster contains easy bounding box instances of an object in the video that have salient *object-like appearance* and *motion*. During iterative growing, we iteratively grow each cluster to detect harder instances of the corresponding object throughout the entire video. Finally, conditioned on the discovered object bounding boxes, we apply a pixel-wise segmentation algorithm to obtain fine-grained object segmentation masks in each frame.

3.1. Initialization

Our initialization step aims to discover a set of clusters, each comprised of the *easy* instances of a candidate object in the video that are spread-out in time. We define as

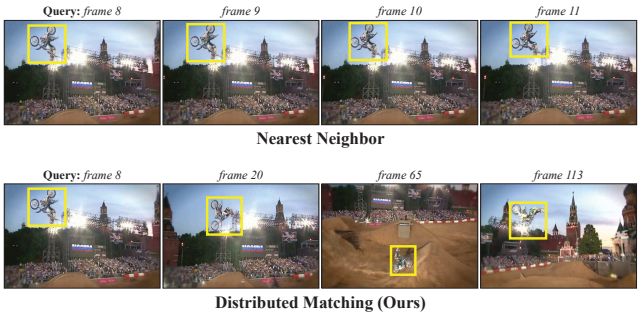

Figure 2. Directly searching for nearest neighbors in all frames tends to return patches from consecutive frames, which results in homogeneous and non-informative clusters (first row). Our approach partitions the video into uniform-length temporal segments, and takes one nearest neighbor from each segment. This produces more diverse and informative clusters (second row).

easy instances those that have salient appearance and motion with respect to their surrounding background, since they will be easier for a clustering algorithm to group. By finding instances of an object that are spread-out in time, we can focus on the foreground objects that consistently appear throughout the video and give less emphasis to background objects that appear over only a short period of time.

To identify the easy instances, we begin with a construction similar to that of [32]. To ensure good coverage of the foreground objects, we first generate a large set of *static* object proposals in each frame. Since there can be many frames in the video, we need a fast object proposals method to reduce runtime complexity. To this end, we use Edge Boxes [60], which produces ∼1000 box proposals in an image in 0.25 seconds. The algorithm scores each proposal based on the edge contours that are wholly-contained in it, which is indicative of the likelihood that the proposal contains a whole object [60]. We use this score to measure the object-like appearance s_a and motion distinctiveness s_m of each proposal. Specifically, we extract 1000 proposals each from the RGB frame and the frame's optical flow magnitude map (for a total of 2000 proposals), and then for each proposal, compute its s_a and s_m on the edgemaps [13] computed on the RGB frame and flow map, respectively. Finally, we compute a single combined objectness score for each proposal: $s = s_a * s_m$. Taking the product gives high score to the proposals that have *both* high appearance and high motion scores. In each frame, we retain the top 25 proposals with the highest objectness scores.

Let R be the set of retained high-scoring proposals across the video. We next identify in R those that belong to the same object *and* are spread-out in time, since we would like to focus on the foreground objects that consistently appear throughout the video and ignore background objects that only appear for a short time. Because we do not know how many foreground objects are in the video, we generate a large number of candidate clusters via a soft-clustering approach. The idea is to take each proposal in R as the query patch and retrieve its k-nearest neighbors to form a cluster. However, directly taking nearest neighbors for a query patch tends to produce clusters of patches from consecutive frames since they will be very similar in appearance (see Fig. 2). Thus, we instead *force* the nearest neighbors to be spread-out in time.

Specifically, we first split the video into N_s=10 uniform-length contiguous segments in time (i.e., each segment has N/N_s frames), and treat each proposal in R as a seed. We then compute a seed's best matching proposal (nearest neighbor) in each of the N_s segments; we compute the matching by taking the inner-product between proposals in L2-normalized fc_7 feature space (fully-connected 7th layer activation feature of AlexNet [28], pre-trained on ImageNet classification). Thus, each seed produces a cluster with N_s instances that are spread-out over the video. To account for any foreground objects that may be missing in some of the N_s segments or have widely-varying appearance throughout the video (e.g., a person who is initially facing the camera and later faces away from the camera), we can further create variable-sized clusters by retaining only the k most similar among the N_s nearest neighbors.

Finally, we compute a score for each cluster by summing the objectness score of its instances multiplied with the instances' appearance-similarity to the seed, which rewards large clusters whose instances likely belong to the same object: $s(c) = \sum_j s(p^j) * \left(\phi(p^j)^T \phi(p^{seed})\right)$, where c is a cluster, j indexes c's instances, p^{seed} is the seed of c, and $\phi(\cdot)$ denotes the L2-normalized fc_7 feature. Since we generate clusters using all proposals in R as a seed, there will be many redundant clusters. We therefore perform cluster-level non-maximum suppression: We start with an empty set $\mathbb{S} = \emptyset$, and rank all the clusters in descending order of their cluster score. We then greedily add a cluster c_i to $\mathbb{S}$=$\{\mathbb{S} \cup c_i\}$ if it is significantly different in appearance to any higher-scoring cluster already in $\mathbb{S}$, as measured by the appearance similarity between the clusters' seed proposals.

3.2. Iterative growing

The top-K ranked clusters $\{c_1, \ldots, c_K\}$ in $\mathbb{S}$ comprise a diverse set of candidate objects in the video. However, each cluster only covers a small number of easy instances (at most N_s) of an object, and some of them could be noisy. We thus need to detect the harder instances of the object in the remaining frames and correct any existing noisy ones. A natural way to proceed would be to train a detector using the cluster's instances as positives and any non-overlapping proposals in their same frames as negatives, and fire that detector on all frames (as done in [32]). However, the object's appearance can change drastically across the video, so such an approach can be prone to drift.

To tackle this, we instead propose to *iteratively* update the detector by starting with the frames that are temporally

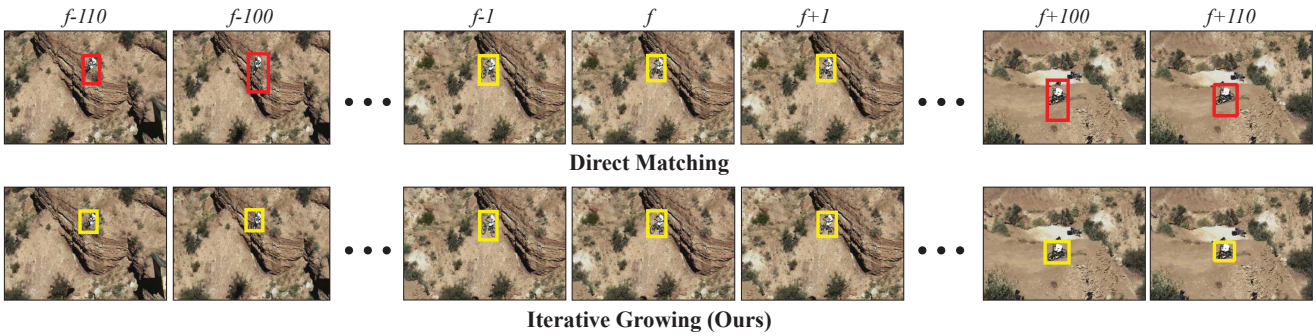

Figure 3. Starting from frame f, both direct matching and iterative growing obtain correct detections in adjacent frames $f+1$ and $f-1$. However, for frames that are very far away from f in time, direct matching can drift due to large appearance changes in the object. By iteratively growing and simultaneously updating the detector, our approach can *adapt* to the object's large appearance variation to obtain a correct detection even in far-away frames.

close to the initial set of cluster instances, and then gradually grow out to cover all N frames. See Fig. 3. When growing out (i.e., detecting new instances), we initially do not leverage any motion-based tracking cues from existing detections in neighboring frames. This is because we do not want to commit to any detection (especially early on) since there could be noisy detections that lead to drifting. We instead iteratively update the detector until all frames are covered, and then combine the final detector's outputs with motion cues to obtain the final detections.

For each cluster c,[1] we start by training an initial linear SVM detector $\mathbf{w}$ using the cluster's instances $P = \{p^j \mid j \in S\}$ as positives and any proposal with intersection-over-union ratio (IOU) less than 0.4 to any instance in P as negatives. Denote S as the initial set of frames that c already covers (i.e., has an instance in). We next exploit the property that an object's appearance will change slowly over time, in order to make our detection problem easier.

Specifically, we first identify the set of frames S', which are the $n = 3$ temporal neighbors of S that are not yet covered. We then fire $\mathbf{w}$ on all 1000 proposals in each frame in both S' *and* S. By firing the detector even on the frames in S, we can update any mis-detections in those existing frames. We take the proposal with the highest detection score in each frame, and add them to the set of cluster instances P. We then retrain $\mathbf{w}$ with the new and updated instances in P as positives, and any proposal with IOU less than 0.4 to any instance in P as negatives. We update the set of frames that cluster c covers as $S = \{S \cup S'\}$. We repeat this process of detecting/updating instances in neighboring/existing frames and retraining the detector until all frames are covered, i.e., $S = V$.

Finally, we take the final trained detector $\mathbf{w}$, and fire it back on all proposals in all frames in V. We combine its per-frame detections with optical flow based motion cues to encourage smooth detections in time. Formally, we solve the following optimization problem to get a final set of detections $P = \{p^1, \ldots, p^N\}$ using dynamic programming:

$$\max_P J(P) = \sum_{j=1}^{N} \mathbf{w}^T \phi(p^j) + \sum_{j=1}^{N-1} \text{IOU}(p^{j+1}, q^{j+1}), \quad (1)$$

where $\phi(p^j)$ is the fc_7 feature of p^j, q^{j+1} is the bounding box location in frame j+1 obtained by shifting p^j in frame j to frame j+1 according to its average optical flow displacement vector, and IOU is the intersection-over-union ratio.

The final set of detections P for cluster c forms a spatio-temporal bounding box tube. We repeat the above for each of the top-K clusters.

3.3. Pixel-wise segmentation

Thus far, our approach produces a set of spatio-temporal bounding box proposals given an unannotated video. For some applications however (e.g., semantic video segmentation), the ensuing algorithm may require the video object proposals to be pixel segmentations instead of bounding boxes. Hence, we next show how to output pixel-wise segmentation masks, given the bounding boxes we obtained in the previous section as initialization. Similar to GrabCut [44], the main idea is to use the bounding boxes as weak supervision, and iteratively refine a pixel-level appearance model of the object and its corresponding foreground object segmentation in each frame of the video.

Since our bounding box detections have already provided a rough localization of the object-of-interest, we can ignore any regions that are spatially far from each bounding box when computing their segmentations. To this end, we define an *operating region*, with size $[3\times w, 3\times h]$, for each bounding box proposal p (where w and h are the width and height of p, respectively) and is centered on the center pixel of p. We then initialize a pixel-level appearance model by taking all the pixels inside each bounding box as positives, and all pixels outside the *safe region*, which is a 32 pixel-wide boundary that immediately surrounds each bounding box, but within the operating region as negatives. The safe region accounts for any mis-localizations of the input bounding boxes. See Fig. 4.

[1] We drop the cluster subscript for simplicity.

Figure 4. The yellow box shows the spatio-temporal bounding box proposal produced from our iterative growing procedure in one frame. To compute its pixel-level segmentation, we first define an *operating region* (orange box) in order to ignore any pixels that are spatially far from the proposal. The blue dotted box is a *safe region*, outside of which we sample negative data to train our pixel-level appearance model, in order to account for any mis-localizations of the initial box proposal.

To represent each pixel, we adapt the *hypercolumn* representation [21], which models both low-level appearance and high-level learned semantics. Specifically, we combine the activation features of the *pool2* and *pool5* layers of AlexNet [28]. In order to make sure we have a large enough spatial resolution to model small objects, we resize each operating region to 2400×2400 pixels. This produces a corresponding activation feature map of size 147×147 and 72×72 for *pool2* and *pool5*, respectively. We bi-linearly interpolate the *pool2* feature map to 72×72 to match the size of the *pool5* map. We then train a logistic regression classifier with the positive (inside the bounding box) and negative (outside the safe region) pixels.

To compute the segmentation for a frame f, we define a graph over its operating region where a node corresponds to a pixel, and an edge between two nodes corresponds to the cost of a cut. The cost function we minimize is:

$$C(l,f) = \sum_{i\in\mathcal{O}} D_i(l_i) + \sum_{i,j\in\mathcal{N}} V_{i,j}(l_i,l_j), \qquad (2)$$

where l is a labeling of the pixels, $\mathcal{O} = \{o_1,\ldots,o_m\}$ is the set of m pixels in the operating region, $\mathcal{N}$ consists of the four spatially neighboring pixels, and i and j index the pixels. Each pixel o_i is assigned to $l_i \in \{1,0\}$, where 1 and 0 correspond to foreground and background, respectively.

We use the pixel-level logistic regression classifier's probability output to compute the data term D_i, which defines the cost of labeling pixel o_i with label l_i. The neighborhood term $V_{i,j}$ encourages label smoothness in space. We compute an edge map using [13] and assign $V_{i,j}$ between o_i and o_j with their edge confidence, which favors assigning the same label to neighboring pixels that do not have a strong edge between them. We then minimize Eqn. 2 using graph-cuts [5] to obtain the pixel-wise segmentation for frame f. Finally, we update the pixel appearance model with the newly obtained segmentations from all frames. We take the new model and use it to update the segmentations; we repeat this process until convergence (i.e., the pixel assignments in all frames do not change). See Fig. 5.

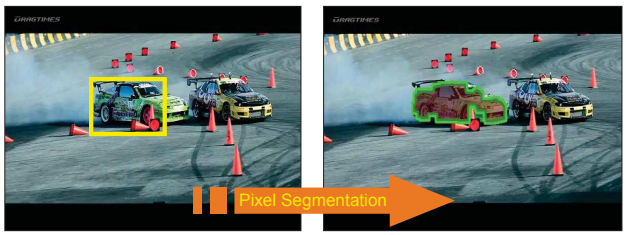

Figure 5. Given a bounding box proposal output in a frame (left, yellow box), our algorithm trains a pixel-level appearance model to produce a segmentation mask (right, green boundary). Note that this not only produces a segmentation proposal, but it can also correct mis-localizations from the bounding box proposal.

Due to the large receptive fields of *pool2* and *pool5* features, it is difficult to obtain accurate *pixel-level* predictions. Therefore, we refine the foreground mask with a simple post-processing step: we represent each pixel with an RGB feature vector to train a GMM with 5 components each for the predicted foreground/background pixels. We then apply Eqn. 2 again on the pixels to get the final foreground mask.

Bounding box proposal refinement. Finally, our pixel segmentation can in turn be used to improve the bounding box localization that it was initialized on. Among all connected components labeled as foreground that are overlapping with the initial bounding box, we simply keep those whose area is larger than $0.6\times$ the area of the largest component. We then take the bounding box that tightly encloses all the selected components. The refined boxes are taken as our final spatio-temporal box proposals, together with the corresponding segmentation proposals.

4. Experiments

We evaluate our bounding box and pixel segmentation proposals against state-of-the-art tracking and video segmentation methods, and conduct ablation studies to analyze the contribution of our iterative growing procedure and the synergy of our bounding box and segmentation tubes.

Implementation details. We set the number of initial proposal clusters to $K = 85$ for the Visual Tracker Benchmark [55] and $K = 150$ for the SegTrack-v2 dataset [33]. For very difficult videos (e.g. "Penguin" in SegTrack-v2) that do not have well-defined foreground objects, we find that more clusters are needed in order to get an initialization that corresponds to the human-annotated object. For generating clusters, we set the number of neighbors of a seed to $k = \{1,3,5,7,9\}$ for SegTrack-v2 and fix it to $k = 9$ for Visual Tracker Benchmark. Empirically, we find that a smaller k produces better cluster initializations for videos that have many confusing appearance patterns.

4.1. Evaluation of box proposals

We first evaluate the quality of our spatio-temporal box proposals against tracking algorithms on the Visual Tracker

	SCM [59]	Struck [20]	TLD [26]	ASLA [25]	CXT [12]	VTD [30]	VTS [31]	TGPR [17]	RPT [34]	[52]	[35]	Ours
Supervised?	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	N
Overall	0.499	0.473	0.437	0.434	0.424	0.416	0.416	0.539	0.576	0.599	0.612	0.437
IV	0.472	0.427	0.399	0.429	0.365	0.42	0.428	N/A	0.555	0.598	0.577	0.453
SV	0.518	0.425	0.421	0.452	0.389	0.405	0.4	N/A	0.535	0.558	0.558	0.451
OCC	0.487	0.412	0.402	0.376	0.369	0.404	0.398	N/A	N/A	0.571	0.615	0.434
DEF	0.448	0.393	0.378	0.372	0.324	0.377	0.368	N/A	N/A	0.644	0.615	0.469
MB	0.298	0.433	0.404	0.258	0.369	0.309	0.304	N/A	0.559	0.580	N/A	0.460
FM	0.296	0.461	0.417	0.248	0.384	0.303	0.299	N/A	0.549	0.565	0.54	0.507
IPR	0.457	0.443	0.416	0.425	0.449	0.43	0.415	N/A	0.569	0.555	N/A	0.423
OPR	0.47	0.431	0.42	0.422	0.416	0.435	0.425	N/A	0.553	0.581	0.605	0.443
OV	0.361	0.459	0.457	0.312	0.427	0.446	0.443	N/A	N/A	0.592	0.596	0.543
BC	0.45	0.458	0.345	0.408	0.338	0.425	0.428	N/A	0.606	0.564	0.58	0.352
LR	0.279	0.372	0.309	0.157	0.312	0.177	0.168	N/A	N/A	0.514	N/A	0.372

Table 1. We compare our *unsupervised* video object box proposals to state-of-the-art *supervised* tracking methods on the Visual Tracker Benchmark [55]. The supervised methods require human annotation in the first frame to initialize their tracker. Even without this requirement, our unsupervised approach is able to outperform many of the baselines. We measure accuracy in terms of the Area Under Curve (AUC) of the One Pass Evaluation (OPE) success plot as defined in [55]. Higher is better. *IV-Illumination Variation, SV-Scale Variation, OCC-Occlusion, DEF-Deformation, MB-Motion Blur, FM-Fast Motion, IPR-In-Plane Rotation, OPR-Out-of-Plane Rotation, OV-Out-of-View, BC-Background Clutter, LR-Low Resolution.* (Baseline results are taken from [55, 17, 34, 52, 35].)

Benchmark [55]. This dataset contains 50 test sequences (with frame lengths that range from 71 to 3872) with various challenging properties like illumination/scale variation, occlusion, deformation, etc., which make it an excellent testbed for evaluating the robustness of our box proposals.

We use the standard *success plot* performance metric defined in [55], which measures the ratio of frames that have an intersection-over-union overlap (IOU) score above a threshold, and draw a curve by varying that threshold from 0 to 1 in 0.05 increments. The overall performance of an algorithm is then the area under the success plot curve (AUC). We evaluate our approach with the box proposal that corresponds to the object with ground-truth annotation, and report the ranking of that proposal based on its cluster score, as described in Sec. 3.1.

First, we show the performance of our proposals compared with several supervised tracking methods. We compare against the baselines using the One Pass Evaluation (OPE) [55], which initializes the tracker with a human-annotated box in the first-frame. Note however, that *we do not use this annotation*, as ours is unsupervised. We measure performance both in terms of the average OPE success plot AUC across *all* videos, as well as the AUC for different sub-categories of videos defined by various challenging factors, e.g., illumination variation, scale variation, occlusion, etc. As shown in Table 1, our algorithm performs competitively with existing state-of-the-art supervised tracking methods—even outperforming several of them [26, 25, 12, 30, 31]—despite the fact that our method does not require *any human annotation*. In contrast, all of the baselines require human annotation on the first frame to initialize their tracker.

On average, we need 123.7 proposals to achieve the results in Table 1, which indicates that our ranking is able to focus on the foreground objects despite the various challenging factors present in these video. See the supp. material for detailed rankings per-subcategory.

4.2. Evaluation of segmentation proposals

We next evaluate our spatio-temporal segmentation proposals. We use the SegTrack-v2 [33] dataset, which contains 14 video sequences with frame lengths varying from 21 to 279 across the sequences. Every frame is annotated with a binary (pixel-level) foreground/background mask. We compare with previous state-of-the-art unsupervised methods [19, 32, 42, 33] and also to a recent supervised method [54] that requires human annotation of the object's boundary in the first frame.

To evaluate segmentation accuracy, we use the intersection-over-union ratio (IOU) measure: $IOU = \frac{|A \cap GT|}{|A \cup GT|}$, where A is the binary segmentation produced by the algorithm and GT is ground-truth binary mask. We evaluate our approach with the segmentation tube that corresponds to the object with ground-truth annotation, and report the ranking of that tube based on its cluster score.

Table 2 shows the results. Our method produces state-of-the-art results compared with previous unsupervised methods [33, 32, 19] with a moderate number of proposals (122 on average across the videos). The approach of [19] clusters pixels using bottom-up motion and appearance cues, and thus needs to generate many proposals to obtain good performance. The state-of-the-art approaches of [33, 32] perform better with fewer proposals, by leveraging top-down cues from learned static object proposals [9, 14]. However, [33] tracks multiple static proposals initialized in the first frame, and so inherits challenges associated with tracking (e.g., drifting). This is likely the reason for their low accuracy on videos with fast moving objects like CheetahDeer and BMXBike. In contrast, our approach discovers the easy instances *throughout* the video, and iteratively updates the model to grow-out from those instances. Thus, we find our approach to be more robust to drifting. For this same reason,

Sequence/Object	[33]+CSI	[32]	[19]	Ours	[54]
Supervised?	N	N	N	N	Y
Mean per object	65.9	45.3	51.8	**69.1**	71.8
Mean per sequence	71.2	57.3	50.8	**73.9**	72.2
Girl	**89.2**	87.7	31.9	86.4	84.6
Birdfall	62.5	49.0	57.4	**72.5**	78.7
Parachute	93.4	**96.3**	69.1	95.9	94.4
CheetahDeer	37.3	44.5	18.8	**61.2**	66.1
CheetahCheetah	**40.9**	11.7	24.4	39.4	35.3
MonkeydogMonkey	71.3	**74.3**	68.3	74.0	82.2
MonkeydogDog	18.9	4.9	18.8	**39.6**	21.1
Penguin#1	51.5	12.6	**72.0**	53.2	94.2
Penguin#2	76.5	11.3	**80.7**	72.9	91.8
Penguin#3	**75.2**	11.3	75.2	74.4	91.9
Penguin#4	57.8	7.7	**80.6**	57.2	90.3
Penguin#5	**66.7**	4.2	62.7	63.5	76.3
Penguin#6	50.2	8.5	**75.5**	65.7	88.7
Drifting Car#1	**74.8**	63.7	55.2	70.7	67.3
Drifting Car#2	60.6	30.1	27.2	**70.7**	63.7
Hummingbird#1	**54.4**	46.3	13.7	53.0	58.3
Hummingbird#2	72.3	**74.0**	25.2	70.5	50.7
Frog	72.3	0	67.1	**80.2**	56.3
Worm	82.8	**84.4**	34.7	82.4	79.3
Soldier	**83.8**	66.6	66.5	76.3	81.1
Monkey	**84.8**	79.0	61.9	83.1	86.0
Bird of Paradise	**94.0**	92.2	86.8	90.0	93.0
BMXPerson	85.4	87.4	39.2	**86.1**	88.9
BMXBike	24.9	38.6	32.5	**40.3**	5.70
Avg. # of Proposals	60.0	10.6	336.6	121.9	N/A

Table 2. Segmentation results on SegTrack-v2 in terms of mean segmentation IOU with ground-truth. Higher is better. For both the "mean per object" and the "mean per sequence" metric, we outperform state-of-the-art unsupervised methods [33, 32, 19] using a moderate number of proposals. We also perform competitively to the supervised method of [54], which requires human annotation on the first frame, whereas we require none. (Baseline results are taken from [33, 54].)

we outperform the key-segments approach of [32], which like ours, discovers easy instances and builds a model to detect the object in new frames, but unlike ours, does not iteratively refine the model.

Our algorithm even performs competitively to the recent state-of-the-art *supervised* method of [54] that requires manual initialization in the first frame. Although our average under the "mean per object" metric is lower (69.1% vs. 71.8%), we perform better in terms of the "mean per sequence" metric (73.9% vs. 72.2%), *without any supervision*. The main reason our average under the "mean per object" metric is lower is due to the sequences in the Penguin video, which has a group of penguins that are spatially very close to each other, and are similar in appearance and motion. This makes it very difficult for an unsupervised method to keep track of a single penguin. If we remove the penguin sequences from the evaluation, our average improves to 70.7%, while the average of [54] drops to 66.3%, under the "mean per object" metric.

The average number of proposals our approach produces also compares favorably to that of previous methods (see last row of Table 2, the lower the better). Unfortunately, the penguin sequences again hurt our average ranking. We need to generate hundreds of proposals to obtain the one corresponding to a specific penguin since many proposals end up with similar rank due to the penguins' similarity in appearance and motion. If we remove the penguin sequences, the average number of proposals needed to localize the ground-truth object drops to 79.6. See supp. material for per-sequence rankings.

	[53]	[58]	[40]	[7]	Ours
Average Pixel Error	4766	25289	5859	16074	**2464.5**

Table 3. Segmentation results on SegTrack-v2 in terms of average pixel error with ground-truth. Lower is better. We outperform all previous unsupervised methods by a large margin (∼48% error reduction compared with [53]) under this metric. (Baseline results are taken from [53].)

Finally, we also compare with other unsupervised methods [53, 58, 40, 7] that report results in terms of average pixel error, which is the average number of incorrectly labeled pixels across frames. Table 3 shows the result. We outperform previous methods under this metric by a large margin. For example, compared with the results of [53], we reduce the error by ∼48%.

4.3. Ablation studies

We next study the different components of our algorithm. We use the Visual Tracker Benchmark [55] and measure the performance of our spatio-temporal box proposals using the AUC of the overall success plot.

We first study the effect of our iterative growing procedure by comparing to a baseline that does not iteratively update a model when growing out. Specifically, the baseline takes the detector $\mathbf{w}$ trained on an initially discovered cluster, and fires it on all frames in the video (instead of firing only on the temporally-adjacent frames and iteratively updating the model). This baseline produces an AUC of 0.319, which is significantly lower than the 0.365 produced with iterative growing. This demonstrates that iterative growing can help avoid drift, as shown in Fig. 3.

The next aspect we investigate is the synergy of our box output and segmentation output. Specifically, we measure how much our pixel segmentation helps in producing better localized box proposals (recall from Sec. 3.3 that we update the bounding box proposals given the segmentation masks). We compare the bounding boxes produced before and after segmentation refinement. The quality of the bounding boxes further increases from 0.365 to 0.437 after refinement, which shows that our segmentations indeed lead to better object localizations.

4.4. Runtime speed analysis

Since object proposals can be building blocks to many vision applications, it is essential that they be fast to compute, so that they are not the computational bottleneck. Thus, we finally perform a detailed run-time speed anal-

Figure 6. (**rows 1-3**) Qualitative results of our spatial-temporal box proposals on the Visual Tracker Benchmark. Our proposals localize the object well, despite challenging factors in the videos, e.g., scale variation (first row), clutter (second row), in-plane rotation (third row), etc. (**rows 4-6**) Qualitative results of our spatial-temporal segmentation proposals on SegTrack-v2. Our approach produces fine details of the object boundary (fourth row), and handles occlusion well (fifth row). Lastly, for the very difficult case in which objects have thin structures, e.g., the bicycle in the last row, our algorithm leaks out qualitatively, even though quantitatively we outperform previous methods.

ysis of our approach. For each of the three steps in our algorithm (initialization, iterative growing, and pixel-level segmentation), we profile the average computation time spent to process each frame. We conduct our analysis on the "Woman" sequence in the Visual Tracker Benchmark, which has 597 frames and is closest to the average frame length (584 frames) of that dataset. We measure the timings on a machine with an Intel i7 3.40GHz CPU and an NVIDIA Tesla K40 GPU.

It takes a total of 3.0 seconds per frame for the initialization stage, which includes computing and scoring the static image proposals (Edge Boxes on RGB image and optical flow magnitude map), optical flow computation (we use the fast GPU-based method of [47]), and obtaining the initial ranked set of clusters $\mathbb{S}$. For iterative growing, extracting fc_7 features for all 2000 proposals for each frame takes 1.25 seconds on a GPU. The remaining spatio-temporal bounding box tube processing—iteratively training an SVM detector and firing it to detect new instances and refine existing ones—takes 0.24 seconds per frame per spatio-temporal tube. For the pixel-wise segmentation stage, the time spent on extracting a hypercolumn feature is 0.85 seconds per frame on a GPU. Generating pixel-segmentations takes 0.46 seconds per frame per spatio-temporal tube.

Note that the feature computation for generating the spatio-temporal bounding box is shared and thus only needs to be done once (i.e., independent of the number of proposals that we generate). Whereas the hypercolumn feature used to generate the segmentation tubes needs to be computed separately for each proposal. While the computational cost in generating our segmentation proposals is comparable to that of previous approaches [32, 33], generating our box proposals is much faster and thus can be applied in a more practical setting.

5. Conclusion

We presented an unsupervised approach for spatio-temporal video object proposals. It identifies and groups easy instances of an object in the video to initialize an appearance model, and iteratively updates the model while detecting new instances in temporally-adjacent frames. We demonstrated our method's effectiveness on several datasets, showing state-of-the-art unsupervised video segmentation results, and competitive bounding box tracking results compared to supervised baselines.

Acknowledgements. This work was supported in part by an Amazon Web Services Education Research Grant and GPUs donated by NVIDIA.

References

[1] B. Alexe, T. Deselaers, and V. Ferrari. Measuring the objectness of image windows. *PAMI*, 2012. 1

[2] P. Arbelaez, J. Pont-Tuset, J. Barron, F. Marques, and J. Malik. Multiscale combinatorial grouping. In *CVPR*, 2014. 1

[3] D. Banica, A. Agape, A. Ion, and C. Sminchisescu. Video object segmentation by salient segment chain composition. In *ICCV Workshops*, 2013. 2

[4] Y. Bengio, J. Louradour, R. Collobert, and J. Weston. Curriculum learning. In *ICML*, 2009. 2

[5] Y. Boykov, O. Veksler, and R. Zabih. Efficient approximate energy minimization via graph cuts. *PAMI*, 2001. 5

[6] W. Brendel and S. Todorovic. Video object segmentation by tracking regions. In *ICCV*, 2009. 1, 2

[7] T. Brox and J. Malik. Object segmentation by long term analysis of point trajectories. In *ECCV*, 2010. 1, 2, 7

[8] T. Brox and J. Malik. Large displacement optical flow: descriptor matching in variational motion estimation. *PAMI*, 2011. 1, 2

[9] J. Carreira and C. Sminchisescu. Constrained Parametric Min-Cuts for Automatic Object Segmentation. In *CVPR*, 2010. 6

[10] J. Dai, K. He, and J. Sun. Boxsup: Exploiting bounding boxes to supervise convolutional networks for semantic segmentation. *arXiv preprint arXiv:1503.01640*, 2015. 1

[11] M. V. den Bergh, G. Roig, X. Boix, S. Manen, and L. V. Gool. Online video seeds for temporal window objectness. In *ICCV*, 2013. 1, 2

[12] T. B. Dinh, N. Vo, and G. Medioni. Context tracker: Exploring supporters and distracters in unconstrained environments. In *CVPR*, 2011. 6

[13] P. Dollár and C. L. Zitnick. Structured forests for fast edge detection. In *ICCV*, 2013. 3, 5

[14] I. Endres and D. Hoiem. Category Independent Object Proposals. In *ECCV*, 2010. 6

[15] K. Fragkiadaki, P. Arbeláez, P. Felsen, and J. Malik. Learning to segment moving objects in videos. In *CVPR*, 2015. 1, 2

[16] F. Galasso, N. S. Nagaraja, T. Jimenez Cardenas, T. Brox, and B. Schiele. A unified video segmentation benchmark: Annotation, metrics and analysis. In *ICCV*, 2013. 2

[17] J. Gao, H. Ling, W. Hu, and J. Xing. Transfer learning based visual tracking with gaussian processes regression. In *ECCV*, 2014. 2, 6

[18] R. Girshick. Fast R-CNN. In *ICCV*, 2015. 1

[19] M. Grundmann, V. Kwatra, M. Han, and I. Essa. Efficient hierarchical graph-based video segmentation. In *CVPR*, 2010. 1, 2, 6, 7

[20] S. Hare, A. Saffari, and P. H. Torr. Struck: Structured output tracking with kernels. In *ICCV*, 2011. 2, 6

[21] B. Hariharan, P. Arbeláez, R. Girshick, and J. Malik. Hypercolumns for object segmentation and fine-grained localization. In *CVPR*, 2015. 1, 5

[22] K. He, X. Zhang, S. Ren, and J. Sun. Spatial pyramid pooling in deep convolutional networks for visual recognition. In *ECCV*, 2014. 1

[23] J. Hosang, R. Benenson, P. Dollár, and B. Schiele. What makes for effective detection proposals? *PAMI*, 2015. 1

[24] C. Huang, B. Wu, and R. Nevatia. Robust object tracking by hierarchical association of detection responses. In *ECCV*, 2008. 2

[25] X. Jia, H. Lu, and M.-H. Yang. Visual tracking via adaptive structural local sparse appearance model. In *CVPR*, 2012. 6

[26] Z. Kalal, K. Mikolajczyk, and J. Matas. Tracking-learning-detection. *PAMI*, 2012. 6

[27] P. Krähenbühl and V. Koltun. Learning to propose objects. In *CVPR*, 2015. 1, 2

[28] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *NIPS*, 2012. 3, 5

[29] M. P. Kumar, B. Packer, and D. Koller. Self-paced learning for latent variable models. In *NIPS*, 2010. 2

[30] J. Kwon and K. M. Lee. Visual tracking decomposition. In *CVPR*, 2010. 6

[31] J. Kwon and K. M. Lee. Tracking by sampling trackers. In *ICCV*, 2011. 6

[32] Y. J. Lee, J. Kim, and K. Grauman. Key-segments for video object segmentation. In *CVPR*, 2011. 1, 2, 3, 6, 7, 8

[33] F. Li, T. Kim, A. Humayun, D. Tsai, and J. M. Rehg. Video segmentation by tracking many figure-ground segments. In *ICCV*, 2013. 1, 2, 5, 6, 7, 8

[34] Y. Li, J. Zhu, and S. C. Hoi. Reliable patch trackers: Robust visual tracking by exploiting reliable patches. In *CVPR*, 2015. 6

[35] C. Ma, X. Yang, C. Zhang, and M.-H. Yang. Long-term correlation tracking. In *CVPR*, 2015. 6

[36] P. Ochs and T. Brox. Higher order motion models and spectral clustering. In *CVPR*, 2012. 2

[37] P. Ochs, J. Malik, and T. Brox. Segmentation of moving objects by long term video analysis. *PAMI*, 2014. 2

[38] D. Oneata, J. Revaud, J. Verbeek, and C. Schmid. Spatio-temporal object detection proposals. In *ECCV*, 2014. 1, 2

[39] G. Palou and P. Salembier. Hierarchical video representation with trajectory binary partition tree. In *CVPR*, 2013. 2

[40] A. Papazoglou and V. Ferrari. Fast object segmentation in unconstrained video. In *ICCV*, 2013. 1, 2, 7

[41] P. O. Pinheiro, R. Collobert, and P. Dollar. Learning to segment object candidates. In *NIPS*, 2015. 1, 2

[42] H. Pirsiavash, D. Ramanan, and C. C. Fowlkes. Globally-optimal greedy algorithms for tracking a variable number of objects. In *CVPR*, 2011. 6

[43] A. Ravichandran, C. Wang, M. Raptis, and S. Soatto. Superfloxels: A mid-level representation for video sequences. In *ECCV Workshops*, 2012. 2

[44] C. Rother, V. Kolmogorov, and A. Blake. Grabcut: Interactive foreground extraction using iterated graph cuts. *ACM Transactions on Graphics*, 2004. 4

[45] G. Sharir and T. Tuytelaars. Video object proposals. In *CVPR Workshops*, 2012. 1

[46] J. Shi and J. Malik. Motion segmentation and tracking using normalized cuts. In *ICCV*, 1998. 1

[47] N. Sundaram, T. Brox, and K. Keutzer. Dense point trajectories by gpu-accelerated large displacement optical flow. In *ECCV*, 2010. 8

[48] J. S. Supancic and D. Ramanan. Self-paced learning for long-term tracking. In *CVPR*, 2013. 2

[49] C. Szegedy, S. Reed, D. Erhan, and D. Anguelov. Scalable, high-quality object detection. *arXiv:1412.1441*, 2014. 1

[50] J. R. Uijlings, K. E. van de Sande, T. Gevers, and A. W. Smeulders. Selective search for object recognition. *IJCV*, 2013. 1

[51] A. Vazquez-Reina, S. Avidan, H. Pfister, and E. Miller. Multiple Hypothesis Video Segmentation from Superpixel Flows. In *ECCV*, 2010. 1

[52] L. Wang, W. Ouyang, X. Wang, and H. Lu. Visual tracking with fully convolutional networks. In *ICCV*, 2015. 6

[53] W. Wang, J. Shen, and F. Porikli. Saliency-aware geodesic video object segmentation. In *CVPR*, 2015. 2, 7

[54] L. Wen, D. Du, Z. Lei, S. Z. Li, and M.-H. Yang. Jots: Joint online tracking and segmentation. In *CVPR*, 2015. 6, 7

[55] Y. Wu, J. Lim, and M.-H. Yang. Online object tracking: A benchmark. In *CVPR*, 2013. 2, 5, 6, 7

[56] C. Xu and J. J. Corso. Evaluation of super-voxel methods for early video processing. In *CVPR*, 2012. 2

[57] C. Xu, C. Xiong, and J. Corso. Streaming hierarchical video segmentation. In *ECCV*, 2012. 1

[58] D. Zhang, O. Javed, and M. Shah. Video object segmentation through spatially accurate and temporally dense extraction of primary object regions. In *CVPR*, 2013. 7

[59] W. Zhong, H. Lu, and M.-H. Yang. Robust object tracking via sparsity-based collaborative model. In *CVPR*, 2012. 2, 6

[60] C. L. Zitnick and P. Dollár. Edge boxes: Locating object proposals from edges. In *ECCV*, 2014. 1, 3