ROBSTRUCK: IMPROVING OCCLUSION HANDLING OF STRUCTURED TRACKING-BY-DETECTION USING ROBUST KALMAN FILTER

Ivan Bogun and Eraldo Ribeiro

Computer Vision Laboratory
Florida Institute of Technology
Melbourne, FL 32901, U.S.A.

ABSTRACT

Starting from an object's location in a video frame, tracking-by-detection methods find the location of that object in a subsequent video frame. The tracker's detection step may produce multiple false positives during short-term occlusions, which can result in loss of track. We propose a tracking-by-detection method that is robust to short-term occlusions and false positives. Here, we extend the *Struck* tracker, which is based on structured SVM, to output bounding boxes at multiple scales. In addition to predicting scale changes, we use the Robust Kalman filter to decrease false-positive detections and to increase the tracker resilience to short-time occlusions. Here, we develop a special strategy for the tracker's update step, which is designed to decrease over fitting and to allow for the tracker to recover from loss of track. We thoroughly evaluate our method on a publicly available video dataset and show that it outperforms the state-of-the-art.

Index Terms— Visual tracking, Model-free tracking, Structured support vector machine, Robust Kalman filter

1. INTRODUCTION

We address the problem of tracking an object in a video without having a pre-defined appearance model of the object (i.e., model-free tracking). Problem setting is simple: given a bounding box enclosing an object in a video frame, find the object's position in subsequent frames. However, locating the object becomes hard as its appearance changes during motion. In fact, being able to adapt to changes throughout the video is a common characteristic of state-of-the-art trackers [1, 2]. The tracker by Kalal et al. [3] adds a new appearance patch to the object model if the label from a nearest-neighbor classifier differs from temporal and spatial experts. The *Struck* tracker [4] incrementally adds new predictions as a support vectors. The trackers by [5, 6] use kernelized correlation filters.

Although appearance adaptation produces robust object models, it is insufficient to handle occluded objects. Occlusion handling in trackers present varying degree of success. Zhong et al. [7] handles occlusion by thresholding the reconstruction error between the object and a sparse combination of local object patches representing the appearance model. Patches that are deemed occluded have their associated sparse coefficient set to zero. Using a different strategy, Kalal et al. [3] assumes that the object is occluded whenever the error of forward-backward optical-flow inside the bounding box is high and a nearest-neighbor classifier returns a value smaller than a predefined threshold.

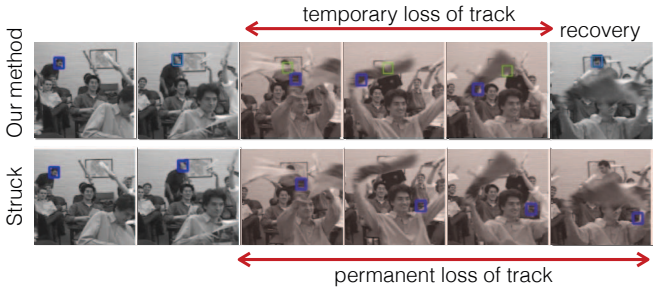


Fig. 1. **Top**: Struck with filter. **Bottom**: Struck without filtering. The green box is filter's prediction.

In this paper, we argue that threshold-based occlusion detection as employed by the above trackers cannot be made robust because it is impossible for trackers to distingsh between novel (and unseen) object appearances and occlusions. Therefore, we propose to combine the highly discriminative tracker in [4] with the robust Kalman filter [8] to increase the tracker's robustness to both false positives and short-time occlusions. We achieve that by only updating the tracker if the detector and the robust Kalman filter agree. If they do not agree, we look for the object in an extended search region. This strategy allows the tracker to recover from occlusions and incorrect detections. We evaluate our tracker on the visual-tracking benchmark [1], and compare it to the Struck [4] and to other state-of-the-art trackers [9, 10, 6]. Our results show that our tracker compares rather favourably the state-of-the-art.

Our contributions are twofold: (i) We extend the structured tracker to work on multiple scales, (ii) We show how to combine Robust Kalman filter with the tracker to handle short-time occlusions and false positives.

2. METHOD

Modern *tracking-by-detection* trackers use machine learning techniques to detect the object of interest in every video frame (e.g., the structured tracker in Struck [4], the kernelized filter in [5]). However, a location detected by these trackers in one frame is independent of the location in previous frames. This lack of spatio-temporal consistency becomes a problem when the tracker detects the wrong object at a location that is far from the correct one (i.e., a false positive). Because most trackers search for the object only in the neighborhood of the current detection, those false positives can result in an unrecoverable loss of track. Here, we solve this problem by using the Robust Kalman filter built on top of structured SVM tracker. Our tracker uses the robust Kalman filter to label detections that are inconsistent with the object's motion model. Once labeled, these false detections can be properly handled, which increases the tracker's recovery capability. The robust version of the Kalman filter is necessary because false positives are detected as outliers by the filter.

2.1. Structured tracker

Our work builds on the tracker introduced by [4]. This tracker trains a structured SVM [11] in an online manner [12]. To achieve real-time performance, the maximum number of support vectors at all times is bounded by a threshold known as a *budget* [13]. When used for tracking, structured SVM estimates the function $f : \mathcal{X} \to \mathcal{Y}$, which corresponds to finding a transformation of the bounding box in the current frame given the previous one. Let $(\mathbf{x_i}, \mathbf{y_i})_{i=1}^{N}$ be a sample from N frames, where $\mathbf{x_i}$ is a feature vector (e.g., Haar features) and $\mathbf{y_i}$ is a translation from the previous frame to the current. For each pair $(\mathbf{x_i}, \mathbf{y_i})$, the structured SVM estimates the compatibility between translation $\mathbf{y_i}$ and feature vector $\mathbf{x_i}$ by calculating a *discriminative function* $F : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$. Prediction is made by maximizing over all possible translations, $\mathcal{Y}_t$:

$$\hat{\mathbf{y}}_t = f(\mathbf{x_t}) = \arg\max_{\mathbf{y} \in \mathcal{Y}_t} F(\mathbf{x_t}, \mathbf{y}). \tag{1}$$

Let $\mathbf{\Phi}(\cdot, \cdot)$ be a feature map, then the discriminative function in simplified dual variables is given by $F(\mathbf{x}, \mathbf{y}) = \sum_{i,\bar{\mathbf{y}}} \beta_i^{\bar{\mathbf{y}}} \langle \mathbf{\Phi}(\mathbf{x_i}, \bar{\mathbf{y}}), \mathbf{\Phi}(\mathbf{x}, \mathbf{y}) \rangle$. Structured SVM then solves the following convex-optimization problem:

$$\begin{aligned} \max_{\beta} \quad & -\sum_{i,\mathbf{y}} \Delta(\mathbf{y}, \mathbf{y_i}) \beta_i^y - \frac{1}{2} \sum_{\mathbf{y},j} F(\mathbf{x_j}, \mathbf{y}) \\ \text{s. t.} \quad & \forall i, \mathbf{y} : \beta_i^{\mathbf{y}} \leq \delta(\mathbf{y}, \mathbf{y_i}) C \\ & \forall i : \sum_{\mathbf{y}} \beta_i^{\mathbf{y}} = 0, \end{aligned} \tag{2}$$

where $\delta(\mathbf{y}, \bar{\mathbf{y}}) = 1$, if $\mathbf{y} = \bar{\mathbf{y}}$, and 0 otherwise. Support vectors are defined as pairs $(\mathbf{x_i}, \mathbf{y})$ for which $\beta_i^{\mathbf{y}} \neq 0$. Equation 3 is solved by the SMO-step algorithm [14], which monotonically improves the objective function. The budget limits the number of support vectors. When this number exceeds the budget, the tracker deletes the support vectors that contribute the least to the objective function.

2.2. Robust Kalman filter

Drifting is common problem for trackers. It occurs when a tracker misdetects either due to false positive detection or occlusion. Drifting especially problematic in tracking-by-detection trackers as they sample bounding boxes only in the neighborhood of the current object location. As a result, one bad detection may lead to loss of track. We observed that often during false positives, the tracker chooses locations that are inconsistent with the object's motion (e.g., a car driving one way cannot suddenly move in the opposite way). In our method, we use motion consistency to detect false positives and occlusions. Here, our method uses a detector to build the object's appearance model online and the robust Kalman filter to spatially smooth detection results.

Unfortunately, the classical Kalman filter's sensitivity to outliers results in poor tracking performance in the event of incorrect detections. Instead, we use the robust Kalman filter, which is resilient to outliers. We follow the robust Kalman filter from [8]. Here, the gain step is replaced by a hubernization of the step. Formally, let $\mathbf{y}_t$ be the ground-truth unobserved state at time t, which is related to the state at a previous time by the following equation:

$$\mathbf{y}_t = F\mathbf{y}_{t-1} + v_t, \tag{3}$$

where F is a transition matrix, and $v_t \sim N(0, Q)$ is a Gaussian noise. Because $\mathbf{y}_t$ is unobserved, we only have its noisy estimate, $\hat{\mathbf{y}}_t$, and called it an *observed* state. Noisy estimates are related to real ones by a linear transformation:

$$\hat{\mathbf{y}}_t = H\mathbf{y}_t + \epsilon_t, \tag{4}$$

where H is a matrix that transforms latent variables into observed ones. The noise $\epsilon_t \sim N(0, R)$ is also assumed to be Gaussian. Let $\mathbf{y}_{t|s}$ be the minimum least-square error estimate of the location at frame t given locations in frames $1, \ldots, s$, while $\Sigma_{t|s}$ is the covariance of the error estimate $\mathbf{y}_{t|s}$. Then, the filter's *prediction step* is given by:

$$\mathbf{y}_{t|t-1} = F\mathbf{y}_{t-1|t-1} \tag{5}$$

$$\Sigma_{t|t-1} = F\Sigma_{t-1|t-1}F^T + Q. \tag{6}$$

The current prediction is then updated by the *correction step* by taking into account a new measurement, i.e.:

$$\mathbf{y}_{t|t} = \mathbf{y}_{t|t-1} + h(\underbrace{K\Delta\hat{\mathbf{y}}_t}_{\text{correction}}) \tag{7}$$

$$\Delta\hat{\mathbf{y}}_t = \hat{\mathbf{y}}_t - H\mathbf{y}_{t|t-1} \tag{8}$$

$$K = \Sigma_{t|t-1}H\Delta_t^{-1} \tag{9}$$

$$\Sigma_{t|t} = (\mathbb{I} - KH)\Sigma_{t|t-1} \tag{10}$$

$$\Delta_t = H\Sigma_{t|t-1}H^T + R \tag{11}$$

Hubernization is achieved by applying the function $h_b(x) = x \cdot \min\{1, \frac{b}{||x||}\}$ to the correction term. Due to hubernization, corrections with high norm are multiplied with a value smaller than 1, which weights down the correction's contribution to the result. The robust constant, b, is a threshold that the filter uses to weight down the correction whenever the correction departs from the model prediction. When $b \to \infty$ the robust Kalman filter approaches the classical Kalman filter.

2.3. Robust Kalman filter for tracking

We combine the structured tracker and the robust Kalman filter into a single tracking method as follows. In every frame, the detector locates the best position of the object. Then, we apply the filter and check if its result overlap significantly (i.e., more than 50%) with the detector's. If it does not, we assume that the object location changed unexpectedly according to the filter's motion model. In this case, the detector either: (i) made a mistake during detection (e.g., false positive) or (ii) the object was partially (or completely) occluded. Regardless of the reason, the current detection is likely to be incorrect and should not be used in the tracker's update step. Thus, for frames with small overlap, our tracker is not updated.

For the tracker to recover when it disagrees with the filter in the next frame, we calculate a discriminative function from bounding boxes sampled around the best predictions from both filter and detector. Here, we use the robust Kalman filter's ability to identify outliers and we do not change filter's position by much. As a result, the detector can recover from a mistake because the object will remain in the neighborhood of the filter's prediction. During partial or short-term occlusions, the detector may make a few consecutive mistakes or it might be simply unreliable. We consider a detection unreliable if the overlap between the best detection of both the filter and the detector is lower then 50%. In this case, we set the value of the robust constant to half of the value originally set by the tracker. Once the overlap becomes larger than 50% again, the robust constant is set back to its original value. While such a strategy cannot address long-term occlusions, our experiments show that it performs well for short-term occlusions.

3. IMPLEMENTATION

Our tracker implementation is similar to that of Hare et al. [4], with the same SVM parameters such as budget and soft-margin constant, C [1]. In contrast, we use the robust Kalman filter with a constant-velocity model. Thus, the unobserved state at frame t is given by $\mathbf{y}_t = \left[x_t^{tl}, y_t^{tl}, x_t^{br}, y_t^{br}, v_t^x, v_t^y\right]$, where $\left[x_t^{tl}, y_t^{tl}\right]$ and $\left[x_t^{br}, y_t^{br}\right]$ are pixel locations of the top-left and bottom-right corners of the bounding box, and $[v_t^x, v_t^y]$ are velocities. Locations of the two corners of the bounding box are observed, $\hat{\mathbf{y}}_t = \left[x_t^{tl}, y_t^{tl}, x_t^{br}, y_t^{br}\right]$. The constant-velocity model means that location of the corners in time change as follows:

$$x_t^{tl} = x_{t-1}^{tl} + v_{t-1}^x \quad \text{and} \quad y_t^{tl} = y_{t-1}^{tl} + v_{t-1}^y, \tag{12}$$

$$x_t^{br} = x_{t-1}^{br} + v_{t-1}^x \quad \text{and} \quad y_t^{br} = y_{t-1}^{br} + v_{t-1}^y. \tag{13}$$

3.1. Role of the robust constant

The robust constant, b, serves as a threshold for identifying outlier bounding boxes. If the correction term exceeds the threshold, its contribution is weighted down, which, in turn increases the contribution of the filter's motion model. We set parameter $b = 10$ during tracking. However, if tracker and filter disagree, we decrease $b \to \frac{b}{2}$ to indicate that the detected outlier cannot be trusted. This is done to avoid fast drifting of the bounding boxes when the detector makes mistakes for a few consecutive frames.

3.2. Kernel

Our tracker uses the intersection kernel, which is implemented using a fast, exact algorithm by Maji et al. [15].

3.3. Features

We use a combination of HoG features and histograms, similar to the ones used in [4]. HoG features are extracted by resizing the image such that height and width of the bounding box are both of 32 pixels. HoG parameters are as follows: 32×32 window size, 16×16 block size, 4×4 cell size, 16×16 block stride, and the number of bins per histogram is 8. Histograms are calculated using 16 bins whereas we set $L = 4$. The length of the feature vector is 992, i.e., the sum of size of HoG features (512) and histogram ones (480).

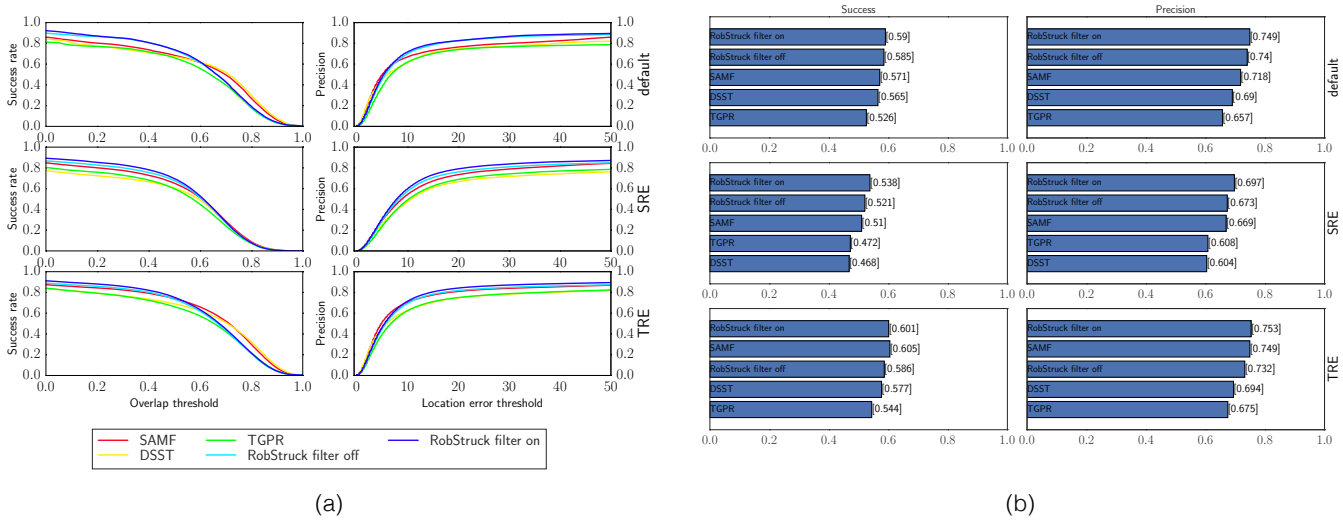
3.4. Tracker update

The structured tracker in [4] updates for every frame. We observed that this strategy can lead to overfitting because, when the tracker is updated, a new positive support vector is created. As the budget fixes the total number of vectors, this new support vector can remove an existing one. As a result, a set of positive support vectors may contain very similar vectors, which will impair the tracker's discrimination ability. Instead, we update the tracker every n frames and only if the tracker agrees with the filter. This less-frequent updating leads to a more balanced set of support vectors. In addition, by restricting the update only when the filter agrees with the tracker allows for updating to occur only when tracker's detection is the most reliable (i.e., it is less likely to make a mistake).

3.5. Speed

RobStruck's running speed is 1.75 frames per second on a single-threaded single-core 2.4 GHz processor. The addition

[1] $B = 100, C = 100$



(a) (b)

Fig. 2. (a) Success and precision curves on the dataset [1]. (b) Results on [1] dataset averaged across all tracker runs in all experiments. [1]. Abbreviations used are from : SAMF [9], TGPR [10], DSST [6]

of the filter requires a constant number of operations per frame when the filter agrees with the tracker. In cases when the tracker disagrees with the filter, the running time increases by a factor of two per frame.

4. RESULTS

Dataset. Dataset [1] contains 50 videos with a total of 29,000 frames. Tracked subjects include humans, cars, animals, and toys. Benchmark contains an evaluation protocol which is discussed next.

Evaluation protocol. The two common metrics for tracking evaluation are: success rate and precision. Precision measures the average distance between the track and the ground truth while success rate measures overlap.

Experiment types. Evaluation protocol includes default, temporal (TRE) and spatial robustness experiments (SRE). In SRE, the initial bounding box in the first frame is spatially perturbed by changing location and it's dimensions. In TRE, the bounding box is perturbed temporally. In total, we run the tracker 32 times on each video.

Results. We evaluated our tracker with and without the Kalman filter, and compared results to the trackers TGPR [10], DSST [6] and SAMF [9] [2]. Results in the form of precision and success plots are shown in Figure 2(a) as well as averaged histograms in the Figure 2(b).

Our results show that our improved structured tracker (even without filtering) outperforms existing trackers in all experiments. This advantage can be partially explained by the features being used. The best trackers in the VOT2014 challenge used HoG features with kernelized correlation filters. Using HoG features closes the performance gap. In our experiments, we found that a combination of HoG features and histograms further improves results.

The tracker with the filter improves tracking accuracy. The most striking difference is seen in the SRE experiments, in which the bounding box is moved to emulate "noise" in the data. Because the initial bounding box is inaccurate, the tracker is more likely to detect a false positive. Despite the inaccuracy of the bounding box, our tracker recovers promptly.

5. CONCLUSION

Thresholding is a popular technique for occlusion detection in visual tracking [7, 3]. Setting a threshold for occlusion parameter in model-free tracking is hard as little information about the object being tracked is available. Also, sensitivity of the threshold to the appearance representation might hinder tracker's performance. In this paper, we proposed to use the robust Kalman filter to detect and recover from short-time occlusions and incorrect detection. We also extended the structured tracker to work on multiple scales, and made it less susceptible to overfitting. We showed that the resulting trackers (with and without filtering) outperformed the state-of-the-art trackers on Wu et al. [1] benchmark. We conclude that adding the robust Kalman filter to tracking-by-detection allows the tracker to limit appearance updates to happen only for locations where the tracker's detection is the most reliable. This simple but effective strategy significantly increase the discriminative power of the appearance model learned by the tracker. Current limitations of our approach include the setting of the robust constant b and the filter's reliance on the constant-velocity motion model.

[2] DSST, SAMF took first and second places respectively on VOT2014 benchmark

References

[1] Y. Wu, J. Lim, and M.-H. Yang, "Online object tracking: A benchmark," in *Computer Vision and Pattern Recognition (CVPR), 2013 IEEE Conference on*. IEEE, 2013, pp. 2411–2418.

[2] A. Smeulders, D. Chu, R. Cucchiara, and S. Calderara, "Visual tracking: An experimental survey," *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

[3] Z. Kalal, K. Mikolajczyk, and J. Matas, "Tracking-learning-detection," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 34, no. 7, pp. 1409–1422, 2012.

[4] S. Hare, A. Saffari, and P. H. Torr, "Struck: Structured output tracking with kernels," in *Computer Vision (ICCV), 2011 IEEE International Conference on*. IEEE, 2011, pp. 263–270.

[5] J. F. Henriques, R. Caseiro, P. Martins, and J. Batista, "Exploiting the circulant structure of tracking-by-detection with kernels," in *Computer Vision–ECCV 2012*. Springer, 2012, pp. 702–715.

[6] M. Danelljan, G. Häger, F. S. Khan, and M. Felsberg, "Accurate scale estimation for robust visual tracking," in *Proceedings of the British Machine Vision Conference BMVC*, 2014.

[7] W. Zhong, H. Lu, and M.-H. Yang, "Robust object tracking via sparsity-based collaborative model," in *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*. IEEE, 2012, pp. 1838–1845.

[8] P. Ruckdeschel, "Optimally robust kalman filtering," 2010. [Online]. Available: http://d-nb.info/102738966X/34

[9] Y. Li and J. Zhu, "A scale adaptive kernel correlation filter tracker with feature integration," in *Computer Vision-ECCV 2014 Workshops*. Springer, 2014, pp. 254–265.

[10] J. Gao, H. Ling, W. Hu, and J. Xing, "Transfer learning based visual tracking with gaussian processes regression," in *Computer Vision–ECCV 2014*. Springer, 2014, pp. 188–203.

[11] I. Tsochantaridis, T. Hofmann, T. Joachims, and Y. Altun, "Support vector machine learning for interdependent and structured output spaces," in *Proceedings of the twenty-first international conference on Machine learning*. ACM, 2004, p. 104.

[12] A. Bordes, N. Usunier, and L. Bottou, "Sequence labelling svms trained in one pass," in *Machine Learning and Knowledge Discovery in Databases*. Springer, 2008, pp. 146–161.

[13] Z. Wang, K. Crammer, and S. Vucetic, "Multi-class pegasos on a budget," in *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, 2010, pp. 1143–1150.

[14] A. Bordes, L. Bottou, P. Gallinari, and J. Weston, "Solving multiclass support vector machines with larank," in *Proceedings of the 24th international conference on Machine learning*. ACM, 2007, pp. 89–96.

[15] S. Maji, A. C. Berg, and J. Malik, "Efficient classification for additive kernel svms," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 35, no. 1, pp. 66–77, 2013.