Tracking Randomly Moving Objects on Edge Box Proposals

Gao Zhu¹, Fatih Porikli¹,², and Hongdong Li¹,³
Australian National University¹ and NICTA²
ARC Centre of Excellence for Robotic Vision³
{gao.zhu,fatih.porikli,hongdong.li}@anu.edu.au

Abstract

This paper addresses the question of whether edge information enables tracking objects reliably. Human visual system is adept at tracking shapes without any texture. Motivated by this, we incorporated an object proposal mechanism that uses sparse yet informative contours to score proposals based on the number of contours they wholly enclose into a detection-by-tracking process for visual tracking. Our method is able to execute search in the entire image quickly and focus only on those high-quality candidates to test and update our discriminative classifier. Using high-quality candidates to chose better positive and negative samples, we reduce the spurious false positives and improve the tracking accuracy. Since our tracker employs only a few candidates to search the object, it has potential to use higher-dimensional features if needed. More importantly, our method can track randomly and very fast moving objects. It is robust to full occlusions as it is able to re-discover the object after occlusion. The presented tracker outperforms all the state-of-the-art methods on four largest benchmark datasets.

1. Introduction

This paper is about robust tracking without any object motion constraint where objects may be moving randomly and very fast, as well as tracking under extremely low-frame rates.

Online object tracking has been one of the most challenging tasks since the early years of computer vision research. Yet, as shown in many times on benchmark datasets [25, 20, 15, 24], the performance of the state-of-the-art methods is still far below than the practicality levels of applications even when different kinds of cues, such as color [22], motion, 3D geometry, sparsity and segmentation [17, 22], are available with the video data. On the other hand, human visual system can track remarkably well even obscure shapes or plain silhouettes without any color or texture clue [16]. A silhouette promotes quick perception when it maintains a faithful resemblance to the contours of the real-world forms.

Here, we probe into this observation and investigate how well an algorithm that utilizes only contours or edge cues can track a given target object. To this end, we incorporate a recently proposed generic object proposal that uses an edge box concept [27] into a discriminative classifier framework inspired by support vector machines [10]. Our motivation is that it is difficult to claim that human visual system establishes the location of a tracked object by exhaustively checking every possible object location hypothesis. Yet, due to its simplicity, such search window based testing is the popular choice in most tracking-by-detection approaches. In addition to its computational issues, exhaustive search...
has a propensity to confuse the oracle, i.e. the classifier and object model update mechanisms, responsible for making object-or-not decisions by burying them under a multitude of similar looking yet irrelevant (negative) samples. Any leak of such incorrect samples into the object model update would eventually lead to drift, thus poor performance. To alleviate this, one can go to the other extreme of not updating the object model at all (rigidity), which might not necessarily be the worst decision in certain cases. Of course, tracking under appearance variations requires object model to be adapted accordingly. Thus, updating the model should be done very carefully.

To help the object update mechanism to make more confident decisions, we consider a sparse (drastically limited number of) hypotheses selection scheme. We note that, conventional edge map in fact fits into this definition by being composed of sparse contours with soft-thresholded edge magnitudes. This edge map is a highly reduced space compared to the original color image, yet sufficient for a human to track an object. In other words, we scrutinize the question that supposing we can reliably track an object by simply watching sparse edge maps, how can an algorithm achieve this? In this paper we demonstrate that our tracker performs considerably and consistently better than the existing methods even though one would assume that limiting candidate hypotheses for tracking only on the edge map might make the online object tracking problem even harder. Our observations support that edge map provides sufficient information for object tracking.

The proposed edge based tracker takes advantage of the sparse yet critical information of the edge map for visual object tracking. We incorporate an edge map based object candidate proposing method to achieve this. Over the whole image, we only need to test around a hundred candidate bounding boxes. Our method has two major benefits:

- Firstly, our method can execute global search over whole image, not within just a small search window as the traditional trackers, to select candidate hypotheses in a computationally efficient manner by incorporating an edge based object proposal method to generate better-quality candidates. Thus, it can track any motion at any frame rate.
- Secondly, these high-quality candidates not only help for reducing the test space and excluding spurious hypotheses (false positives), but also offer a better training set for updating the classifier and object model, boosting the tracking performance.

We validate the above claims with extensive evaluations on benchmark datasets. To the best of our knowledge, our tracker achieves the best results on all common benchmark datasets including OTB [25], TB50 [24], VOT2014 [15] and ALOV300 [20]. We accomplish the AUC score of 54.5 on OTB (the second best: 51.7 - KCF), 45.5 on TB50 (the second best: 40.2 - KCF), 46.1 on VOT2014 (the second best: 38.9 - KCF) and 60.9 on the Moving Camera category of ALOV300 (the second best: 56.4 - KCF) while significantly improving the top scores in all cases.

2. Related Work

Providing an inclusive overview of the object tracking literature is outside the scope and capacity of this paper. We refer readers to the excellent surveys on object tracking. Here, we only compare with some relevant algorithms. We briefly examine different search schemes and then summarize recent object proposal methods.

Search Schemes in Tracking There is a wide-spectrum of styles to select which windows will be tested in a current frame to locate the target object and also update its model.

Single Window Search: Several trackers use the local window around the former object location to find the object in the current frame. Examples include the tracking on Lie groups [21], which applies iteratively a feature-motion regressor learned from perturbing local window to estimate object window in the next frame, and the mean-shift tracker [5], which uses gradient-based local optimization iteratively by employing a metric derived from the Bhattacharyya coefficient as a measure to determine the mode of the underlying similarity distribution.

Particle Based Search: In recent years, tracking algorithms [18, 26, 13] based on particle filtering has been extensively studied. Experimental results show that particle filtering outperforms other conventional state estimation method such as Kalman filter for nonlinear and non-Gaussian systems. Particle filters apply importance sampling on the previous particle states (e.g. candidate locations) within mostly a mixed number of candidates. In other words, it selects candidates windows sampling from Gaussian distributions around previous particle states depending on the likelihood of the particles. On the negative side, the random sampling is blind to the underlying texture, edge-ness, and other spatial information.

Searching for the Difficult Negatives: It is worthwhile to mention that tracking-by-detection, which allows an online trained classifier [1, 19] as an object model to distinguish the object from its surrounding background, has recently become particularly popular. Most tracking-by-detection methods update the classifier by a set of binary labeled training samples that are obtained using heuristics such as the distance of a sample from the estimated object location. One implication of this is that slight inaccuracy during tracking can lead to poorly labeled samples, thus, tracking failure. Rather than explicitly coupling to the accurate estimation of object position, [2] limits its focus on increasing the robustness to poorly labeled samples. [10] proposes directly predicting the change in object location between
frames by an online structured output SVM. Even though it produces comparably accurate tracking, it uniformly samples the state space to generate positive and negative support vectors. Such a brute force approach on a larger search window is computationally intractable.

Object Proposals in Object Detection
As shown in [12, 27], use of proposal has significantly improved the object detection benchmark along with the convolutional neural nets. Since, a subset of high-quality candidates are used for detection, object proposal methods improve not only the speed but also the accuracy by reducing false positives. The top performing detection methods [9, 23] for PASCAL VOC [8] use detection proposals.

Edge Box: [27] proposes object candidates based on the observation that the number of contours wholly enclosed by a bounding box is an indicator of the likelihood of the box containing an object. Edge Box is designed as a fast algorithm to balance between speed and proposal recall. Its 1-D feature generates remarkably accurate results.

BING: [4] made a similar observation that generic objects with well-defined closed boundary can be discriminated by looking at the norm of gradients, with a suitable resizing of their corresponding image windows into a small fixed size. They further designed a feature called binarized normed gradients (BING), which can be used for efficient objectness estimation and requires only a few atomic operations. BING is able to run around 300fps.

3. Our Framework
3.1. Tracking by Detection
Our method follows a popular Structured Support Vector Machine (SSVM) based tracking-by-detection framework. The object location is initialized manually at the first frame \( t = 1 \). Denote \( B_t \) as a bounding box at frame \( t \) and can be represented as coordinates of its four corners. Then, given a classification function \( F_{t-1} \) trained on previous frames, the current location of the object is estimated through:

\[
B_t^* = \arg \max_{B_t \in \mathcal{B}_t} F_{t-1}(B_t),
\]

where \( \mathcal{B}_t \) is a set of candidate samples at the current frame. To select samples, traditional trackers use heuristic search windows around the previously estimated object location for computational reasons. They apply each sample into a classifier. For example, a search radius of 30 pixels is used in [10]. Suppose the support vector set maintained by SSVM as \( \mathcal{V}_{t-1} \) and the classification function can be written as a weighted sum of affinities:

\[
F_{t-1}(B_t) = \sum_{B_{t-1} \in \mathcal{V}_{t-1}} w_{t-1} K(B_{t-1}, B_t),
\]
Figure 3: Histogram of maximal object center motion (measured by Euclidean distance in pixels) between two neighboring frames over sequences on dataset TB50 [24]. The maximal center distance is normalized by the shorter dimension of the object bounding box.

where \( w_{t-1} \) is a scalar weight associated with the support vector \( B_{t-1}^i \). Kernel function \( K(B_{t-1}^i, B_t) \) calculates the affinity between two feature vectors extracted from \( B_{t-1}^i \) and \( B_t \) respectively.

The classifier will then revise its model with the new location of the object to adapt possible appearance changes. To update the support vector set \( V_{t-1} \to V_t \) in an online fashion, a critical step is the selection of negative supports vector according to the following function:

\[
B_t^- = \arg \max_{B_t \in B_t \setminus B_t^*} F_{t-1}(B_t) + L(B_t, B_t^*), \tag{3}
\]

where the loss function \( L(B_t, B_t^*) = 1 - (B_t \cap B_t^*)/(B_t \cup B_t^*) \) defines on the bounding box overlap. Optimization (3) corresponds to finding such a negative training sample that locates far from the positive one (high \( L(B_t, B_t^*) \)) yet presents close appearance (high \( F_{t-1}(B_t) \)). For more details, refer to [10].

### 3.2. Proposed Method

The method proposed in this paper uses a similar framework as introduced in Section 3.1, yet we made two critical changes to it. The first change is that we recognize not all candidate bounding boxes \( B_t \in B_t \) should be treated equally (as the traditional trackers often do) since those boxes possess different object-like appearance, i.e., objectness characteristics, which should be taken into account. Secondly, we do not constrain the search radius to a small window that causes throwing so much available image information away.

To execute our changes, we take advantage of the sparse, yet critical edge information. The current frame \( I_t \) is processed into a single-line edge map \( E_t \) with soft-thresholded edge magnitude information (see the first column of Figure 2). Then, we employ an edge based object proposal algorithm [27] to generate a number of candidate bounding boxes (the second column of Figure 2) denoted as \( B_t^E \). Notice that, we have the bounding size change smoothly between neighboring frames.

Suppose the bounding box set generated by sampling only around the previous object location as \( B_t^R \) (as in traditional methods). Now we have two different sets of candidates, i.e., \( B_t^E \) and \( B_t^R \). The first one possesses object regularity while the second one is with no discriminative information. As shown in the experimental section, the choice of using only the Edge Box proposals \( B_t^E \) generates the best results, better than combining them together. This confirms our argument that object proposals not only reduce the candidate sample space but also reduce spurious false positive and improve tracking accuracy.

During the update stage, we also have different options for using \( B_t^E \) and \( B_t^R \). As validated in the experimental part, the combination of using both of them to choose negative support vector results the best performance. This can be easily explained: \( B_t^E \setminus B_t^* \) only represents other good object-like regions. By putting them as negative support vectors, we would only increase the discriminative power among objects-like candidates. While the negative sample space contains a lot more other negative sample. Thus, the most advantageous option is to augment \( B_t^E \setminus B_t^* \) with \( B_t^R \) in order to achieve the best discriminative ability.

### 3.3. Selecting Candidate Proposals

We employ an efficient edge-based object proposal method, Edge Box [27], for candidate proposals. Given an edge map \( E_t \) extracted from current frame \( I_t \), Edge Box generates a pool of sampled bounding boxes in a sliding window way, then ranks these candidates based on a very simple yet effective feature, i.e., the number of contours wholly enclosed by a bounding box. This feature can be calculated very efficiently in real-time. We refer [27] for more details.

As stated at the beginning of Section 3.2, we generate proposals in the entire image in contrast to the state-of-the-art trackers [10, 22, 13, 3, 26]. As evident in Figure 3, objects can jump to a distant location, e.g., due to camera vibration, fast object motion, or low-frame rate video. It will be very hard for existing trackers to recovery under such conditions since they always looking impose a nearby search region region. To our advantage, even the object disappears due to occlusion, our tracker is able to rediscover it. Testing of all object-like regions runs in a computationally very efficient manner.

### 3.4. Candidate Classification

Instead of Eq.1, we use the following decision function to estimate the new location of the object (the third column of Figure 2):

\[
B_t^* = \arg \max_{B_t \in B_t} F_{t-1}(B_t) + S(B_t, B_t^*). \tag{4}
\]
5

Figure 4: Pyramid of histogram of edge orientation, weighted by edge magnitude. Feature vectors generated from various cells are concatenated into a single feature vector.

\[ S(B_t, B_{t-1}^*) \] is a term representing the motion smoothness between the previous object location and the candidate box. This is important in our formulation as we are testing candidates all over the image. We use a simple weighting function in this paper: \( S(B_t, B_{t-1}^*) = \min(\sigma)\|c(B_t) - c(B_{t-1}^*)\|_1 \), where \( c(B_t) \) is the center of bounding box \( B_t \) and \( \sigma \) is a constant.

As for the feature extraction, we use a simple yet successful edge orientation based feature [7, 6] as shown in Figure 4. Bounding box is firstly divided into multiple levels, and then histogram of magnitude weighted edge orientation is build separately in each cell at different levels. We fixed the number of orientation at 9 and use 6 levels. The final feature vector is formed by concatenate all histograms together.

4. Experiments

The experimental part is organized in the following way. In the first part, we compare our method with the state-of-the-art trackers on benchmark datasets for a general performance evaluation. We also test on fast-motion related categories to put it under the spotlight to understand how well our method can handle the challenging scenarios such as fast moving objects, randomly moving objects, and tracking under low-frame-rate. Our observation is that it performs remarkably better than the existing trackers. In the second part, we analyze different components of the algorithm.

4.1. Full Benchmark Evaluations

Our algorithm (called as EBT to imply its relation to Edge Box) is tested on three large datasets: OTB [25], TB50 [24] and VOT2014 [15]. The first two datasets are composed of around 50 sequences each. They are annotated with ground truth bounding boxes and various visual attributes. TB50 is an upgraded version of OTB and contains much more challenging sequences. VOT2014 dataset contains 25 sequences and offers rotated bounding boxes as ground truths. As several baseline methods do not handle rotations, we rotated ground truths boxes back to axis-aligned bounding boxes for an objective evaluation.

Evaluation metrics and code are provided by the benchmark [25, 24]. We employ the one-pass evaluation (OPE) and use two metrics: precision plot and success plot. The former one calculates the percentage (precision score) of frames whose center location is within a certain threshold distance with the ground truth. A commonly used threshold is 20 pixels. The latter one calculates a same percentage but based on bounding box overlap threshold. A typical value is 0.5 as used in object detection evaluation [8].

4.1.1 Technical Details

**Edge Map:** We calculate the gradient map from the original image, then apply non-maximum suppression to get the final edge map (the first column of Figure 2), which is then used for the Edge Box proposals. This operation takes less than one millisecond per image generally.

**EdgeBox Proposals:** For Edge Box proposals, the translation step of sliding window is set at \( \alpha = 0.85 \) as we would like a high accurate localization. Scale and aspect values are fixed during the sliding window sampling stage. But we allow the algorithm to refine the size of bounding box slightly to best align the edges. Non-maximum suppression parameter is fixed at \( \beta = 0.8 \). The minimal objectness score is 0.01 and the maximal candidate number we used is 400.

**Other Parameters** For the SSVM part, we use exactly the same parameters as in [10]. For the smooth motion function \( S(B_t, B_{t-1}^*) \) in Eq.4, \( \sigma \) is set as the diagonal length of the initialized bounding box.

Note that, we fixed all of the parameters for all of the sequences in all datasets we tested. We did not fine tune parameter values for optimum performance.

4.1.2 Benchmark Results

We compare with all top-ranked trackers [25, 24]. We also test one very recently proposed tracker, KCF [11], which demonstrates impressively good performance in comparison to the state-of-the-art trackers. The results are summarized in Table 1 and Figure 6. The overall performance of our method is superior to all others. It consistently outperforms the state-of-the-art trackers by a large margin.

This result is not surprising as incorporating object proposal schemes has proven to be a successful strategy in single image object detection [9, 23, 8]. We believe that our proposed method is a counterpart in the tracking domain as no existing tracking methods successfully adopted such object proposal schemes before to the best of our knowledge.
Table 1: Area Under Curve (AUC) of success plot and Precision Score (20 pixels threshold) reported on various datasets (AUC/PS) corresponding to the one-pass evaluation (OPE). fps: frames per second.

| Datasets | Proposed EBT | KCF [11] | Struck [10] | SCM [26] | ASLA [13] | TLD [14] | RandomGuess |
|----------|--------------|----------|-------------|----------|-----------|-----------|-------------|
| OTB (51) | 54.5/76.8    | 51.7/74.2| 47.2/65.3   | 49.8/64.8| 43.4/52.9 | 43.4/60.1| 3.8/2.1     |
| TB50 (50)| 45.5/67.2    | 40.2/61.1| 36.3/49.9   | 35.5/47.8| 35.8/46.2 | 32.1/45.0| 3.0/1.9     |
| VOT (25) | 46.1/64.9    | 38.9/53.7| 36.5/52.3   | 32.8/42.3| 35.1/44.2 | 31.6/44.4| 3.6/2.0     |
| fps      | 4.4          | 70.9     | 4.8         | 0.3      | 3.8       | 8.8       | -           |

Table 2: Area Under Curve (AUC) of success plot and Precision Score (20 pixels threshold) reported on various fast-motion related categories (AUC/PS). IV: illumination variation, SV: scale variation, OCC: occlusion, DEF: deformation, MB: motion blur, FM: fast motion, BC: background clutters. MC: moving camera.

| Attributes | Proposed EBT | KCF [11] | Struck [10] | SCM [26] | ASLA [13] | TLD [14] |
|------------|--------------|----------|-------------|----------|-----------|-----------|
| FM (17) (OTB)| 56.5/74.8    | 46.8/61.0| 45.7/59.6   | 29.4/32.9| 24.4/24.6 | 40.7/53.2|
| MB (12)    | 58.0/76.8    | 50.8/66.0| 42.6/54.0   | 29.5/33.3| 25.1/26.8 | 39.0/49.0|
| FM (25) (TB50)| 49.1/68.9    | 39.0/54.0| 34.4/42.5   | 25.2/29.6| 25.0/29.6 | 35.6/46.5|
| MB (19)    | 51.7/73.2    | 40.6/56.4| 30.9/35.5   | 21.7/25.1| 23.3/25.5 | 39.3/49.7|
| MC (22) (ALOV300)| 60.9/68.4 | 56.4/62.9| 44.9/44.8   | 35.7/37.9| 38.6/38.8 | 56.1/67.9|

Figure 5: The performance bounds for using objectness proposals (Edge Box) on two datasets. The best candidate in each frame is used to calculate the performance. As shown, employing around 100 proposals is sufficient to reach the performance limit. Note that, all existing tracking methods perform under these bounds.

4.1.3 Tracking Fast Objects

Since our proposed method searches over the entire image, it is reasonable that our method would be good at tracking fast motion objects which could move outside of the search radius of the traditional trackers. As shown in Table 2, our proposed method significantly outperforms other trackers in the fast-motion related categories as well. An extra category Moving Camera from ALOV300 [20] is tested. This category contains many sequences that depict camera shake, sudden object motion, and abrupt jumps. ALOV300 provides a high number of short sequences with 14 visual attributes. The main source of their data is real-life videos from YouTube.

4.1.4 Tracking under Low-Frame-Rate

To test how the proposed tracker performs for very low frame rate, we additionally created a dataset, called as VOT2014+, by temporally sampling sequences at every 20 frames on VOT2014. Thus, this version of VOT2014 has 20× faster motion in comparison to the original one. Our algorithm is tested against with other top-ranked trackers, KCF and Struck. Even though both Struck and KCF rapidly degraded, our tracker retained very high performance scores (see Table 3).

4.2. Further Remarks

4.2.1 Combination of $B^E_t$ and $B^R_t$

As discussed in Section 3.2, we tested different combinations of the Edge Box proposals $B^E_t$ and candidate bounding boxes $B^R_t$ sampled around the previous object location within a radius. The results are shown in Table 4. For combinations which use only $B^E_t$ in the testing stage, we apply sampling within a 30-pixels radius to achieve a compara-
To demonstrate the difference between using an edge-based feature and an intensity-based feature, we designed another feature that basically uses the same structure of the original.

Table 4: Experimental results when different combinations of $B^E_{t}$ and $B^R_{t}$ are used.

| DB     | (Test) $B^R_{t}$ | $B^E_{t}$ | $B^E_{t} + B^R_{t}$ |
|--------|------------------|-----------|---------------------|
| TB50   | 37.0/52.3        | 40.4/58.2 | 38.1/53.4           |
| $B^R_{t}$ (Update) | 35.7/50.2        | 42.4/62.6 | 39.0/55.1           |
| $B^E_{t}$   | 35.1/50.8        | 45.5/67.2 | 39.2/57.7           |

Figure 6: Success plot and precision plot on three datasets of OPE. The number followed by the algorithm name is the area under the curve (AUC) and the precision score (PS) at the location error threshold of 20 pixels, respectively. Our method has consistently superior performance.

Figure 7: Success plot and precision plot of Fast Motion attribute on two datasets. Our method outperforms other trackers significantly.

We observed the combination of using samples from the Edge Box and local region in update stage samples only from Edge Box locations in the test stage performs the best.
one. For this, we compute the pyramid of histogram of intensity (16 bins, 6 levels), denoted as EBTphoi. The results (as shown in Figure 8) demonstrate that edge-based feature provides more powerful information for tracking.

For the number of candidate needed for successful tracking, we run experiments to obtain graphs in Figure 5, which show the upper bounding performance when various number of proposals are used. We tested performance when this number is 200, denoted as EBT200 in Figure 8. The results show that even when 200 candidates are used, our method achieves better performance than others.

4.2.3 Random-Guess Tracker and Upper Bound

We implemented a random-guess based tracker for comparison. For each frame, the random-guess tracker would randomly select one location in the frame with a size-fixed bounding box (use the initialized bounding box in the first frame). We then run this tracker for 100 times to get an average performance and include it in Table 1 and Figure 8. As apparent the random-guess tracker managed to get a few correct guesses in each sequence, in particular when the overlap threshold is very small. However, the graphs also prove that for all, including the lower values of the overlap threshold, the performance gain that our tracker achieves is very significant and statistically meaningful.

We also computed the upper bound performance curves as in Figure 8, which measure the best performance of a size-fixed (fixed to the size of the initial bounding box) tracker can achieve. As we can see, the upper bound drops quickly for larger overlap thresholds. Since there is still a gap between the current best and the ground truth best, adapting object scale during tracking has potential to provide further performance improvement.

| Struck [10] | BING (VOC) | BING (Adaption) |
|---------------|-------------|-----------------|
| TB50          | 36.3/49.9   | 30.8/47.6       | 33.7/48.0       |

4.2.4 BING Objectness Proposal

We considered using another popular object proposal method BING [4] instead of Edge Box for candidate generation. We evaluated two ways of incorporating BING into our tracker. The first one (BING VOC) uses the pretrained model on VOC dataset [8], while the second one (BING Adaption) retrain the model using the first frame of each sequence.

We tested two proposed methods on TB50 and demonstrate results in Table 5. Both performances are even below the baseline Struck. This is not a big surprise. As shown [12, 27], BING results in a relatively low recall of the objects, which is one reason for its mediocre performance.

4.3. Computational Speed

The overall computational speed of the proposed is comparable to the state-of-the-art trackers, yet we are searching over the entire image. The Edge Box proposal part takes less than 100 milliseconds and the classifier also takes same time as shown in Table 1.

5. Conclusion

This paper presented a robust method that can locate objects that are moving randomly and very fast, as well as perform tracking under extremely low-frame rates. To the best of our knowledge, our tracker achieves the best results on all common benchmark datasets including OTB [25], TB50 [24], VOT2014 [15] and ALOV300 [20].
References

[1] S. Avidan. Support vector tracking. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(8):1064–1072, 2004.

[2] B. Babenko, M.-H. Yang, and S. Belongie. Visual tracking with online multiple instance learning. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2009.

[3] B. Babenko, M.-H. Yang, and S. Belongie. Robust object tracking with online multiple instance learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2011.

[4] M. Cheng, Z. Zhang, W. Lin, and P. H. S. Torr. BING: binarized normed gradients for objectness estimation at 300fps. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2014.

[5] D. Comaniciu, V. Ramesh, and P. Meer. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2003.

[6] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2005.

[7] M. Eitz, J. Hays, and M. Alexa. How do humans sketch objects? *ACM Transactions on Graphics*, 2012.

[8] M. Everingham, S. M. A. Eslami, L. V. Gool, C. K. I. Williams, J. M. Winn, and A. Zisserman. The pascal visual object classes challenge: A retrospective. *International Journal of Computer Vision*, 2015.

[9] R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2014.

[10] S. Hare, A. Saffari, and P. H. S. Torr. Structured output tracking with kernels. In *IEEE International Conference on Computer Vision (ICCV)*, 2011.

[11] J. F. Henriques, R. Caseiro, P. Martins, and J. Batista. High-speed tracking with kernelized correlation filters. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2015.

[12] J. Hosang, R. Benenson, and B. Schiele. How good are detection proposals, really? In *British Machine Vision Conference (BMVC)*, 2014.

[13] X. Jia, H. Lu, and M.-H. Yang. Visual tracking via adaptive structural local sparse appearance model. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012.

[14] Z. Kalal, J. Matas, and K. Mikolajczyk. P-N learning: Bootstrapping binary classifiers by structural constraints. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2010.

[15] M. Kristan et al. The visual object tracking vot2014 challenge results. In *European Conference on Computer Vision (ECCV) Visual Object Tracking Challenge Workshop*, 2014.

[16] C. Malamed. *Visual Language for Designers: Principles for Creating Graphics that People Understand*. Rockport Publishers, 2011.

[17] X. Ren and J. Malik. Tracking as repeated figure/ground segmentation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2007.

[18] D. A. Ross, J. Lim, R.-S. Lin, and M.-H. Yang. Incremental learning for robust visual tracking. *International Journal on Computer Vision*, 2008.

[19] A. Saffari, M. Godec, T. Pock, C. Leistner, and H. Bischof. Online multi-class LPBoost. In *IEEE Computer Vision and Pattern Recognition (CVPR)*, 2010.

[20] A. W. M. Smeulders, D. M. Chu, R. Cucchiara, S. Calderara, A. Dehghan, and M. Shah. Visual tracking: An experimental survey. *IEEE Transaction on Pattern Analysis and Machine Intelligence*, 2014.

[21] O. Tuzel, F. Porikli, and P. Meer. Learning on Lie groups for invariant detection and tracking. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2008.

[22] S. Wang, H. Lu, F. Yang, and M.-H. Yang. Superpixel tracking. In *IEEE International Conference on Computer Vision (ICCV)*, 2011.

[23] X. Wang, M. Yang, S. Zhu, and Y. Lin. Regionlets for generic object detection. In *IEEE International Conference on Computer Vision (ICCV)*, 2013.

[24] Y. Wu, J. Lim, and M.-H. Yang. Visual Tracker Benchmark. *https://sites.google.com/site/benchmarkpami/*.

[25] Y. Wu, J. Lim, and M.-H. Yang. Online object tracking: A benchmark. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2013.

[26] W. Zhong, H. Lu, and M.-H. Yang. Robust object tracking via sparsity-based collaborative model. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012.

[27] C. L. Zitnick and P. Dollár. Edge boxes: Locating object proposals from edges. In *European Conference on Computer Vision (ECCV)*, 2014.