LDA-segmentation based tracking

Felix Vilensky
Department of Electrical
and Computer Engineering
Ben Gurion University
Be'er Sheva, Israel
Email:felixvil@post.bgu.ac.il

Prof. Stanley Rotman
Department of Electrical
and Computer Engineering
Ben Gurion University
Be'er Sheva, Israel
Email:srotman@ee.bgu.ac.il

Dr. Aharon Bar Hillel
Department of Industrial
Engineering and Management
Ben Gurion University
Be'er Sheva, Israel
Email: barhille@bgu.ac.il

***Abstract*—Despite recent advances, visual object tracking remains a challenging algorithmic task. When viewed as an object:background pixel classification problem, we hypothesize that it requires a combination of strong CNN-based features with an efficient mechanism for on-line learning. We propose a novel method for short term single object tracking, based on Linear Discriminant Analysis (LDA) as the main on-line learning component. It utilizes an ensemble of LDA classifiers, which are simple to update, applied to several hidden representation levels of a backbone CNN. Each such LDA classifier is applied to all pixels in an object window, thus creating an objecthood map. The maps are then fused to find an object proposition optimising a global score including terms controlling for segmentation quality, as well as location and scale continuity. The method hence solves simultaneously the two problems of object tracking and segmentation. In spite of its simplicity, LDA application to hidden CNN layers is shown to have an impressive ability for transfer learning and modeling of object:background distinctions. The method was evaluated on the tracking benchmarks VOT2016, VOT2018 and VOT2019. It is one of the top performers of the VOT2016 and VOT2018 datasets and achieves competitive results on VOT2019.**

I. Introduction

Visual object tracking is a basic problem in the computer vision domain. Given the location (bounding box) of a target in the first frame of a video sequence, a visual object tracker has to infer the object location in subsequent video frames. This problem is easily solved by humans and yet, despite significant recent progress, robust visual object tracking remains a challenging task. Challenges arise due to multiple reasons: illumination variation, object viewpoint or background change, camera movement and image blur, non-rigid deformation and partial occlusion. The tracking problems family includes several formulations, differing w.r.t assumptions made and the typical scenarios handled. For instance, in some scenarios multiple objects have to be tracked simultaneously, and in others we have prior knowledge about the target type, like in face or car tracking. In this work we consider single object, short term tracking, of unknown object type (object agnostic). Specifically, short term tracking assumes that the object does not disappear from the scene for significant time durations, and so re-detection is not explicitly considered.

In recent years, deep convolutional neural networks (CNNs) enabled significant performance advances in most computer vision tasks [1]–[3]. Such networks construct a hierarchy of image representations, including discriminative features computed locally and organized in spatial maps. Lower level layers contain general feature maps with high spatial resolution, and higher level layers contain more semantic feature maps with low spatial resolution. While training on large scale datasets is often a key for obtaining high accuracy, CNNs also perform remarkably well in low-data domains with task-transfer techniques. Specifically the ability to successfully reuse features from a pre-trained CNN was early observed [4], and is widely used. Currently all leading tracking methods use CNNs as building blocks, with a wide spectrum of approaches. A main question is: how should CNNs be utilized in the tracking environment, where data (of the sequence to track) is scarce and unsupervised, and significant computational complexity constraints apply.

A main distinction in recent tracking work is between offline and online learning trackers. Offline trackers [5]–[10] train a neural network specifically for the tracking task, using large video tracking datasets. They learn typical object and background dynamics, but do not employ online learning during actual tracking. This allows for high inference speed, but performance may degrade due to lack of adaptation to the specific object and background of the currently tracked sequence. The degradation may be significant if such objects/background were under-represented in the training data. Online trackers [11]–[19] do not train with video datasets, and rely instead on transfer learning. They use features from CNNs pretrained for image classification (typically on the ImageNet [20] dataset) and train a model for object:background distinction during tracking. Correlation filter based trackers [11]–[18] adopt this approach, and learn a rigid filter that should obtain its maximal response on the target location in a search window. Other trackers such as TCNN [19] use finetuning of the top levels of a CNN during tracking.

Almost by definition, online trackers can better adapt to the tracked sequence, yet they have to operate under severe constraints. First, the sequence is short and so even with full supervision, the sample size for training is small, ranging between one and a few hundreds object instances. Second, the demand for real time (or otherwise efficient) response renders full training of complex models prohibitive. Third, and most complex: tracking poses an unsupervised learning problem (with the exception of the first frame), where supervision for

training at step i is obtained via bootstrapping - successful inference of the model from step i 1. It is therefore highly challenging to keep supervision integrity, especially when the target or background appearance changes rapidly and inference is not perfect. In our work, we adopt the online tracking approach, and our specific choices are derived as answers to the three difficulties mentioned here. The small data set limits online learning to simple classifiers (a small hypothesis family), so linear classifiers are a natural choice. Among the possible algorithms for learning a linear classifier, the efficiency and online constraints suggest LDA as the natural linear classifier choice. Finally, the need for supervision integrity motivates two additional choices: the usage of an ensemble of classifiers, providing supervision for each other, and posing the problem as pixel level classification, thus avoiding the object:background confusion introduced by using a rectangular bounding box.

Traditional trackers such as Lucas-Kanade [21], [22], as well as modern correlation filters, learn a rigid spatial object template, and use it to infer a rectangle bounding the target object. With this approach, near-object background pixels often become part of the object template, leading to errors when the background changes. The severity of this phenomena depends on the object shape, but certain objects, and specifically non-convex and deformable ones, are not well fitted by a bounding rectangle, and the percentage of object pixels in the rectangle may drop below 50%. An attractive alternative is to learn pixel-wise object:background classification. Traditionally simple features like color were used for this task, but the pixel-wise feature vectors (columns) of modern CNNs provide a much stronger, more discriminative alternative. These features encapsulate the shape around the pixel at multiple scales and semantic levels. Our method therefore trains linear pixel-based object:background classifiers, based on column vectors obtained from multiple intermediate tensors of a backbone CNN. Each such classifier provides an object likelihood map when applied in a search window. Fusing these 'objecthood maps' provides a pixel-wise segmentation of the target, which is used as supervision for model update. The merits of the method depend on the choices of classifiers and fusion method, which are discussed next.

As a linear classifier we use LDA - Fisher's Linear Discriminant Analysis [23], which is most suitable for fast online learning. An LDA classifier has a closed-form formula, only depending on class means and a joint covariance matrix as sufficient statistics, and these can be easily updated online. Efficient algorithms for LDA computation exist, with low complexity. For example, in [24] an algorithm is presented with complexity of $O(d^2)$ and linear in n, where d in the dimension and n the sample size. In contrast, convex formulations like SVM or logistic regression require solving an optimization problem with iterative algorithms. Empirically, LDA classifiers applied to CNN columns achieve remarkable generalization across specific features like color or even object's viewpoint. In Figure 1, an LDA ensemble is trained on a single frame from one sequence and tested on an arbitrary selected frame

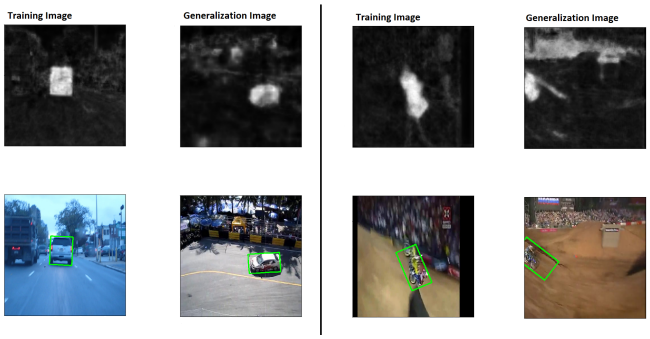

Fig. 1: LDA generalization. Ensemble of LDA classifiers trained on object in one sequence, successfully generalizes to another sequence. **Left**: Generalizing from one car sequence to another. **Right**: Generalizing from one motorcycle sequence to another. The top row shows the pixel masks obtained from the LDA ensemble.

from another sequence, with target object of the same class. As can be seen, the ensemble provides good local segmentation, insensitive of viewpoint or exact color, while learning from a single object.

We train a diverse ensemble of pixel object/background classifiers from several CNN intermediate tensors, with each tensor up-sampled to the original input resolution. Classifiers based on different layers are complementary in several ways. Low level features, which have high inherent resolution, provide more accurate segmentation in cases of plain motion. Higher level features are important for coping with severe appearance changes, hence providing robustness. From a learning perspective, high level representations have higher dimensions, hence the respective classifiers are prune to overfit at initial sequence frames. Lower layer features have a different disadvantage of being less discriminative, so sometimes it is difficult to separate object and background using a single linear classifier. We hence train multiple LDA classifiers from these layers, where each classifier is trained with a different background model, and separates the object from a different region of the background.

The fusion of objecthood maps into a final segmentation mask is done in two stages: computation of a combined objecthood map, and then choosing a binary object segmentation mask by optimizing a scoring function with several terms. One segmentation quality term measures the average gradient magnitude on the mask boundary, creating a preference for sharp objecthood demarcation. A second confidence term demands high objecthood probability of the combined classifier inside the mask. In addition, the score includes terms enforcing tracking continuity by preferring small translation and scale changes w.r.t the mask in the previous frame. The score is optimized with an iterative process in which a proposal mechanism suggests candidate masks, and an evaluation module evaluates their scores.

The proposed method achieves highly competitive tracking results on the VOT2016 [25] and VOT2018 datasets [26], and

competitive results on the VOT2019 dataset [27]. Its main strength point is the high robustness obtained by incorporating segmentation information into an online pixel classification training. We also consider its video segmentation quality on DAVIS2016 [28] dataset, and conduct an extensive ablation study to expose the contribution of method's components.

To summarize our contribution, it consists in

- We suggest a tracking algorithm based on *online-only learning of a pixel-wise object:background classifier*. This is the first tracker enabling on-line segmentation learning in the modern CNN era.
- The algorithmic enablers of this tracker are developed, including online LDA classifiers applied at multiple layers, an adaptive fusion mechanism, and segmentation-based score optimization.
- The competitive empirical results show that this strategy, which is significantly different from most currently-leading techniques, provides a viable robust tracking alternative.

We next discuss related work in section II and present the suggested method in detail in section III. Experimental results are then presented in section IV, followed by concluding remarks in section V.

II. Related Work

For a comprehensive review of existing tracking methods the reader is referred to recent surveys [29]–[34]. We here provide a brief overview of recent tracking families, highlighting the main differences from the suggested method and focusing on algorithms with nevertheless similar notions. In recent years several robust tracking strategies have emerged in the Visual Object Tracking challenges [25]–[27], with two main approaches prevailing: The first includes online-learning of spatial correlation filters [11]–[18], which are applied to various feature maps, including deep features [11]–[15]. The second is focused in deep learning, with or without post processing [5]–[10], [35]–[39]. Deep learning based methods can be further divided into two main groups: Methods relying only on offline learning with large annotated videos dataset [5]–[10], and methods including online model update, often combined with offline learning [19], [35]–[40].

Correlation trackers are part of the tracking-learning-detection [41] paradigm. They find the optimal displacement of a spatial template between consecutive frames. To do that, a set of spatial correlation filters is used, whose coefficients are updated as tracking progresses. Filter optimization is done by L_2 minimization, with efficient techniques based on working in the frequency domain. The exact function to minimize varies between different trackers. The filters are applied to a wide variety of feature maps, including gray level [17], HOG [42] and neural net features [15]. KCF [16] introduced the concept of kernels to the line of correlation based trackers. While our method employs pixel-wise classification and learns the weighting of different channel features, these methods learn full rigid spatial filters, usually of three dimensions. Among them, the CSRDCF [18] method is closer to our ideas, as it uses a weighting strategy for each feature channel at the decision stage.

Offline deep learning methods are predominantly Siamese trackers. They employ Siamese networks and use correlation layers to find the relative displacement of the target. Since no online learning is done, such methods have an important advantage of high tracking speeds. However, they suffer from under-utilization of the information present in the tracked sequence, sometimes resulting in weaker handling of distractors [36]. These methods require training on large video annotated datasets, which are not always available for a given tracking scenario or a specific real world application. In contrast, the method proposed here can be easily adapted to new CNN models trained on image classification datasets, by replacing the CNN backbone used. Most of these methods do not attempt to segment the tracked object. Exceptions are SiamMask [8], and Siam R-CNN [10], where the tracker is pre-trained to output not only the object location, but also its segmentation mask. However, no attempt is made to use the segmentation for learning relevant features online. In Section IV-B, we show that this leads to inferior robustness compared to the suggested method. The recently proposed D3S tracker [39] employs two offline trained sub networks for segmentation and tracking, that are adapted online. It is similar to our method in its usage of segmentation to help tracking and vice versa, but these two flows are in essence independent, whereas in our method tracking and segmentation are inter-dependent.

When CNNs are used in trackers employing online learning, a main challenge is that full CNN training is too expensive computationally for typical tracking applications. Somewhat similar to our method, TCNN [19] tracker uses an ImageNet-trained backbone network for feature extraction, and performs online learning applied with an ensemble of models. However, the models in their case are three-layer neural networks kept in a tree structure, and online learning is done with standard SGD, which is computationally expensive. Our online models are much simpler (linear), and efficiently trained with LDA. Methods combining online model update with offline learning from video sequences [35], [36], [38]–[40] were developed in recent years in parallel to the Siamese trackers. Trackers like MDNet [38] and RankingT [37] train a full network offline, but retrain a separate classification head online for each tracked sequence. Trackers like ATOM [35] and DiMP [36] separate between estimation of the object's bounding box, which is trained offline, and a classification network trained online in order to reduce errors due to distracting objects. To keep the online training efficient enough, conjugate gradient optimization is suggested.

III. Method

We start with describing the tracking step and tracking dynamics in general in section III-A, and detail the algorithmic components in the following sections. The CNN backbone is described in section III-B. The LDA classifiers used for spatial location classification are described in III-C. The hierarchical

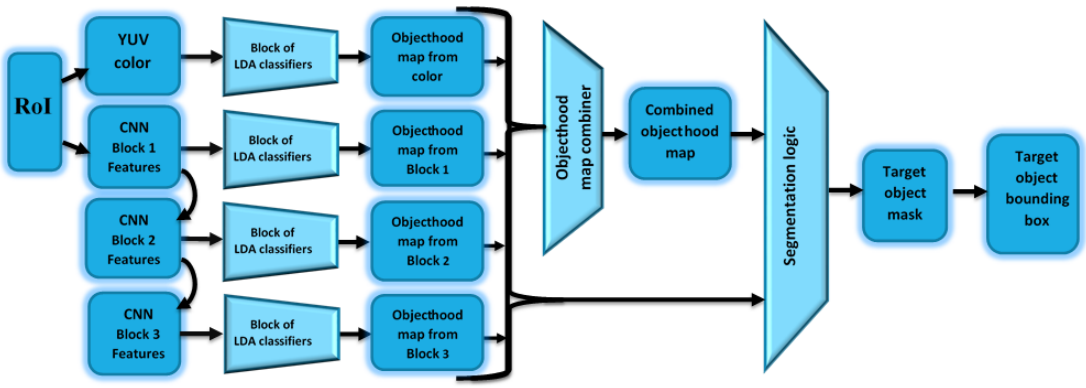

Fig. 2: Tracking step layout. An ROI is processed by several CNN blocks, and each location in their output is submitted to one or more LDA classifiers (LDA operation can hence be done as a $1 \quad 1$ convolution). The LDA outputs are combined to an object probability map, and a segmentation mask is chosen to optimize a segmentation score defined based on it and the individual objecthood maps. The binary mask provides the labels for LDA updates, and its bounding box is the tracking output. See section III-A for details.

fusion of the LDA classifiers into a single ensemble is described in III-D. The segmentation mask optimization based on the ensemble maps is described in III-E.

A. The general framework

Given a conjectured object mask in frame $t \quad 1$, a tracking step is applied to a Region of Interest (ROI) Z^t around it to find its position in frame t. The tracking step layout is presented in Figure 2. The ROI is resized to a fixed $W \quad H$ size, and is propagated through a CNN backbone. The outputs of several CNN blocks are up-sampled to the spatial dimensions of the input ROI, and each position column is fed into one or more linear classifiers. A different set of linear classifiers is maintained for each block. In order to enrich the representations, the ROI is also converted to YUV color space and fed into another block of linear classifiers. Each linear classifier is binary, trained to discriminate between object and background pixels. Applied to each spatial position, each block of linear classifiers produces an objecthood score map. The block objecthood maps are then fused linearly, yet with adaptive weights, to form a single combined objecthood map. Following the fusion, the fusion weights are adjusted based on the block's agreement with the combined consensus map. Blocks whose maps align well with the consensus have their weights enhanced, and vice versa.

Following the fusion, a mask tracking score is defined with four terms. The correct object mask is expected to have a sharp objecthood gradient at its boundary, where object probability is expected to drop, and a high objecthood probability for at least some of the mask pixels. A score reflecting these demands, as well as preference for small position and scale adjustments is formulated and optimized by scoring various mask propositions. The propositions are created by considering various thresholds applied to the combined and the block-specific objecthood maps. The labels of the final mask are then used to update the LDA classifiers in all blocks for the current frame. Finally, an output bounding box is defined based on the mask, as well as a new ROI for processing the next frame.

B. The CNN backbone

A general layout of the CNN backbone structure used is presented in Figure 3. Most modern CNNs [43]–[47] are characterized as a sequence of blocks, with each block processing the input image at a fixed spatial resolution. The transition between two consecutive blocks is an operation reducing the spatial resolution by a $2 \quad$ factor (pooling or convolution with stride 2). Usually an ImageNet trained CNN includes 5 such blocks. Our layout, presented in Figure 3 for a general network uses the output of the first $p = 3$ blocks. These tensors are upsampled to the resolution of the input ROI using bi-linear interpolation. Thus, when a linear LDA classifier is applied location-wise (as a $1 \quad 1$ convolution), an object probability map with the ROI resolution is obtained from each such tensor. The resulting structure is similar to an hourglass [48] type CNN, but with plain upsampling layers replacing the deconvolution operator.

The CNN backbone structure as stated is defined in general block terms, and can be implemented with any existing (or future) CNN with a block structure. In practice we use a CNN backbone based on DenseNet121 [45] in our main experiment (we have also experimented with VGG [43], see section IV). In DenseNet, each convolutional layer is connected directly to all the preceding layers in its block. To keep the computation burden manageable, each layer computes a relatively small number of feature maps, and feature maps from all the layers in the block are concatenated for the final block output. The reason this architecture was chosen is that contrary to other architectures, the output tensor provides direct access to features from every layer in the block.

Assuming the ROI processed in time t is of size $H \quad W$, the output tensor of block $p = 1;2;3$ are denoted by $T_p^t \; 2$

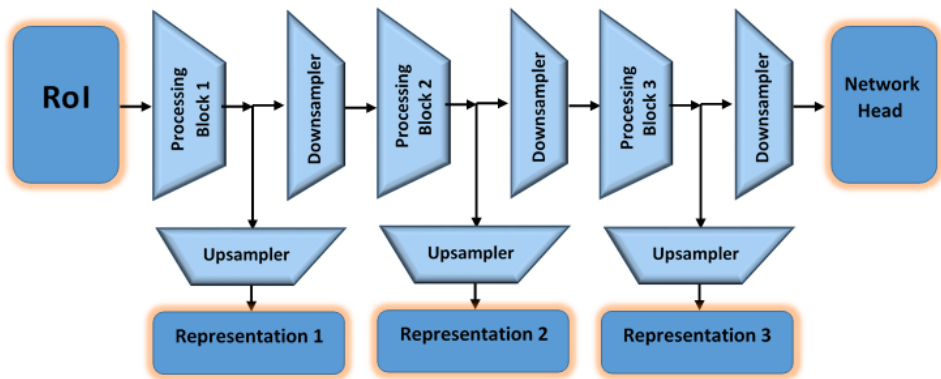

Fig. 3: A CNN Backbone structure used by the tracking method. A CNN block is a sequence of layers culminating in a 2 spatial resolution reduction. The representations created by the first three blocks are up-sampled to the original ROI resolution.

$R^{H \quad W \quad D^p}$. The color representation $T_0^t \; 2 \; R^{H \quad W \quad 3}$ is added to these representations, for a total of 4 representation spaces.

C. LDA classifiers

LDA location-wise classifiers are the main component of our method, as they are very efficient in online training and inference. We start by providing a short reminder about binary LDA classification. Next we discuss how LDA is applied to object versus background pixel classification. Specifically online training, regularization and normalization are discussed, which are critical for obtaining reliable probabilistic classifiers.

Binary LDA classification: Assume a binary hidden variable of interest $y \; 2 \; f \quad 1; 1g$ and an observed continuous feature vector $x \; 2 \; R^d$ from which y is to be inferred. We make two simplifying assumptions

1) The class conditional distributions $P(xjy \; = \; 1)$ and $P(xjy = \quad 1)$ are multivariate Gaussian.
2) $P(xjy = 1), P(xjy = \quad 1)$ differ in their mean parameters $\quad {}_1; \quad {}_1$, but have the same covariance matrix $\quad$.

Given these assumptions, it can be shown [49] that the log likelihood ratio has a linear form

$$LLR(x) = log \frac{P(xjy = 1)}{P(xjy = \quad 1)} = w \quad x \quad b \tag{1}$$

with

$$\begin{aligned} w &= \quad {}^1(\; {}_1 \quad {}_1) \\ c &= \quad \tfrac{1}{2}(\; {}_1^t \quad {}^1 \; {}_1 + \; {}^t_1 \quad {}^1 \quad {}_1) \end{aligned} \tag{2}$$

The sign of $LLR(x)$ hence indicates the more likely class value (positive values entail $y = 1$ and vice versa), and the class probability is given by the sigmoid operation

$$p(y = 1jx) = \frac{1}{1 + e^{\quad LLR(x)}} \tag{3}$$

In practice, often the covariance matrices of classes $1; \quad 1$ are not the same, i.e. $\quad {}_1 \; \cancel{6} = \quad {}_1$, and assumption 2 is false. In this case LDA is not guaranteed to provide the optimal classifier, yet it can be used as an approximation with an averaged covariance matrix. Assuming $\quad {}_1; \quad {}_1$ has been estimated from samples of size $N_1; N \quad {}_1$ respectively, the maximum likelihood estimate for a joint $\quad$ matrix is

$$= \frac{N_1}{N_1 + N \quad {}_1} \quad {}_1 + \frac{N \quad {}_1}{N_1 + N \quad {}_1} \quad {}_1 \tag{4}$$

online training: In the tracking-segmentation task we solve, we learn LDA classifiers that discriminate between object and background pixels, so we denote the two classes of interest by $o; b$ subscripts instead of $1; \quad 1$. For a representation level $p \; 2 \; f0; 1; 2; 3g$, pixel $(i; j)$ in time t is described by the features vector of the form $x_{i;j}^{t;p} \; = \; T_p^t[i; j; :] \; 2 \; R^{D_p}$. A basic LDA classifier in our system is trained to discriminate between pixel vectors $x_{i;j}^{t;p}$ describing object pixels and those describing a set of the background pixels. Denote the current frame index by T. When the LDA classifier is updated, objecthood labels for pixels in frames $t = 1; ::: ; T$ were already inferred by the algorithm - see section III-E for the details.

In this context, the main advantage of the LDA classifier is that it depends solely on the sufficient statistics $N_i; \wedge_i; \wedge \; {}_i$ for $i = o; b$, i.e. the pixel count of the class so far, and the empirical estimations of the first and second class centralized moments. Specifically, denote by $N_i^{1:T \quad 1}; \wedge_i^{1:T \quad 1}; \wedge_i^{1:T \quad 1}$ the sufficient statistics summarizing time $t = 1; ::: ; T \quad 1$, and by $N_i^T; \wedge_i^T; \wedge_i^T$ the sufficient statistics of pixel vectors in frame T. The sufficient statistics can be updated by plain summation and averaging:

$$\begin{aligned} N_i^{1:T} &= N_i^{1:T \quad 1} + N_i^T \\ \wedge_i^T &= \frac{N_i^{1:T-1}}{N_i^{1:T}} \wedge_i^{1:T \quad 1} + \frac{N_i^T}{N_i^{1:T}} \wedge_i^T \\ \wedge_i^T &= \frac{N_i^{1:T-1}}{N_i^{1:T}} \wedge_i^{1:T \quad 1} + \frac{N_i^T}{N_i^{1:T}} \wedge_i^T \end{aligned} \tag{5}$$

Following sufficient statistic update, one can use Eq. 4 for obtaining the joint covariance matrix estimate, and then Eq. 2 for obtaining the updated classifier.

Covariance regularization: In some cases the matrix $\quad$ which is inverted in Eq. 2 may be ill ranked, leading to numerically unstable results. Specifically in initial frames the number of pixel vectors may be smaller than the dimension D_p (which is hundreds or thousands for higher blocks), and in

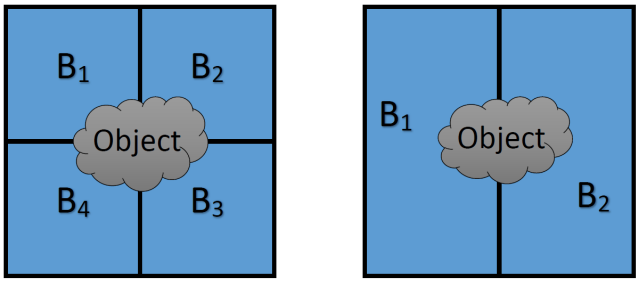

Fig. 4: Background partition for LDA classifiers inside a block. The background is divided into k equally sized disjoint sectors. **Left:** Partition with $k = 4$ sectors. **Right:** $k = 2$ sectors.

such cases the matrix is of low rank and un-invertable. To overcome such difficulties, a common practice is to stabilize the classifier output by regularizing the matrix

$$_{reg} = (1 \quad _r) \quad + \quad _r \quad trace(\quad) \quad I \tag{6}$$

Where $_r$ is a predefined small constant. This regularization is equivalent to a Maximum A-Posteriory (MAP) estimation of the covariance under an isotropic prior with a fixed strength. This regularized version of is used in equation (2).

Confidence normalization: The LDA classifiers are not used in isolation - they are fused instead into an ensemble, and successful fusion requires reliable confidence estimation, beyond mere classification. The LLR score of Eq. 2 has a natural confidence interpretation, as $p(y = 1jX)$ can be obtained from it using Eq. 3. However, due to the high dimension D_p, the LLR score is over-confident: it is based on an inner-product whose absolute magnitude scales with D_p (see [50] for a similar problem in attention models), and leading to hard $f0; 1g$ estimated probabilities. We found empirically that normalization the LLR by its standard deviation provides stable and useful confidence scores. The standard deviation estimate itself should be stable across frames, so we update it with a running average formalism. Denoting by ^{T} the standard deviation of LLR scores across pixels in frame T, a stable estimator $^{1::T}_d$ is updated using

$$^{1::T}_d = \quad _d \quad ^T + (1 \quad _d) \quad ^{1::T \quad 1} \tag{7}$$

With $_d \; 2 \; (0; 1]$ a smoothing parameter. A normalized score $LLR_n(X) = \frac{LLR(x)}{^{1::T}_d}$ is then used for objecthood probability estimation via Equation 3.

D. Ensemble construction

LDA classifiers can be maintained efficiently, yet they are limited by their linear functional form. In order to obtain a powerful classifier, hierarchical ensemble construction is required. First, for a fixed block p, separation between object and background pixel vectors $x^p_{i;j}$ with a single linear classifier is sometimes not possible, specifically if the dimension D_p is low. Hence we partition the background area, for a single block p into disjoint spatial sectors and learn a set of LDA classifiers discriminating the object from different cells of the background partition. These classifiers are then fused into a single classifier for block p. Following that, classifiers of different blocks, are fused into a single global classifier. We next discuss the details of these two levels: inner-block fusion and between-blocks fusion.

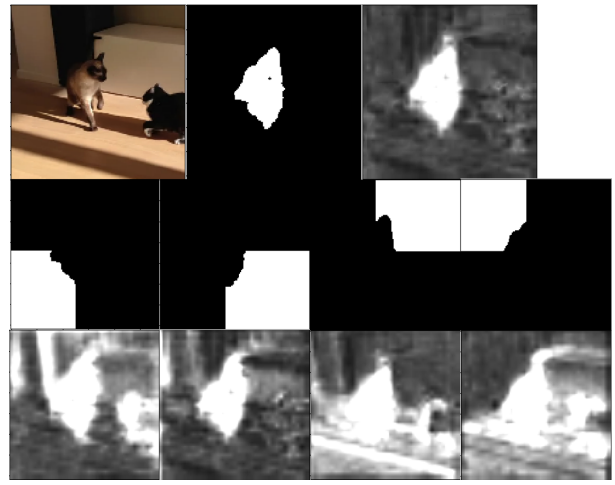

Fig. 5: Inner block spatial LDA ensemble. The example shows a 4-LDA classifiers ensemble trained from a single image at block $p = 1$. **Top Left:** An object (cat) image. **Top Middle:** the object mask. **Top Right:** The objecthood map $p(y = o : x)$ predicted by the trained ensemble. **Middle row:** the background masks defining the 4 background sectors. **Bottom row:** Objecthood maps predicted by the LDA components.

Inner-block LDA ensemble: For a fixed representation space $p \; 2 \; 0; 1; 2; 3$, we train K^p LDA classifiers. The tracking ROI is chosen with the object in its center, and the the K^p classifiers are trained to discriminate between object pixels and background pixels of a certain ROI sector. See Figure 4 for a visual sector demonstration. The LDA parameters of each classifier are updated independently. Denote by $\hat{p}^p_k(y = ojx)$ the objecthood probability obtained from LDA classifier k via Equation 3. If a pixel cannot be separated from background in one of the sectors, it is probably not an object pixel. We hence fuse the K^p classifiers confidences into a single objecthood confidence $\hat{p}^p(y = ojx)$ using the minimum operation

$$\hat{p}^p(y = ojx) = min_k \hat{p}^p_k(y = ojx)] \tag{8}$$

An example for the need of this procedure is shown in Figure 5. Note that each LDA component discriminates well background pixels in its training sector (in this sector $\hat{p}(y = ojx)$ is low and so it appears darker in the objecthood map), but may fail on background pixels from other sectors (which hence appear brighter). The fused ensemble, taking the minimum over the 4 maps, segments the object well.

Between-block ensemble: Different LDA blocks are complementary in several respects. Blocks of the lower CNN features accurately model the boundaries of the object, but fail to distinguish between the object and a distractor from a different semantic class. LDA blocks of the higher CNN features are

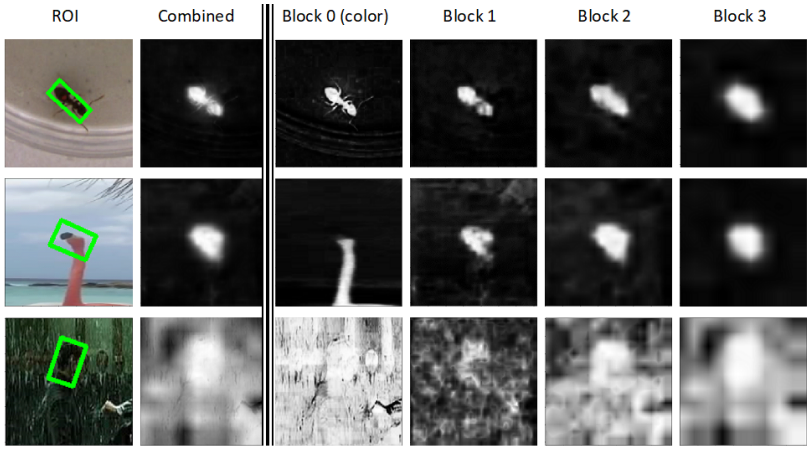

Fig. 6: Combined objecthood map calculation using an adaptive mixture of experts. In each row, a different frame example is displayed. The ROI is displayed on the left, and next to it the objecthood map of the combined classifier. Next, the objecthood maps of blocks $0; 1; 2; 3$ are shown. **Top:** The combined objecthood map is dominated by the color block, **Middle:** Domination of network block 2. **Bottom:** Domination of network block 3.

more robust to distractors, but fail to catch the boundaries and the localization of the target object. In addition, due to the different dimensions D_p, object classifiers of different blocks train at different rates, with low dimensional blocks training fast and high dimensional ones tending to overfit with small sample and hence train slower.

We regard each block classifier as an 'expert' and use a mixture of experts model for their fusion. They are combined with linear weights, but the weights are time-dependent. At time t the combined objecthood estimator is given by

$$\hat{p}_t(y = o|x) = \sum_{p=0}^{3} w_t^p \hat{p}_t^p(y = o|x) \tag{9}$$

with $w_t^p \in [0; 1]$ mixture weights satisfying $\sum_{p=0}^{3} w_t^p = 1$. The mixture weights $\{w_t^p\}$ are very dynamic, and their value at time t is determined based on consistency of the expert with the combined decision at time $t-1$, as next described.

At time $t-1$, the expert consistency is judged by comparing its segmentation mask to the final segmentation mask. For an expert p, its segmentation mask is $E^p = \{x : p^p(y = o|x) > 0.5\}$, i.e. the set of likely object pixels according to the expert. The final object segmentation mask S is a second pixel set, obtained by optimizing the score described in section III-E. If we consider E^p as a set suggested for the detection of pixels in S, recall and precision can be naturally defined by $R = \frac{|E^p \cap S|}{|S|}$ and $P = \frac{|E^p \cap S|}{|E^p|}$. The F_1 statistic for measuring the detection performance of expert p is

$$F_1^p = \frac{2RP}{R + P} = \frac{2|E^p \cap S|}{|E^p| + |S|} \tag{10}$$

Intuitively, a high F_1^p indicates that expert p is a good predictor of the final mask S, and can be considered reliable for the next time step. The expert weights are hence determined by a softmax transformation of the F_1 scores with an inverse-temperature hyper parameter α_f

$$w_t^p = \frac{\exp(\alpha_f F_{1;t-1}^p)}{\sum_{q=0}^{3} \exp(\alpha_f F_{1;t-1}^q)} \tag{11}$$

Note that while the LDA classifiers change smoothly as tracking persists, the dynamics of the expert weights is not smoothed: it may abruptly change between two frames $t-1; t$ if a certain expert provides a much better prediction at frame $t-1$. This provides the tracker with an ability to immediately switch between lower blocks providing better segmentation (for 'easy' frames with appearance continuity) to higher blocks providing robust object discrimination (for 'difficult' frames requiring object semantic identity), and vice versa.

While the proposed mixture performs well in most scenarios, we have noticed empirically that when the color channel $p = 0$ is performing well, better accuracy is obtained if the objecthood estimation is based only on this channel. This corresponds to cases of uni-color objects with significant color difference from the background. Hence if the weight of the color channel satisfies: $w^0 > t_c$ for a threshold t_c (i.e. color is identified as dominant in successful segmentation), we set: $w^0 = 1$, ignoring all other channels.

Several cases of objecthood maps fusion are presented in Figure 6. While color tracking is dominant for unicolor object like ants, medium block features are mostly useful for tracking the flamingo head shape, and for the difficult face tracking through low SNR the highest block features become dominant, providing robustness.

E. Segmentation logic

The LDA classifiers are applied to single pixel locations, and as such, they require labeling of each position as object or background for their update. We infer these labels by defining a loss function over the object mask, i.e. the set

of pixels labeled as object. This function is minimized by evaluating a set of mask options rising from several proposal mechanisms: thresholding of the combined and CNN-based block objecthood maps, and proposals based on zero-order hold, i.e. no motion assumption. This algorithm is outlined in detail in algorithm 1. We next describe the loss, its components and the proposal mechanisms.

Object mask loss: In the following notation, we drop time superscripts wherever analysis is done in a single time frame. The ROI include a set of WH pixel positions $P = f(i;j)g_{i=1;j=1}^{W;H}$, and we wish to find an object mask, which is a pixel subset $O = fo_k g_{k=1}^{N_O} = f(i_k;j_k)g_{k=1}^{N_O} \quad P$. Applying the combined objecthood estimator $\hat{p}(y = ojx)$ and the block-specific estimators $f\hat{p}^p(y = ojx)g$ over the ROI one gets objecthood maps of size $W \quad H$ denoted by $M^c; M^p$ respectively. As will be detailed later, candidate masks O are obtained by thresholding one of the maps $M^c; fM^p g_{p=1}^3$, so we can associate with each proposal O a map index $q(O) \ 2 \ f1;2;3;cg$. For a mask proposal O we compute the center of mass $C(O) = \frac{\sum_{p=0}^{N_0}(i_k;j_k)}{N_0}$, and the bounding rectangle with minimal area $R(O)$. The minimal area rectangle is not axis aligned - it is a rotated rectangle in the general case. However, it can be computed by finding the convex hull of the mask and applying the rotating calipers algorithm [51], [52] with complexity linear in N_O. Denote by $W(O); H(O)$ the width and height of the minimal area bounding rectangle $R(O)$.

The loss defining an object mask quality contains four terms:

$$L(O) = \ _g L_g(O) + \ _c L_c(O) + \ _t L_t(O) + \ _s L_s(O) \quad (12)$$

The first term $L_g(O)$ quantify the segmentation quality by demanding high object probability gradient on the object boundaries. This is the most important term (see VI for an ablation study), demanding that the object mask will have natural demarcation in a probability map. The second, $L_c(O)$ demands high object likelihood for some of the mask pixel's, and is required to avoid the choice of a well segmented distracting object. The terms $L_t(O); L_s(O)$ are imposing a 'slow and smooth' motion prior [53] by demanding continuity of translation and scale changes respectively. The coefficients $_g; \ _c; \ _t; \ _s > 0$ are hyper parameters chosen empirically. The terms are defined as follows:

Gradient loss $\mathbf{L_g(O)}$: For each pixel location o_k we consider a 4 $pixel$ neighborhood. Defining

$$= f \ _r g_{t=1}^4 = f(0;1);(0; \quad 1);(1;0);(\quad 1;0)g \quad (13)$$

the neighbors of a location o_k are $fo_k + \ _r g_{r=1}^4$. We define the set of all (object, background) neighbors by

$$B = f(o_k; o_k + \ _r) : o_k \ 2 \ O; \ _r \ 2 \quad ; o_k + \ _r \ 2 \ Og \quad (14)$$

Enumerating $B = f(b_i^1; b_i^2)g_{i=1}^{N_B}$, we expect that in a good segmentation each pair b_i^1, which is an object pixel, would have a much larger objecthood probability than b_i^2, which is a background pixel. Such a sharp boundary should exist in the objecthood map from which O was computed. We hence wish the average gradient:

$$g(O) = \frac{1}{N_B} \sum_{(b^1;b^2) \ 2 \ B} M^{q(O)}(b^1) \quad M^{q(O)}(b^2) \quad (15)$$

to be high as possible, with a value of 1 indicating a perfectly confident segmentation. In practice the average gradient $g(O)$ is not approaching 1 in most cases, as the object probability map is far from perfect, and so we minimize:

$$L_g(O) = (max(1 \quad g(O); 0))^q \quad (16)$$

Where $q > 1$ is a hyper parameter. Higher value of q entail faster decrease of the loss with rising $g(O)$, and enable approaching 0 loss for imperfect segmentation.

Confidence loss $\mathbf{L_c(O)}$: This score demands high object probability in the final object probability map M^c. Denote by $(M_{v_1}; ...; M_{v_{K_c}})$ the K_c highest values among the mask objecthood scores $M_O^c = fM^c(o_k) : o_k \ 2 \ Og$. The confidence score is:

$$L_c = (1 \quad \frac{1}{K_c} \sum_{k=1}^{K_c} M_{v_k}) \quad (17)$$

While we wish at least some of the locations to have high objecthood confidence, we do not demand it from all mask locations, and the hyper parameter K_c enables moving between maximum and average confidence. Empirically, low K_c values were found beneficial.

Translation loss $\mathbf{L_t(O)}$: We enforce preference for smaller translation using a log-laplacian prior:

$$L_t(O) = \ \frac{kC(O^t) \quad C(O_{min}^{t \ 1})k}{\sqrt{W(O_{min}^{t \ 1})^2 + H(O_{min}^{t \ 1})^2 + \ _t}} \quad (18)$$

with $_t > 0$ a small constant added for numerical stability.

Scale loss $\mathbf{L_s(O)}$: For scale deviations, a natural variable to consider is the scale ratio between the current and previous frame in log scale. We punish scale deviations with a log-Gaussian prior imposed over this quantity:

$$l_s(O) = \ log \frac{H(O) \ \ W(O)}{H(O_{min}^{t \ 1}) \ \ W(O_{min}^{t \ 1})}^{\ 2} \quad (19)$$

Mask proposal system: Multiple hypotheses are evaluated to find the one with the minimal loss. A baseline hypothesis is obtained using a zero-order hold assumption, but the main hypothesis set is obtained by thresholding the maps $M^c; fM^p g_{p=1}^3$. The zero order hold hypothesis keeps the object mask $O^t = O_{min}^{t \ 1}$, and hence also the bounding rectangle. However, to estimate its objecthood gradient support term $L_g(O)$, the gradient support (Equation 15) is computed for each map index $q \ 2 \ f1;2;3;cg$, and the minimal value is taken. Thus this hypothesis gets a good score if the previous mask enjoys gradient support in any of the feature maps.

The main proposal set is generated by applying a set of thresholds $T = ft_1; ...; t_{N_T}g$ to the objecthood maps $fMg \ [\ fM^p g_{p=1}^3$. For each map and threshold, connected component analysis is performed on the resulting location

Algorithm 1 Segmentation Logic

Input: Maps $\{M\} \cup \{M^p\}_{p=1}^{3}$, Rectangle $R(O_{min}^{t-1})$, threshold set $T = \{t_i\}_{i=1}^{N_T}$
Output: Object mask O_{min}^{t}
Init: $minLoss = \infty$
for map in Maps **do**
  $O_{ZOH} = O_{min}^{t-1}$
  $loss \leftarrow L(O_{ZOH})$ ▷ Using Eq. 12 with $q = map$
  if $loss < minLoss$ **then**
    $O_{min}^{t} \leftarrow O_{ZOH}, \quad minLoss \leftarrow loss$
  for t in T **do**
    S $\leftarrow$ $\{p : map(p) > t\}$
    $B = \{b_i\}_{i=1}^{N_B} \leftarrow$ connectedComponents(S)
    for blob b_i in B **do**
      $loss \leftarrow L(b_i)$ ▷ Eq. 12 with $q = map$
      if $loss < minLoss$ **then**
        $O_{min}^{t} \leftarrow b_i, \quad minLoss \leftarrow loss$

set, and every connected component blob is considered as an object proposal. Therefore, a total number of at least PN_T proposals are evaluated. This tactic produces a large number of proposals. Empirical observations show that considering more proposals improves tracking performance, yet at the cost of additional computational complexity. To ignore small blobs resulting from noise artifacts only blobs larger than 1% of the object size in the previous frame are considered.

The output rectangle of the algorithm in time t is $R^t = R(O_{min}^t)$. The ROI for frame $t+1$, Z^{t+1}, is an enlarged axis-aligned rectangle around O_{min}^t. Denote the axis-aligned bounding rectangle of O_{min}^t as $R_{al}(O_{min}^t)$.

$$R_{al}(O) = [x_1; y_1; x_2; y_2] = [\min_k i_k; \min_k j_k; \max_k i_k; \max_k j_k] \tag{20}$$

We define a margin:

$$a = \ _m \ max(x_2 \quad x_1; y_2 \quad y_1) \tag{21}$$

with $_m > 1$ a multiplicative hyper parameter. The new ROI is given by:

$$Z^{t+1} = R_{al}(O_{min}^t) + [\ a; \quad a; a; a] \tag{22}$$

IV. Experimental validation

We discuss implementation details and hyper parameters at section IV-A, and present results on VOT tracking benchmarks in IV-B. Results on the DAVIS video segmentation task are reported in section IV-C. Finally, an ablation study measuring the contribution of various algorithm components is reported in IV-D.

A. Implementation details

The method was implemented in python using keras+tensorflow backend for network construction. The values used for the algorithms hyper parameters are presented in Table I. We use the three ($P = 3$) lower blocks of DenseNet121 as the network backbone. Our initial experiments indicated that using three blocks provides a good trade-off between the semantic representation needed to model the target object and the spatial resolution needed for accurate tracking. We found that the LDA scores can be normalized only based on the current standard deviation. Hence, we set $_d = 1$ in Equation 7. Since the loss terms were built to scale mostly in $[0; 1]$, there was no need to carefully tune most of them, so a value of 2 was used for the segmentation term, which we consider dominant, and 1 for the others. The scale coefficient baseline was $_s = \frac{1}{(2 \log 1.5)^2}$ - the value that would keep the scale penalty of equation 19 in $[0; 1]$ for re-scales of up to $1.5X$, and then we found empirically that doubling it provides better tuning.

Parameter	defined in	Value
ROI resolution	Section III-A	144×144
$_r$ covariance regularization	Equation 6	0.0001.
$_d$ momentum coefficient	Equation 7	1
$\{K^p\}_{p=0}^{3}$ #LDA classifiers	Section III-D	4,4,1,1
$_f$ fusion temperature	Equation 11	4
t_c color threshold	Section III-D	0.8
T segmentation thresholds	Section III-E	0.3:0.05:0.8
K_c #locations	Equation 17	10
q gradient loss power	Equation 16	10
$_g; \ _c; \ _t; \ _s$ coefficients	Equation 12	2,1,1,3.04
$_m$ ROI margin	Equation 21	1.5

TABLE I: Hyper parameter values used. The values were kept fixed across all the experiments reported.

VOT2016			
Tracker	EAO	Accuracy	Robustness
D3S	0.493	0.66	0.131
MHIT	0.451	0.580	0.111
SiamMask	0.442	0.67	0.233
ROAMpp	0.441	0.599	0.174
SPM	0.434	0.62	0.21
ATOM	0.430	0.61	0.18
LDASegment	**0.410**	**0.504**	**0.168**
ASRCF	0.391	0.56	0.187
SiamRPN	0.344	0.56	0.302
CSRDCF	0.338	0.51	0.238
CCOT	0.331	0.54	0.238
TCNN	0.325	0.55	0.268

TABLE II: VOT2016 - Performance comparison with state of the art trackers. LDASegment is the proposed method.

B. Performance on tracking datasets

Our system was evaluated on three major short-term tracking datasets: VOT2016 [25], VOT2018 [26] and VOT2019 [27]. Each of these datasets consists of 60 sequences. Targets are annotated by rotated rectangles to enable higher localization accuracy evaluation compared to other datasets [31], [54]. In VOT evaluation protocol [55] the tracker is reset when the tracker loses the target. Performance is measured by accuracy - the average intersection over union measured over successfully tracked frames, robustness - the rate of tracking failures for a given sequence length and the EAO (Expected Average Overlap) - a fusion of accuracy and robustness into a single measure of tracking quality [56].