2016 IEEE International Conference on Pattern Recognition (ICPR)

Object-Aware Tracking

Ivan Bogun Eraldo Ribeiro
Department of Computer Sciences
Florida Institute of Technology,
Melbourne, Florida, U.S.A.
ibogun2010@my.fit.edu, eribeiro@cs.fit.edu

***Abstract*—In this paper, we address the problem of visual tracking in videos without using a pre-learned model of the object. This type of model-free tracking is a hard problem because of limited information about the object, abrupt object motion, and shape deformation. We propose to integrate an object-agnostic prior, called *objectness*, which is designed to measure the likelihood of a given location to contain an object of *any* type, into structured tracking framework. Our objectness prior is based on image segmentation and edges; thus, it does not require training data. By extending a structured tracker with the prior, we introduce a new tracker which we call *ObjStruck*. We extensively evaluate our tracker on publicly available datasets and show that objectness prior improves tracking accuracy.**

I. Introduction

Tracking objects in video is a computer-vision problem whose solution underpins many practical applications including augmented reality and video surveillance. Tracking approaches can be *model-based* or *model-free*. Model-free approaches start tracking with no information about the object, except for its initial position in an image. To pursuit the object across frames, model-free trackers must adapt to object's changing appearance as the object deforms, becomes occluded, changes shape and size. It is unsurprising that the best trackers in benchmarks [18, 27] were adaptive.

Adaptation ability of modern trackers is achieved by using semi-supervised learning algorithms, where unlabeled data (i.e., object's location in an image) is used for the training (i.e., tracker detects object then uses it for adaptation). Trackers based on various semi-supervised learning algorithms such as multiple-instance learning [2], structured support vector machine [12], Gaussian-process regression [11] were proposed. However, reliance on unlabeled data for adaptation can cause trackers to learn incorrect object appearance once it makes a single mistake - a problem known as *drift*.

Drift is addressed in many ways. Wen et al. [25] perform joint tracking and segmentation in a unified energy minimization framework. Lee et al. [20] choose the most consistent trajectory from a set of trajectories based on texture, color, and illumination. Trackers based on cascade of classifiers were also considered. The tracker [11] uses a latent variable to control its state and consists of two regressors that are fused to perform detection. The two regressors model long and short-term object appearances, so one is updated aggressively on each frame, and the other is updated conservatively. Inspired by the Atkinson-Shiffrin's model of the human memory, Hong et al. [14] cast tracking as a short and long-term memory retrieval. The short-term component adapts to the changing object appearance on-the-fly while the long-term component stores a stable object model. However, drift remains unsolved.

In this paper, we improve tracking-by-detection by including *object-agnostic* measures to it. These measures indicate the potential for an object to be in a given image location. They are known as *objectness*, and were first used for object detection [1]. The two objectness measures we use are *straddling* and *edge density*. The measures are well-suited to tracking as they are fast to compute and do not need training data. We summarize our contributions as follows: **(i)** We increase tracker's robustness to drift by adding an objectness prior that penalize locations that are unlikely to contain an object. **(ii)** We show that, because they are fast to compute, straddling and edge density can be used on *all* locations considered by the tracker, without resorting to pruning heuristics **(iii)** Based on extensive tests done on benchmark datasets by Wu et al. [27, 26] and by Kristan et al. [19], we show that our objectness prior improves various tracking metrics.

II. Related work

Our tracker is based on the structured tracker by Hare et al. [12]. It uses the structured support vector machine to build the object's appearance model in an on-line manner. The model is built using selected features such as histograms and Haar features. To run in real time, the number of support vectors is fixed. In a recent benchmark [27], the structured tracker achieved state-of-the-art results. Also recently, adaptive correlation filters gained attention after their successful application to visual tracking [4, 13, 21, 9]. Trackers based on correlation filters detect objects by finding the image location that maximizes the correlation response. After their introduction by Bolme et al. [4], correlation filters were extended to use kernels [13], and to work on multiple scales [21, 9]. Alternatively, Gao et al. [11] used Gaussian-process regression for tracking.

Model-free trackers use limited information about the object. We can make up for this limitation by providing extra cues about the object to the tracker. Saliency cues or "objectness" of the bounding box can help the tracker to detect objects. Objectness is an object-agnostic measure of how likely is the bounding box to contain any object [1]. In object-detection approaches, objectness serves two functions: to prune bounding boxes that are unlikely to contain objects and to serve as a prior for recognition on a set of bounding boxes. For a saliency detection survey, see [6]. Carreira and Sminchisescu [7] generate object proposals by doing segmentation, after which segments are ranked according to attributes such as region, graph partition, and Gestalt laws. Endres and Hoiem [10] ranks object proposals using the structured SVM with a

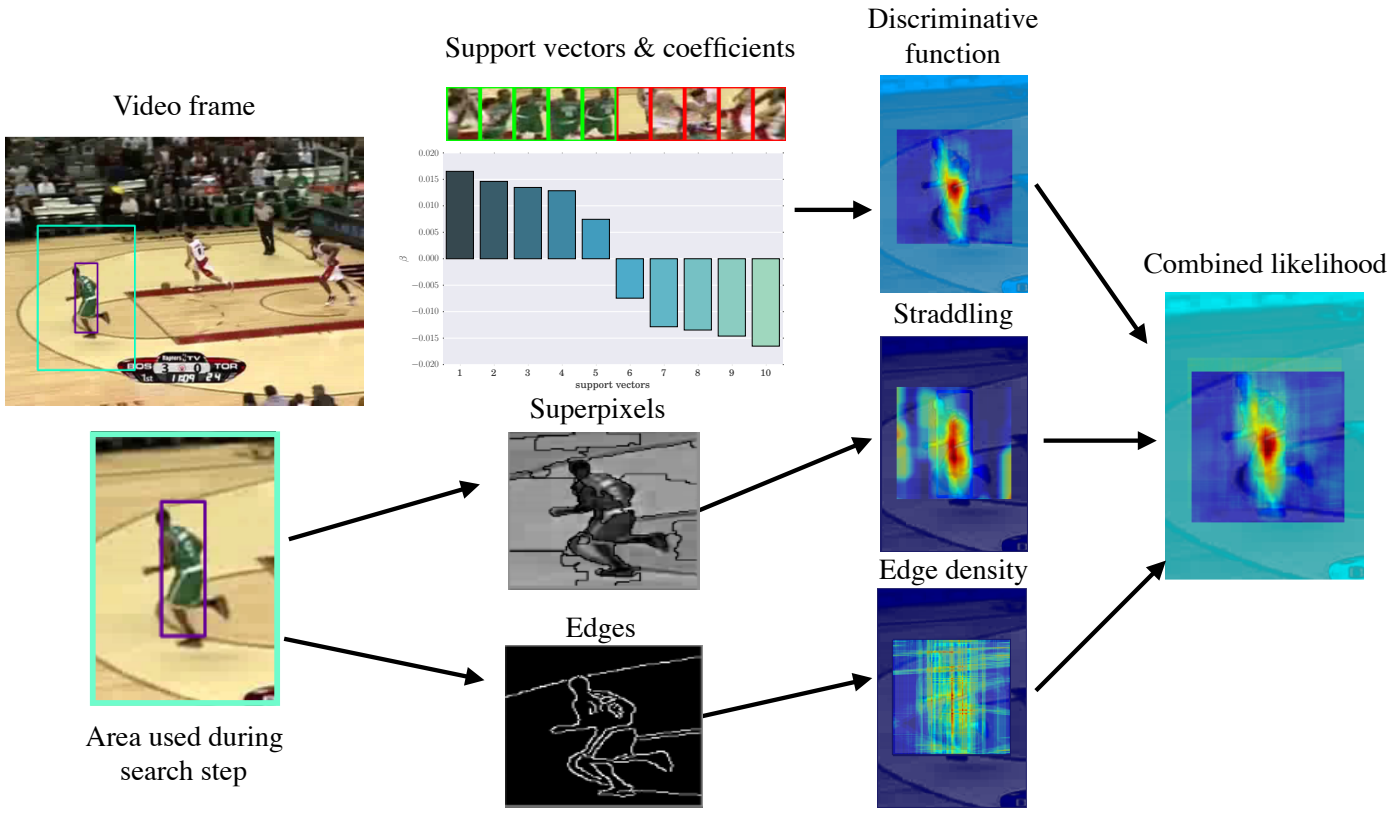

Fig. 1: Our tracker. Left column (top) shows the image with the ground-truth bounding box, while the bottom shows the part of the image that is used for the tracker's search step. The second column shows the images that correspond to support vectors and their coefficients. In the middle, the result of the segmentation and edge detection are shown. The third column shows locations where the bounding box is more likely to be based on the straddling measure and edge density. The last column shows a linear combination of the discriminative function and objectness measures.

modified loss function. While these approaches work well on object-detection benchmarks, they do not run in real-time, a common requirement in tracking.

Proposal generation can be done in real time. Cheng et al. [8] compute objectness at rates up to 300 fps. Their method produces objectness proposals from a linear SVM learned on small-size binarized normed gradients. In this paper, we use straddling and edge density - unsupervised objectness measures introduced by Alexe et al. [1]. Straddling assumes that after segmentation, bounding boxes that fully enclose segments are more likely to contain the object, unlike boxes that divide segments. Edge density assumes that objects have well-defined boundaries that are more likely to contain edges. These two measures are well suited for tracking as they are fast to compute and need no external data for training.

Including objectness in tracking is a hot topic. To detect changes in scale during tracking, Huang et al. [15] used edge boxes as a post-detection step. The tracker by Liang et al. [22] fuses object-specific structured SVM with a structured SVM based on BING features [8]. The method initializes the objectness model from the weights learned on the VOG 2007 dataset. Zhu et al. [30] re-rank sampled boxes based on a structured SVM learned using edge-box scores. Methods [15, 30] sample few bounding boxes (i.e., 200) as input for a proposal-generation step. Our method, unlike [22], uses unsupervised objectness scores. Unlike trackers introduced by [15, 30], our tracker does not filter bounding boxes when applying the objectness prior. This is possible because the objectness measures we use are fast to compute - both straddling and edge density are computed in constant time per bounding box, which allows us to add the object-agnostic prior over *all* bounding boxes sampled by our tracker.

III. Tracker

A. Structured tracker

Our work builds on the structured tracker *Struck* by Hare et al. [12]. Struck uses the structured support vector machine. The learning step is done using the OLaRank algorithm by Bordes et al. [5]. To keep computational complexity constant, the total number of support vectors (i.e., the budget) is fixed. The tracker has two steps. In the search step, the tracker samples bounding boxes around the object's predicted location and then chooses the best location that matches the appearance model. In the update step, it samples another set of bounding boxes and updates the appearance model. During the search step, our tracker samples fixed-size bounding boxes throughout the image, while varying the aspect ratio (e.g., one of the dimensions). Let $b = \{c_x, c_y, w_{\text{prev}}, h_{\text{prev}}\}$ be a bounding box

where (c_x, c_y) is the location of the bounding box's center, and (w, h) are its dimensions. Given the object's bounding box in the previous frame, bounding boxes for search and update are sampled from the following set:

$$B(m, n, R, w, h) = \{b = (c'_x, c'_y, w, h)\}, \quad (1)$$

such that $\phi \in [0, \frac{2\pi}{m}, \ldots, 2\pi]$ and $\rho \in [0, \frac{R}{n}, \ldots, R]$, with $c'_x = c_x + \rho \cos\phi$ and $c'_y = c_y + \rho \sin\phi$. Our tracker's features are a combination of histogram-of-oriented gradients (HOG) and grayscale histograms. These features are similar to those used by Hare et al. [12]. We also use fast implementation of an intersection kernel [23].

B. Robust Kalman filter

To improve the tracker's robustness to both false-positive detections and short-time occlusions, we use the Robust Kalman filter [24] with a constant-velocity model to correct the location of the best-detected bounding box. Our correction strategy is as follows. If the filter's bounding box has a small overlap (i.e., less than 50%) with the detector's, the tracker is not updated. Also, if the filter and detector disagree, we perform the search step both in the neighborhood of the tracker and in the neighborhood of the filter. This search strategy allows our tracker to recover if the detector makes a mistake. Using the robust variant of the Kalman filter is important because the standard Kalman filter's correction is heavily affected by the detector's bounding box which might be wrong. In contrast, the filter's robust counterpart allows us to identify incorrect detections and limit their influence, which helps the tracker recover subsequent video frames.

IV. Objectness measures

Object tracking is closely related to object detection. In tracking, the tracker not only has to find the best location of the object, but also it needs to find its dimensions. Solving these two problems simultaneously is hard because the tracker must also handle variations of object scale. To help guide the tracker through changes in object scale and shape, we propose to use a *objectness* measure. Objectness is a measure of how likely a given bounding box is to contain an object in an object-agnostic manner. According to Alexe et al. [1], a good objectness measure has at least one of the following properties: (i) Be sensitive to the object's boundary, and/or (ii) Capture bounding boxes that look different from its surroundings. We propose to use two objectness measures as a prior over bounding boxes in the tracker's search step. These measures are straddling and edge density.

A. Straddling

Alexe et al. [1] suggested the use of superpixel segmentation for capturing closed boundaries in bounding boxes. Their intuition is as follows. Ideally, a perfect segmentation separates object from background. As a result, the bounding box that contains such a segmentation, and does not divide other segments into large parts is likely to contain the object. The *straddling* metric captures the degree of which the bounding box is cutting segments. Formally, let $b = [c_x, c_y, w, h]$ be a bounding box, also let S be a set of superpixels after segmentation. Straddling [1] measures how much the bounding box divides the superpixels:

$$s(b) = 1 - \sum_{i \in S} \frac{\min(|i \setminus b|, |i \cap b|)}{|b|}, \quad (2)$$

where $s(b) \in [0, 1], \forall b$. For a given boundig box $b = \{c_x, c_y, w, h\}$, straddling is computed over smaller bounding box enclosed in the b. This allows to make sure that the object is completely contained within the larger bounding box, but not on the exact boundry. Mathmatically, this is corresponds to computing straddling over the bounding box parametrized via scale, θ, i.e.:

$$s_\theta(b) = s([c_x, c_y, \theta w, \theta h]). \quad (3)$$

where $\theta < 1$. Alexe et al. [1] compute straddling in time proportional to the number of superpixels. The method uses integral images and is independent of bounding-box location. We use this fast method in our implementation.

B. Edge density

Edges can serve as an objectness measure [1, 31]. Here, we use a metric based on the edges known as edge density [1]. Edge density is the fraction of pixels classified as edges in the bounding box, divided by the perimeter:

$$e(b) = \frac{\sum_{(x,y) \in Perimeter(b)} \text{edge}(x, y)}{2(w + h)}, \quad (4)$$

where $\text{edge}(x, y)$ is a binary mask with ones at edge pixels. Edge masks are computed using the Canny edge detector. As for straddling, we augment edge density with the scale:

$$e_\theta(b) = e([c_x, c_y, \theta w, \theta h]). \quad (5)$$

For a given bounding box, edge density can also be computed in constant time from integral images for horizontal and vertical edges.

V. Objectness in tracking

To include objectness in tracking, we use it as a prior in our tracker's search step. Let $b = (c_x, c_y, w, h)$ be the bounding box, and $d(b) \rightarrow [0, 1]$ be a tracker's detector function. In our case, we normalize the scores calculated using structured SVM's discriminative function to lie in the range $[0, 1]$. To find the best location in the current frame, we use a linear combination of the detector, straddling, and edge density, i.e.:

$$\hat{d}(b) = d(b) + \lambda_s s_\theta(b) + \lambda_e e_\theta(b), \quad (6)$$

$$\hat{b} = \arg\max_{b \in B} \hat{d}(b). \quad (7)$$

where $\lambda_s, \lambda_e \in [0, 1]$ and B is the set of bounding boxes from around the previous tracker's location. Objectness measures do not have explicit object-appearance information. Thus, λ_s, λ_e contribute less than the tracker's detector function. Figure 2 shows the difference between the discriminative functions of the tracker with and without objectness measures.

Our tracker is designed to be robust to both partial occlusions and false positives. Thus, if the filter's and detector's bounding boxes have small overlap, we search in the neighborhood of both. In this case, objectness priors may lead to more false positives. Consequently, if filter and detector overlap less than 50%, objectness priors are ignored.

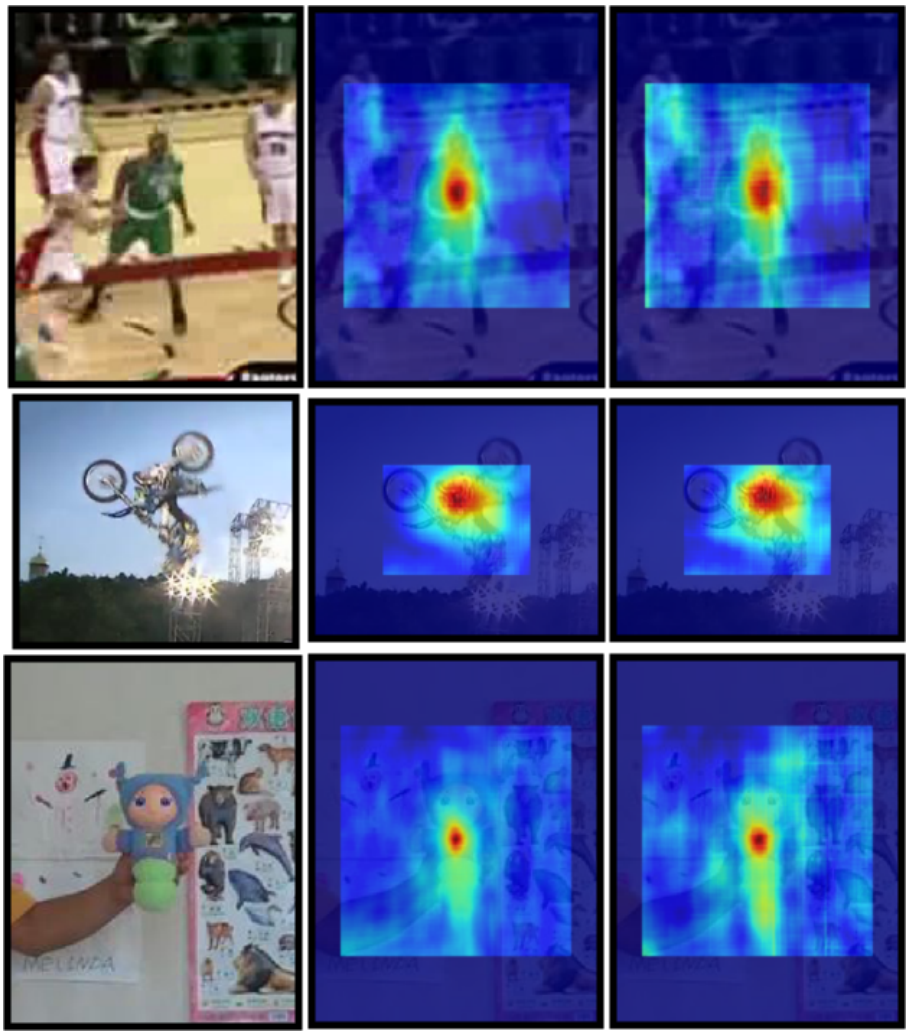

Fig. 2: Comparison of the detection functions of ObjStruck and RobStruck as a function of center translation. Left column: original image. Middle column: RobStruck discriminative function. Right column: ObjStruck discriminative function from Equation 6.

VI. Implementation

Tracker parameters We set the tracker's learning parameters as in [12], i.e., $C = 100$, $B = 100$, $n_O = 10$, and $n_R = 10$. Given a bounding box, HoG features are extracted by resizing the image so that the bounding box's height and width of are both 32 pixels. HoG parameters are: Window size is 32×32, block size is 16×16, cell size is 4×4, block stride is 16×16, and the number of bins per histogram is 8. Histograms on each level of the pyramid, L, are calculated by first dividing the bounding box into $L \times L$ cells and then concatenating resulting histograms into a feature vector. The histograms are then normalized to sum to 1. Histograms are calculated using 16 bins (i.e., we set $L = 4$). The total size of the feature vector is 992, which is given by the sum of the sizes of HoG features (512) and histograms (480).

Objectness parameters. To compute straddling, we crop an extended area around the bounding box of a size of 60 extra pixels in each direction of the bounding box. Then, superpixel segmentation [3] is done with 200 superpixels. To calculate edge density, we apply the Canny edge detector with the lower threshold equal to 0.66 times the image mean, and the higher threshold equal to 1.33 times the image mean. Parameters λ_e, λ_s are set to $0.4, 0.3$, respectively, which were chosen empirically. In our experiments, any values of $\lambda_e, \lambda_s \in (0, 0.5]$ increased tracking accuracy w.r.t. the baseline. ObjStruck runs at 1.7 frames per second (FPS) with objectness priors and at 1.75 FPS without on a single-core 2.4 GHz processor.

VII. Results

Evaluation metrics. We tested our method on benchmark datasets [27, 26, 19]. The two most-common metrics for tracking evaluation are *precision* and *success rate*. Precision measures the average distance between the track and ground truth. The ground truth is given as rectangular with known center, width and height. For frame k, precision is defined as the distance between the center of the tracked object, r_t, and the ground truth, r_{gt}:

$$p_k = ||r_{gt} - r_t||. \tag{8}$$

The precision plot, $P(\gamma)$, shows the fraction of frames where $p_t \leq \gamma$. Success rate, $S(\sigma)$, is a measure of overlap, which is

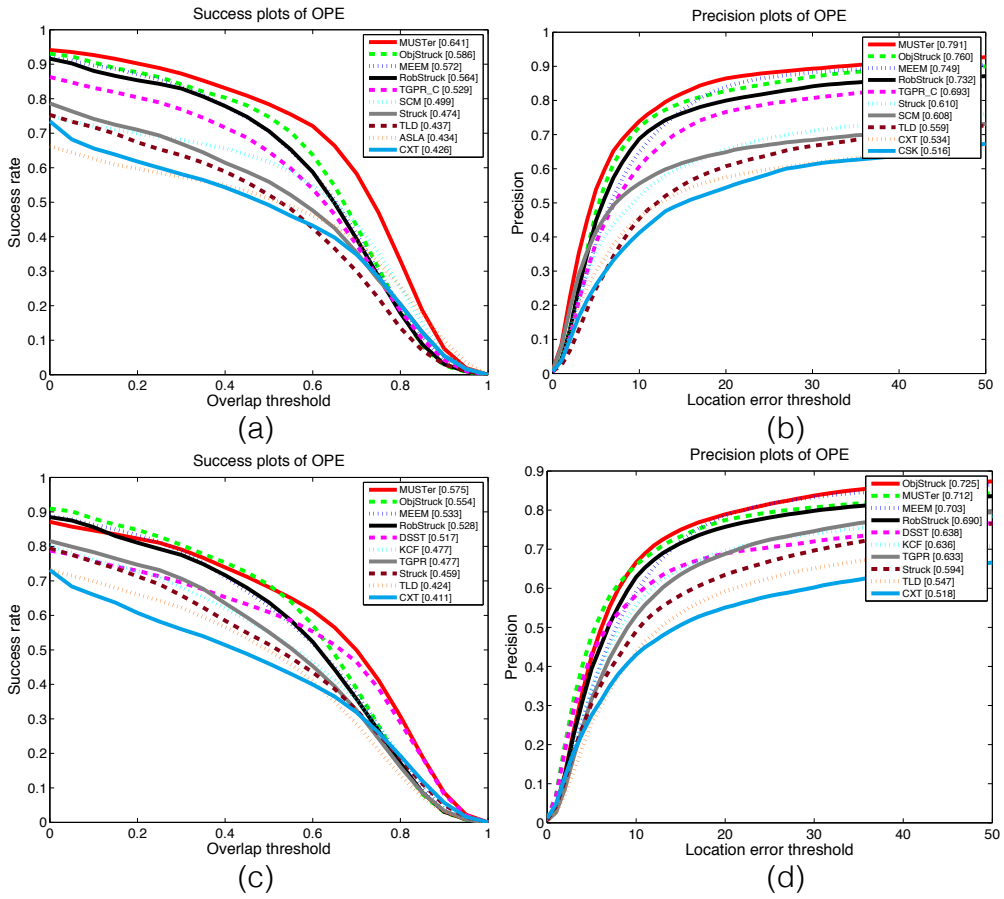

Fig. 3: (a) Overlap metric on [27] dataset. (b) Precision metric on [27] dataset. (c) Overlap metric on [26] dataset. (d) Precision metric on [26] dataset.

defined as the fraction of frames where

$$\frac{r_t \cap r_{gt}}{r_t \cup r_{gt}} \geq \sigma. \tag{9}$$

Evaluation setup. In all experiments, to assess how objectness affects tracking, we compared our tracker with (i.e., ObjStruck) and without objectness measures (i.e., RobStruck).

A. *OTB50 dataset*

The dataset in [27] has 50 videos. It includes evaluations of 29 different trackers such as MIL [2], SCM [29], and TLD [16]. Additionally, we added other trackers which were highly ranked on datasets such as MUSTer [14], MEEM [28], and TGPR_C [11]. Figures 3 (a), 3 (b) show that objectness improves tracking results, compared to the baseline.

B. *OTB100 dataset*

The benchmark [26] is an extension of the [27] benchmark. It contains 100 videos. Same metrics, precision and overlap, were used in the evaluation. Figures 3 (c), 3 (d) show that objectness similarly improves tracking results. Interestingly, ObjStruck outperforms MUSTer tracker on precision metric. This suggests that objectness measures reduce overfitting by providing additional cues of the target's location.

C. *VOT 2015*

TABLE I: Evaluation results on the [19] dataset.

Tracker name	Overlap
MDNet	0.3783
DeepSRDCF	0.3181
EBT	0.3130
srdcf	0.2877
LDP	0.2785
sPST	0.2767
scebt	0.2548
nsamf	0.2536
struck	0.2458
ObjStruck	0.2355
sumshift	0.2341
SODLT	0.2329
DAT	0.2238
RobStruck	0.2198
MUSTer	0.1950

The benchmark [19] contains 60 videos which were chosen to maximize visual attributes such as illumination change, object size change, and object motion. The benchmark is

different from others because of its evaluation protocol. In [19], whenever the tracker fails (i.e., fail is defined as a detection that has zero overlap with ground truth), the tracker is re-initialized. This strategy allows to reduce overfitting during tracking evaluation [17]. Results are shown in Table I. Here, objectness improves tracking as success for ObjStruck are higher than for RobStruck.

Overall, objectness measures improve both success and precision metrics on all of the datasets with different evaluation protocols. This is surprising as ObjStruck uses exactly the same set of potential candidates as RobStruck.

VIII. Conclusion

We extended structured tracking by including object-agnostic measures of the bounding box to improve the tracker's discrimination power. The measures, known as straddling and edge density, guide the tracker to bounding boxes that are more likely to contain objects. The resulting tracker, which we call *ObjStruck*, improves tracking metrics on major datasets [27, 26, 19]. Although we used the objectness prior along with a robust version of the structured tracker, our objectness measure works with any tracker that samples bounding boxes. Straddling and edge density are fast to compute and unsupervised, which makes them ideal for tracking.

The objectness measures we used work independently on each frame. This could limit their accuracy and increase computational cost as both segmentation and edge detection have to be done in each frame. In the future, we plan to address this issue by reusing computations and extending the objectness measure in the temporal domain.

References

[1] Bogdan Alexe, Thomas Deselaers, and Vittorio Ferrari. "Measuring the objectness of image windows". In: *IEEE PAMI* 34.11 (2012), pp. 2189–2202.

[2] Boris Babenko, Ming-Hsuan Yang, and Serge Belongie. "Visual tracking with online multiple instance learning". In: *IEEE CVPR*. 2009, pp. 983–990.

[3] Michael Van den Bergh et al. "Seeds: Superpixels extracted via energy-driven sampling". In: *European conference on computer vision*. Springer. 2012, pp. 13–26.

[4] David S Bolme et al. "Visual object tracking using adaptive correlation filters". In: *IEEE CVPR*. 2010, pp. 2544–2550.

[5] Antoine Bordes, Nicolas Usunier, and Léon Bottou. "Sequence labelling SVMs trained in one pass". In: *MLKDD*. Springer, 2008, pp. 146–161.

[6] Ali Borji, Dicky N Sihite, and Laurent Itti. "Salient object detection: A benchmark". In: *ECCV*. Springer, 2012, pp. 414–429.

[7] Joao Carreira and Cristian Sminchisescu. "CPMC: Automatic object segmentation using constrained parametric min-cuts". In: *IEEE PAMI* 34.7 (2012), pp. 1312–1328.

[8] Ming-Ming Cheng et al. "BING: Binarized normed gradients for objectness estimation at 300fps". In: *IEEE CVPR*. 2014, pp. 3286–3293.

[9] Martin Danelljan et al. "Accurate scale estimation for robust visual tracking". In: *BMVC*. 2014.

[10] Ian Endres and Derek Hoiem. "Category-independent object proposals with diverse ranking". In: *IEEE PAMI* 36.2 (2014), pp. 222–234.

[11] Jin Gao et al. "Transfer learning based visual tracking with gaussian processes regression". In: *ECCV*. Springer, 2014, pp. 188–203.

[12] Sam Hare, Amir Saffari, and Philip HS Torr. "Struck: Structured output tracking with kernels". In: *IEEE ICCV*. 2011, pp. 263–270.

[13] João F Henriques et al. "Exploiting the circulant structure of tracking-by-detection with kernels". In: *ECCV*. Springer, 2012, pp. 702–715.

[14] Zhibin Hong et al. "MUlti-Store Tracker (MUSTer): a Cognitive Psychology Inspired Approach to Object Tracking". In: *IEEE CVPR*. 2015, pp. 749–758.

[15] Dafei Huang et al. "Enable Scale and Aspect Ratio Adaptability in Visual Tracking with Detection Proposals". In: *BMVC*. 2015, pp. 185.1–185.12.

[16] Zdenek Kalal, Krystian Mikolajczyk, and Jiri Matas. "Tracking-learning-detection". In: *IEEE PAMI* 34.7 (2012), pp. 1409–1422.

[17] Matej Kristan et al. *A Novel Performance Evaluation Methodology for Single-Target Trackers*. 2016.

[18] Matej Kristan et al. "The visual object tracking vot2014 challenge results". In: *ECCV Workshops*. Springer. 2014, pp. 191–217.

[19] Matej Kristan et al. "The Visual Object Tracking VOT2015 Challenge Results". In: *ICCV Workshops*. 2015, pp. 1–23.

[20] Dae-Youn Lee, Jae-Young Sim, and Chang-Su Kim. "Multihypothesis Trajectory Analysis for Robust Visual Tracking". In: *IEEE CVPR*. 2015, pp. 5088–5096.

[21] Yang Li and Jianke Zhu. "A scale adaptive kernel correlation filter tracker with feature integration". In: *ECCV Workshops*. Springer. 2014, pp. 254–265.

[22] Pengpeng Liang et al. "Adaptive Objectness for Object Tracking". In: *arXiv preprint arXiv:1501.00909* (2015).

[23] Subhransu Maji, Alexander C Berg, and Jitendra Malik. "Efficient classification for additive kernel SVMs". In: *IEEE PAMI* 35.1 (2013), pp. 66–77.

[24] Peter Ruckdeschel. *Robust kalman filtering*. Springer, 2000.

[25] Longyin Wen et al. "JOTS: Joint Online Tracking and Segmentation". In: *IEEE PAMI*. 2015, pp. 2226–2234.

[26] Yi Wu, Jongwoo Lim, and Ming-Hsuan Yang. "Object tracking benchmark". In: *IEEE PAMI* 37.9 (2015), pp. 1834–1848.

[27] Yi Wu, Jongwoo Lim, and Ming-Hsuan Yang. "Online object tracking: A benchmark". In: *Computer Vision and Pattern Recognition (CVPR), 2013 IEEE Conference on*. IEEE. 2013, pp. 2411–2418.

[28] Jianming Zhang, Shugao Ma, and Stan Sclaroff. "MEEM: robust tracking via multiple experts using entropy minimization". In: *ECCV*. 2014.

[29] Wei Zhong, Huchuan Lu, and Ming-Hsuan Yang. "Robust object tracking via sparsity-based collaborative model". In: *IEEE CVPR*. 2012, pp. 1838–1845.

[30] Gao Zhu, Fatih Porikli, and Hongdong Li. "Tracking randomly moving objects on edge box proposals". In: *arXiv preprint arXiv:1507.08085* (2015).

[31] C Lawrence Zitnick and Piotr Dollár. "Edge boxes: Locating object proposals from edges". In: *ECCV* (2014), pp. 391–405.