MBestStruck: M-Best diverse sampling for structured tracker

Ivan Bogun
http://my.fit.edu/~ibogun2010
Eraldo Ribeiro
http://cs.fit.edu/~eribeiro

Department of Computer Sciences and Cybersecurity
School of Computing
Florida Institute of Technology
Melbourne, U.S.A.

Abstract

We approach the problem of *model-free* visual tracking of objects in videos. Model-free tracking has its state-of-the-art in a class of methods called *tracking-by-detection*, as shown in recent benchmarks. Some top-performing methods use deep neural networks (i.e., convnets) to solve the learning-based steps of the tracking algorithm (e.g., bounding-box prediction and evaluation). Despite improving accuracy, *convnets* impose a high computational cost on trackers, limiting their real-time applications. In this paper, we propose to use deep features from a pre-learned deep-convolutional network in a computationally efficient way. Here, we use *M-Best diverse-sampling* to sample a small yet diverse set of bounding boxes that are likely to contain the tracked object. Given these bounding boxes, our method performs detection using deep features. The resulting tracker, named *MBestStruck*, uses a high-quality feature representation while being computationally efficient. Our tracking approach compares very well to the state-of-the-art, as shown by experiments done on popular benchmark datasets.

1 Introduction

Locating a moving object in the frames of a video is the main goal of visual-tracking algorithms. These algorithms find applications in robotics, unmanned-vehicle navigation, and surveillance, which have motivated much progress in the development of solutions. State-of-the-art tracking methods belong to a class of approaches called tracking-by-detection, whose excellent performance is evident from results presented at recent benchmarks [19, 28, 29].

Tracking by detection works by the recurring application of a detection algorithm, which is usually based on classification methods. Avidan [2] posed tracking as a binary classification problem where the target object is the positive class, and the background is the negative. Instead of using a binary classifier, Babenko et al. [3]'s tracker uses multiple-instance learning. Recently, Hare et al. [10] posed tracking as a structured-learning problem. While effective, these algorithms are computationally heavy. Faster performance is achieved by searching around the target's last location and by sampling a few locations for detection.

The sampling of locations that might contain the target object is a key step in tracking methods. Despite its importance, sampling has received little attention. Typically, sampling

Pages 93.1-93.12
DOI: https://dx.doi.org/10.5244/C.30.93

is done at pre-defined equidistant image locations [3, 10, 11], or sometimes at random locations as in Nam and Han [22] and Zhong et al. [31]. Nevertheless, such approaches are suboptimal because the correctness of the detection step depends on the detection algorithm.

We present a novel sampling strategy that assesses high-quality diverse locations. Our approach is based on two assumptions. First, the optimal sampling strategy should only consider locations with a high likelihood of containing the target. Secondly, the locations must be *diverse*. Here, we pose sampling as an *M-Best diverse sampling problem* [4, 16]. Our contributions are: (i) we pose the sampling step as an M-Best diverse sampling problem; (ii) we present a cascade tracker, called *MBestStruck*, which uses deep-learning features in a computationally-efficient manner; (iii) we evaluate our method on publicly available datasets [19, 28, 29] and show that our tracker performs comparably to the state-of-the-art.

2 Related work

Our tracker is based on the structured tracker by Hare et al. [10] known as *Struck*. The structured tracker learns a mapping directly from the space of images onto the space of bounding boxes. It uses the structured support vector machine to build the object's appearance model in an on-line manner. The appearance model in Hare et al. [10] consists of a number of positive (i.e., target) and negative (i.e., background) support vectors. To run in real time, the number of support vectors is fixed by a quantity known as a *budget*. The structured tracker achieved state-of-the-art results on challenging benchmarks [28, 29].

The remarkable success of deep learning in computer vision has motivated the development of trackers based on neural networks (i.e., convnets). Deep convnets extract meaningful mid-level features in a data-driven way instead of relying on hand-engineered visual features. This data-driven feature extraction has been incorporated into a number of tracking methods [12, 22, 26, 27]. Wang *et al*. used a particle-filter tracker with features based on pre-trained stacked autoencoder. [12] used a pre-trained convolutional neural network (CNN) with a online support vector machine to create a target-specific map which performs detection at a pixel level. Although these methods showed an improvement, the best trackers from the benchmarks in [17, 28] were still based on hand-engineered features. This changed when the tracker known as MDNet [22] won the VOT2015 challenge [19] by a large margin.

Indeed, MDNet [22] has been the top performer on all benchmarks [19, 22]. MDNet uses *domain adaptation* – a technique where the learning is performed on multiple domains, and domain information is used for learning. The weights in MDNet's convolutional network are initialized from the pre-trained ImageNet model [22]. Prior to tracking, MDNet is pre-trained to do object detection on a large number of videos. During tracking, MDNet uses stochastic gradient descent to adapt to the changing appearance of the tracked object.

MDNet is a powerful tracker, but with a high computational cost. It is reported to run at 1 fps on a GPU. In its detection step, MDNet samples 256 bounding boxes from a Gaussian distribution. As such, there is an implicit trade-off between how many locations are considered and the tracking accuracy. Therefore, sampling is a key step in the tracking pipeline.

Sampling has received little attention when compared to other components of tracking methods. Tracking-by-detection methods sample bounding boxes deterministically around previously seen locations [9, 10] or even randomly [22, 23]. These techniques are suboptimal because they do not guarantee that the tracked object will be in one of the sampled boxes. A better approach is to sample in a information-driven way where sampled bounding boxes are more likely to contain objects, and consequently the target object.

The use characteristics of generic objects to tell whether a bounding box is likely to contain *any* object has been successfully applied to tracking [14, 21, 32]. These methods use object-agnostic metrics, known as *objectness measures*, to rank sampled bounding boxes. Zhu et al. [32] re-rank sampled boxes using a structured SVM learned from edge-box scores [33]. Huang et al. [14] sample bounding boxes with high objectness metrics for scale adaptation. Although the above methods sample bounding boxes that are indeed likely to contain the object, the sampling process is unaware of the object being tracked. In this paper, we argue that the sampling can be improved if it takes into account object's appearance.

3 Base tracker

Our tracker is built as a cascade of three main components: a base tracker, a sampling procedure, and a top-level tracker that performs final detection. Here, we describe both trackers in our cascade, which are based on the structured tracker by [10]. The structured tracker uses a structured support vector machine to learn a direct mapping from the space of images to the space of bounding boxes. Its formulation uses the *overlap-over union* metric (i.e., a tracking-evaluation metric) as a loss function for the SVM's learning step. The structured tracker outperformed other trackers on the OTB50 and the OTB100 benchmarks [28, 29].

3.1 Structured SVM and the structured tracker

Tsochantaridis et al. [25] generalized *support vector machines* (SVM) to learn functions in arbitrary, *structured* spaces, i.e., $f : \mathcal{X} \to \mathcal{Y}$. Let $(\mathbf{x}_i, \mathbf{y}_i)_{i=1}^n$ be a set of input-output pairs, then the structured SVM solves the following convex-optimization problem:

$$\max_{\beta} \quad \sum_{i,\mathbf{y}} \Delta(\mathbf{y},\mathbf{y}_i)\beta_i^{\mathbf{y}} - \frac{1}{2}\sum_{i,\mathbf{y},j,\bar{\mathbf{y}}} \beta_i^{\mathbf{y}}\beta_j^{\bar{\mathbf{y}}}\langle\phi(\mathbf{x}_i,\mathbf{y}),\phi(\mathbf{x}_j,\bar{\mathbf{y}})\rangle \tag{1}$$

$$\text{s.t.} \quad \forall i, \forall \mathbf{y} : \beta_i^{\mathbf{y}} \leq \delta(\mathbf{y},\mathbf{y}_i)C, \quad \text{and} \quad \forall i : \sum_{\mathbf{y}} \beta_i^{\mathbf{y}} = 0, \tag{2}$$

where $\delta(\mathbf{y},\bar{\mathbf{y}})$ is 1 for $\mathbf{y} = \bar{\mathbf{y}}$ and 0 otherwise. Function $\Delta(\cdot,\cdot)$ is called a *loss* function, which for tracking is defined as the "one minus intersection-over-union" metric:

$$\Delta(\mathbf{y},\hat{\mathbf{y}}) = 1 - \frac{|\mathbf{y} \cap \hat{\mathbf{y}}|}{|\mathbf{y} \cup \hat{\mathbf{y}}|}. \tag{3}$$

The loss function is designed to penalize bounding boxes that have low overlap with the ground truth given their size. Detection is performed using *discriminative* function $F : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$, which is defined as a linear combination of the support vectors, i.e.:

$$F(\mathbf{x},\mathbf{y};\beta) = \sum_{i,\bar{\mathbf{y}}} \beta_i^{\bar{\mathbf{y}}}\langle\phi(\mathbf{x_i},\bar{\mathbf{y}}),\phi(\mathbf{x},\mathbf{y})\rangle. \tag{4}$$

The structured tracker by Hare et al. [10] solves Equation 1 using the SMO-step algorithm [7], which monotonically improves the objective function. To keep the computational cost fixed when tracking, the number of support vectors is limited by a *budget*. When the number of support vectors exceeds the budget, the contribution of each vector to the objective function is computed, and those vectors having the lowest contribution are deleted.

3.2 Trajectory correction using Robust Kalman filter

To improve the tracker's robustness to both false-positive detections and short-time occlusions, we use the Robust Kalman filter [6, 24] with a constant-velocity model to correct the location of the best-detected bounding box. 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 filter's. This search strategy allows our tracker to detect and recover from misdetections.

3.3 Objectness prior

We also add an object-agnostic prior, known as *objectness*, to the discriminative function to guide the tracker into locations that are likely to contain any object [5]. Following Alexe et al. [1], we use the *straddling* and the *edge density* objectness priors. The *straddling* metric captures the degree in which the bounding box cuts segments after segmentation. Formally, let $b = [c_x, c_y, w, h]$ be a bounding box, and let S be a set of superpixels after segmentation. Straddling measures how much the bounding box divides the superpixels, and is given by:

$$s(b) = 1 - \sum_{i \in S} \frac{\min(|i \setminus b|, |i \cap b|)}{|b|}, \tag{5}$$

where $s(b) \in [0,1], \forall b$. Edges can also serve as an objectness measure [1, 33]. Here, we use a metric known as edge density [1], which measures the fraction of pixels classified as edges in the bounding box, divided by the perimeter, i.e.:

$$e(b) = \frac{\sum_{(x,y) \in Perimeter(b)} \mathbb{1}(\text{edge}(x,y))}{2(w+h)}, \tag{6}$$

where $\text{edge}(x,y)$ is a binary mask that has ones at edge pixels.

4 Method

4.1 Sampling procedure

To detect the bounding boxes with a high likelihood of the target, we use the detection function from the base tracker (i.e., ObjStruck). However, if the set of bounding boxes is chosen as a subset of ObjStruck with the maximum of the detector function, bounding boxes are all very similar (Figure 1). This lack of diversity of the sampled bounding boxes increases the likelihood of loss of track. The underlying question is then how to sample a small set of bounding boxes that are all of high quality, yet are different (i.e., diverse).

The problem of finding high quality, yet diverse solutions of an energy function is called *M-Best-Diverse labeling* [4, 16]. M-Best-Diverse contrasts with traditional energy-minimization approaches, which select only the solution with the lowest energy. Instead, the M-Best-Diverse labeling seeks a number of low-energy solutions that are diverse. M-Best-Diverse labeling is formalized as follows. Let $E : \mathcal{Y} \to \mathbb{R}$ be an energy function that we define as a negative ObjStruck discriminative function:

$$E(\mathbf{y}) = -\sum_{i,\bar{\mathbf{y}}} \beta_i^{\bar{\mathbf{y}}} \langle \phi(\mathbf{x_i}, \bar{\mathbf{y}}), \phi(\mathbf{x}, \mathbf{y}) \rangle - \lambda_s s(\mathbf{y}) - \lambda_e e(\mathbf{y}), \tag{7}$$

Figure 1: Left: tracking the "Basketball" sequence. Right: ObjStruck detector values as a function of the bounding box's center. Bounding boxes with high detection scores are not diverse – they are next to each other and have similar dimensions.

where $\lambda_e, \lambda_s > 0$ are objectness parameters, and $s(\cdot), e(\cdot)$ are the objectness measures straddling and edge density, respectively [5]. Batra et al. [4] uses a greedy sequential procedure for finding M diverse labelings, $\mathbf{y_1}, \ldots, \mathbf{y_M}$, according to the following criterion:

$$\mathbf{y}^m = \arg\min_{y \in \mathcal{Y}} \left[E(\mathbf{y}) - \lambda \sum_{i=1}^{m-1} \Delta(\mathbf{y}, \mathbf{y}^i) \right], \tag{8}$$

for $i = 1, \ldots, M$, where parameter $\lambda > 0$ controls a trade-off between the diversity of the labelings and their quality. The function $\Delta(\cdot,\cdot) : \mathcal{Y} \times \mathcal{Y} \rightarrow \mathbf{R}$ is called a *dissimilarity kernel*. It represents the diversity between any two labellings. Different labeling solutions result in high diversity values while similar solutions result in small diversity values.

4.2 MBestStruck algorithm

Our combination of the structured SVM with the M-Best selection of bounding boxes is summarized in Algorithms 1 and 2. The initialization step, compared to the same step in ObjStruck has an additional structured support vector machine (SSVM) and an additional dissimilarity function. In the detection step, after applying the objectness measures, M-Best sampling adds new bounding boxes according to Equation 8 on each iteration. Our method lowers computation cost by caching the dissimilarity function. Figure 2 shows examples of tracking using our method *MBestStruck*.

Algorithm 1: MBestStruck initialization

input: feature, kernel, top feature, top kernel,
dissimilarity feature g_{dis}, dissimilarity kernel K_{dis},
image I, ground truth bounding box B_{in}

$D_{\text{ObjStruck}} := \text{SSVM}(\text{feature}, \text{kernel})$;
$D_{\text{DeepStruck}} := \text{SSVM}(\text{top feature}, \text{top kernel})$;
$\Delta(\mathbf{y}, \mathbf{y}') := K_{\text{dis}}(g_{\text{dis}}(I|_{\mathbf{y}}), g_{\text{dis}}(I|_{\mathbf{y}'}))$;
$S_{\text{update}} = \text{sample}(B_{\text{in}})$;
Initialize($D_{\text{ObjStruck}}$, S_{update}, I);
Initialize($D_{\text{DeepStruck}}$, S_{update}, I);

Algorithm 2: MBestStruck detection

input : image I, last bounding box B_{last}
output: bounding box B_{out}

$S_{\text{search}} = \text{sample}(B_{\text{last}})$;
$P_{\text{ObjStruck}} = \text{predict}(D_{\text{ObjStruck}}, S_{\text{search}})$
$\quad + \lambda_e \text{ straddeling}(S_{\text{search}}) + \lambda_e \text{ edge density}(S_{\text{search}})$;
initialize empty list S_{MBest} ;
while *size of* S_{MBest} *is not* M **do**
  $\mathbf{y}^m = \arg\min_{\mathbf{y} \in S_{\text{search}}} [-P_{\text{ObjStruck}}(\mathbf{y}) - \lambda \sum_{\mathbf{y}' \in S_{\text{MBest}}} \Delta(\mathbf{y}, \mathbf{y}')]$;
  add $\mathbf{y}^m$ to S_{MBest} ;
$P_{\text{DeepStruck}} = \text{predict}(D_{\text{DeepStruck}}, S_{\text{MBest}})$;
$B_{\text{out}} = \arg\max_b P_{\text{DeepStruck}}(b)$

5 Implementation

We used the same parameters as in ObjStruck [5]. But the proposed method MBestStruck introduces two new parameters: the number of bounding boxes to use during sampling, M, and the trade-off between quality and the diversity, λ. We use two trackers with a different set of parameters: one that samples 32 bounding boxes using the M-Best procedure (M32BestStruck) and one that samples 64 boxes (M64BestStruck). We set λ to 0.1 and 0.0875, respectively. These values were chosen empirically. The dissimilarity kernel, $\Delta(\cdot,\cdot)$, in Equation 8, is defined as one minus cosine distance. The cosine distance is computed over HOG features [8]. The top tracker is a structured SVM with a linear kernel and features from the pre-trained AlexNet [20]. We use features from the *fc7* layer that result in 4,096-dimensional feature vector, and implementation from Caffe [15].

ScaleDeepStruck. To isolate the effect of sampling from the deep features, we define another tracker called *ScaleDeepStruck*, which works just as MBestStruck but it does not use the M-Best sampling procedure. ScaleDeepStruck samples bounding boxes deterministically in the small vicinity (i.e., 15 pixels away or less) of the base tracker's best detection. The samples are computed by linearly spacing the center of the bounding box in the polar coordinates and by varying translation, scale, and aspect ratio. In this experiment, we sample the same number of bounding boxes as we do for M32BestStruck (i.e., 32).

Figure 2: MBestStruck tracking the "Ironman" sequence. Rectangles correspond to bounding boxes of the final tracker's detection (Blue), the current state of the Robust Kalman filter location (Green), and bounding boxes sampled via Equation 8 (Red). After initialization on Frame 1, M-Best sampling allows to assess a rich set of potential candidates as seen in Frames 7 and 9. Because of strong illumination changes, the tracker mis-detects in Frames 11, 13, 16, and 18. Nevertheless, M-Best sampling includes the correct bounding box in Frames 16 and 18. After a drift of the tracker, and lack of a correct bounding box via M-Best sampling procedure, the tracker recovers via the Robust Kalman filter (Frame 19).

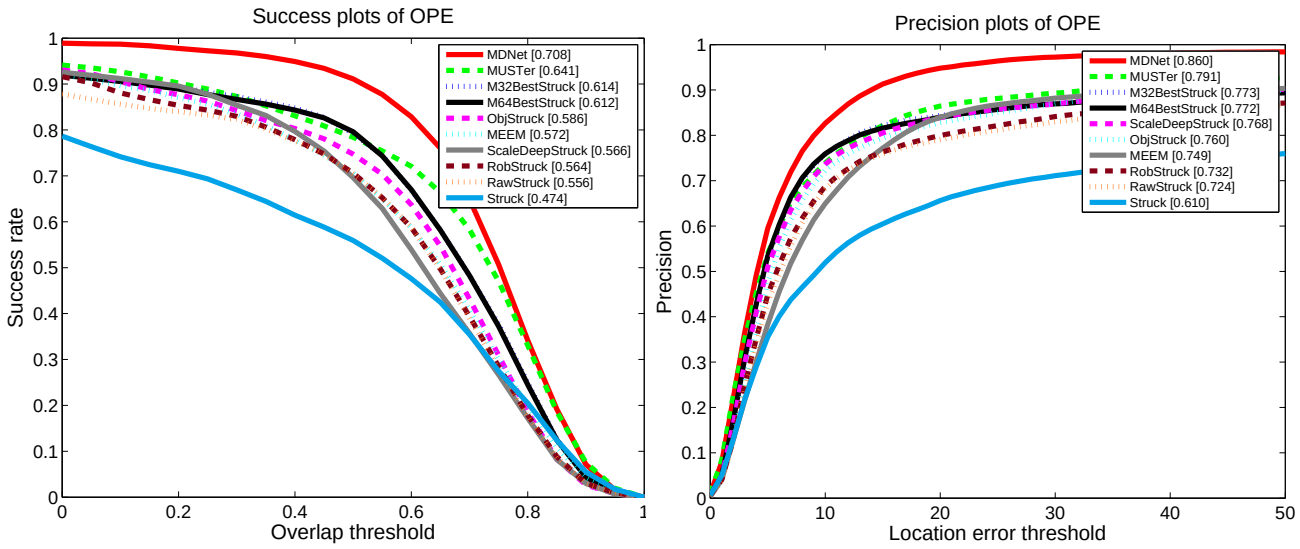

Figure 3: Overlap and precision metrics on [28] dataset.

Speed. M32BestStruck's running speed is 0.15 frames per second on a single-threaded single-core 2.4 GHz processor. Deep feature extraction accounts for 30% of the total computation. In comparison, ObjStruck (base tracker in the cascade) runs at 1.7 FPS.

6 Results

We tested our method on the datasets [19, 28, 29], which have been used for recent tracking benchmarks, and are publicly available. By using these datasets, we can demonstrate our method's effectiveness and also show how it compares to the state-of-the-art. Our evaluation included the state-of-the-art tracker MDNet [22], which won the VOT 2015 benchmark [19], as well as other trackers such as MUSTer [13], MEEM [30], and the original Struck [10]. We also included our own Struck implementation (RawStruck), and a number of structured tracker extensions: structured tracker with trajectory correction using Robust Kalman filter called RobStruck [6] and object-aware structured tracker called ObjStruck [5].

OTB50 dataset. We tested MBestStruck on Wu et al. [28]'s dataset. Success and precision curves are shown in Figure 3. The curves show that ScaleDeepStruck has lower tracking performance than our base tracker (ObjStruck). This drop in performance happens because the bounding boxes used for detection in the deep SSVM rely on a single-best detection from a base tracker. This dependency on a single detection implies that a single error of the base tracker immediately propagates to the deep SSVM. Additionally, we observed that when ScaleDeepStruck failed, it did early. To benefit from the discriminative ability of deep features, many support vectors need to be created. By failing early, ScaleDeepStruck was unable to collect enough support vectors to correctly detect scale and aspect ratio.

Our proposed tracker, MBestStruck, in contrast, significantly improved tracking metrics. This improvement is due to the quality of the bounding boxes sampled by the M-Best procedure. Having a few, but different bounding boxes that have high-detection scores from the base SSVM simplifies detection done via deep SSVM. Remarkably, M32BestStruck uses the same number of bounding boxes as ScaleDeepStruck (e.g., 32).

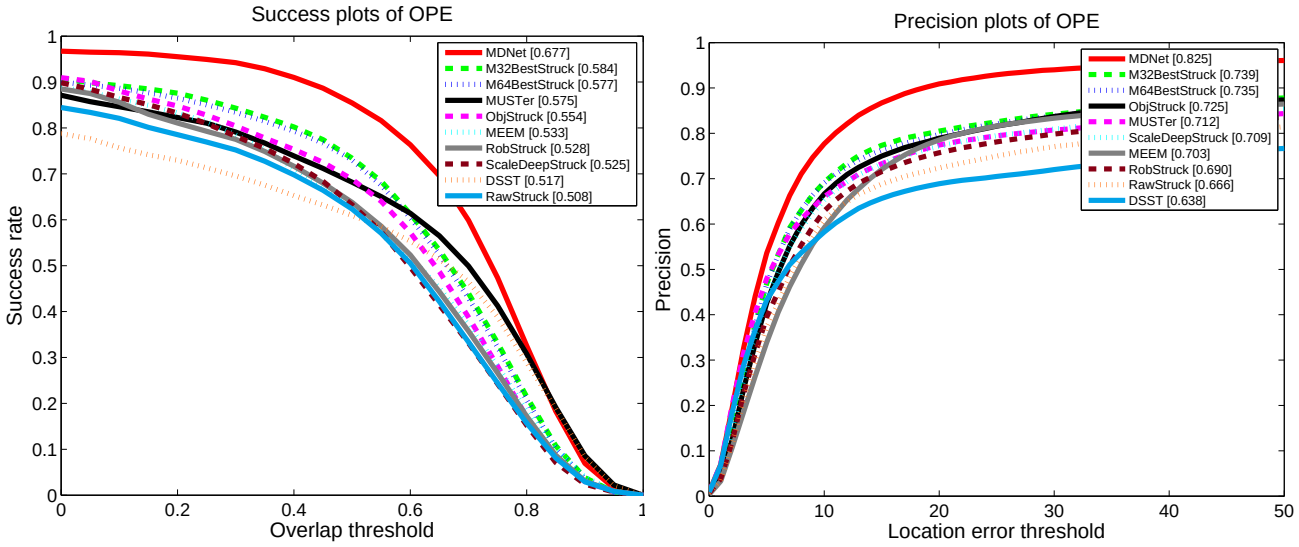

Figure 4: Overlap and precision metrics on [29] dataset.

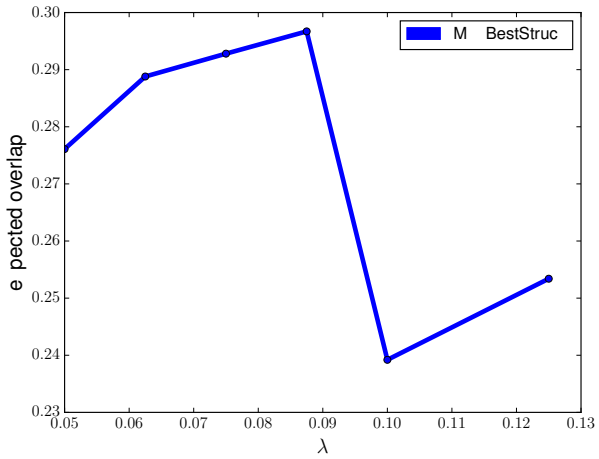

Figure 5: Sensitivity analysis (λ parameter) done on the VOT2015 dataset with $M = 64$.

OTB100 dataset. Results of the evaluation done on the extended benchmark [29] are shown in Figure 4. Here, MBestStruck improves upon both ObjStruck and ScaleDeepStruck.

VOT 2015. We also tested our tracker on the VOT 2015 benchmark Kristan et al. [19]. Here, a tracker called struck [10] is an extension of the original structured tracker by Hare *et al*. Results are summarized in Table 1. The winner and the runner-up in the VOT 2015 benchmark used deep learning in their feature extraction step. Thus, it was expected for MBestStruck to perform well. What is more remarkable is the amount of improvement in the overlap metric. Unlike evaluations on OTB50 and on OTB100, M64BestStruck performs much better than M32BestStruck. This is due to the re-initialization evaluation protocol of the VOT2015 benchmark. Re-initialization allows for multiple failures during the tracking which in turn decreases the role of how early the tracker failed. In the OTB100 evaluation, once the tracker loses track, recovery becomes unlikely. Because more bounding boxes are sampled, M64BestStruck has a better discriminative ability as shown in the evaluation.

Parameter λ controls the trade-off between accuracy and diversity, and plays a crucial role in our method. Figure 5 shows the accuracy versus λ for $M = 64$. The results suggest that there is a non-linear dependency. A drop in accuracy after $\lambda = 0.0875$ indicates that once solutions are too diverse, the tracker is more likely to make a mistake.

Table 1: Evaluation results on the [19] dataset.

Expected overlap	Overall	Expected overlap	Overall
MDNet	0.3783	sumshift	0.2341
DeepSRDCF	0.3181	SODLT	0.2329
EBT	0.3130	DAT	0.2238
M64BestStruck $\lambda = 0.0875$	**0.2967**	**RobStruck**	**0.2198**
srdcf	0.2877	OACF	0.2190
LDP	0.2785	MCT	0.2188
M32BestStruck $\lambda = 0.1$	**0.2774**	mkcf_plus	0.2095
sPST	0.2767	tric	0.2088
scebt	0.2548	AOGTracker	0.2080
nsamf	0.2536	sme	0.2068
struck	0.2458	mvcft	0.2059
rajssc	0.2420	srat	0.2031
s3tracker	0.2403	dtracker	0.2022
ObjStruck	**0.2355**	muster	0.1950

7 Conclusions

We presented a tracking-by-detection method that both samples high-quality bounding boxes and takes advantage of deep-learning features, while keeping computational cost low. Here, we use a technique for sampling a small but diverse set of bounding boxes and showed how our tracker can be integrated with powerful features extracted using a pre-trained deep-convolutional network. We also showed that sampling a small number of bounding boxes built on top of the base tracker is feasible, thus making it computationally efficient while still keeping the tracker accurate.

We extensively evaluated each of our trackers on the most popular datasets and benchmarks [18, 19, 28, 29], and showed that our method improves upon the baseline.

Acknowledgment Research supported in part by the U.S. National Science Foundation under Grant No. 1152306.

References

[1] Bogdan Alexe, Thomas Deselaers, and Vittorio Ferrari. Measuring the objectness of image windows. *IEEE PAMI*, 34(11):2189–2202, 2012.

[2] Shai Avidan. Ensemble tracking. *IEEE PAMI*, 29(2):261–271, 2007.

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

[4] Dhruv Batra, Payman Yadollahpour, Abner Guzman-Rivera, and Gregory Shakhnarovich. Diverse M-best solutions in Markov random fields. In *ECCV*, pages 1–16. Springer, 2012.

[5] Ivan Bogun and Eraldo Ribeiro. Object-aware tracking. *ICPR (to appear)*, 2016.

[6] Ivan Bogun and Eraldo Ribeiro. Robstruck: Improving occlusion handling of structured tracking-by-detection using robust Kalman filter. *ICIP (to appear)*, 2016.

[7] Antoine Bordes, Léon Bottou, Patrick Gallinari, and Jason Weston. Solving multiclass support vector machines with larank. In *ACM ICML*, pages 89–96, 2007.

[8] Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. In *IEEE CVPR*, volume 1, pages 886–893, 2005.

[9] Martin Danelljan, Gustav Häger, Fahad Shahbaz Khan, and Michael Felsberg. Accurate scale estimation for robust visual tracking. In *BMVC*, 2014.

[10] S Hare, S Golodetz, A Saffari, V Vineet, MM Cheng, S Hicks, and P Torr. Struck: Structured output tracking with kernels. *IEEE PAMI*, 2015.

[11] João F Henriques, Rui Caseiro, Pedro Martins, and Jorge Batista. Exploiting the circulant structure of tracking-by-detection with kernels. In *ECCV*, pages 702–715. Springer, 2012.

[12] S. Hong, T. You, S. Kwak, and B. Han. Online tracking by learning discriminative saliency map with convolutional neural network. In *ICML*, 2015.

[13] Zhibin Hong, Zhe Chen, Chaohui Wang, Xue Mei, Danil Prokhorov, and Dacheng Tao. Multi-store tracker (muster): a cognitive psychology inspired approach to object tracking. In *IEEE CVPR*, pages 749–758, 2015.

[14] Dafei Huang, Lei Luo, Mei Wen, Zhaoyun Chen, and Chunyuan Zhang. Enable scale and aspect ratio adaptability in visual tracking with detection proposals. In *BMVC*, pages 185.1–185.12, September 2015.

[15] Yangqing Jia, Evan Shelhamer, Jeff Donahue, Sergey Karayev, Jonathan Long, Ross Girshick, Sergio Guadarrama, and Trevor Darrell. Caffe: Convolutional architecture for fast feature embedding. In *ACM Intl. Conf. on Multimedia*, pages 675–678, 2014.

[16] A. Kirillov, D. Shlezinger, D. P. Vetrov, C. Rother, and B. Savchynskyy. M-best-diverse labelings for submodular energies and beyond. In *NIPS*, pages 613–621, 2015.

[17] Matej Kristan, Roman Pflugfelder, Ale Leonardis, Jiri Matas, Fatih Porikli, Luka Cehovin, Georg Nebehay, Gustavo Fernandez, Toma Vojir, Adam Gatt, et al. The visual object tracking VOT2013 challenge results. In *IEEE ICCVW*, pages 98–111, 2013.

[18] Matej Kristan, Roman Pflugfelder, Aleš Leonardis, Jiri Matas, Luka Čehovin, Georg Nebehay, Tomáš Vojíř, Gustavo Fernandez, Alan Lukežič et al. The visual object tracking VOT 2014 challenge results. In *ECCVW*, pages 191–217, 2014.

[19] Matej Kristan, Jiri Matas, Ales Leonardis, Michael Felsberg, Luka Cehovin, Gustavo Fernandez, Tomas Vojir, Gustav Hager, Georg Nebehay, and Roman Pflugfelder. The visual object tracking VOT 2015 challenge results. In *IEEE ICCVW*, pages 1–23, 2015.

[20] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *NIPS*, pages 1097–1105, 2012.

[21] Pengpeng Liang, Chunyuan Liao, Xue Mei, and Haibin Ling. Adaptive objectness for object tracking. *IEEE Signal Processing Letters*, Vol. 23, No. 7, pages 949–953, 2016.

[22] Hyeonseob Nam and Bohyung Han. Learning multi-domain convolutional neural networks for visual tracking. In *IEEE CVPR*, 2016.

[23] David A Ross, Jongwoo Lim, Ruei-Sung Lin, and Ming-Hsuan Yang. Incremental learning for robust visual tracking. *IJCV*, 77(1-3):125–141, 2008.

[24] Peter Ruckdeschel. *Robust Kalman Filtering*. Springer, 2000.

[25] Ioannis Tsochantaridis, Thorsten Joachims, Thomas Hofmann, and Yasemin Altun. Large margin methods for structured and interdependent output variables. *JMLR*, pages 1453–1484, 2005.

[26] Naiyan Wang and Dit-Yan Yeung. Learning a deep compact image representation for visual tracking. In *NIPS*, pages 809–817, 2013.

[27] Naiyan Wang, Siyi Li, Abhinav Gupta, and Dit-Yan Yeung. Transferring rich feature hierarchies for robust visual tracking. *arXiv preprint arXiv:1501.04587*, 2015.

[28] Yi Wu, Jongwoo Lim, and Ming-Hsuan Yang. Online object tracking: A benchmark. In *IEEE CVPR*, pages 2411–2418, 2013.

[29] Yi Wu, Jongwoo Lim, and Ming-Hsuan Yang. Object tracking benchmark. *IEEE PAMI*, 37(9):1834–1848, 2015.

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

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

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

[33] C Lawrence Zitnick and Piotr Dollár. Edge boxes: Locating object proposals from edges. In *ECCV*, pages 391–405, 2014.