Subpixel-Precise Tracking of Rigid Objects in Real-time

Tobias Böttger, Markus Ulrich, and Carsten Steger

MVTec Software GmbH, Munich, Germany
{boettger,ulrich,steger}@mvtec.com
http://www.mvtec.com

Abstract. We present a novel object tracking scheme that can track rigid objects in real time. The approach uses subpixel-precise image edges to track objects with high accuracy. It can determine the object position, scale, and rotation with subpixel-precision at around 80fps. The tracker returns a reliable score for each frame and is capable of self diagnosing a tracking failure. Furthermore, the choice of the similarity measure makes the approach inherently robust against occlusion, clutter, and nonlinear illumination changes. We evaluate the method on sequences from rigid objects from the OTB-2015 and VOT2016 dataset and discuss its performance. The evaluation shows that the tracker is more accurate than state-of-the-art real-time trackers while being equally robust.

Keywords: Visual object tracking, real-time tracking, template matching

1 Introduction

Visual object tracking is a fundamental problem in computer vision that is concerned with estimating the 2D pose of an object in a video sequence. It has a wide range of applications, such as robotics, human-computer interaction, and visual surveillance [3,6,9].

To cover the large amount of applications, the performance of trackers is usually evaluated on very diverse, publicly available, benchmarks, such as VOT2016 [7], OTB-2015 [17], or MOT16 [10]. Although the videos in the benchmarks are very diverse, they do not necessarily cover specific applications, but rather try to be as general as possible. This leads to the fact that, in general, trackers are optimized towards their generalization capabilities and not to a specific application [3,4,11]. Furthermore, the objects in these datasets are manually labeled with either axis aligned or oriented bounding boxes and the accuracy of a tracker is measured by its bounding box overlap [8]. Hence, trackers with subpixel precise localization or ones not restricted to oriented or axis-aligned rectangles do not necessarily have higher overlap scores in the benchmarks. Nevertheless, many industrial applications such as autonomous driving or the visual monitoring of industrial production processes require a good localization accuracy in real-time.

For example, when picking an object from a conveyor belt with a robot, the bounding box of the object is not sufficient.

We present a real-time capable tracker that is able to determine the similarity transformation of a rigid object between frames. We leverage the fact that image edges can be determined with subpixel-precision to obtain a subpixel-precise object localization and do not restrict the object to a bounding box. The tracker returns a reliable score for each frame and is capable of self diagnosing a tracking failure. The two examples sequences in Fig. 1. show the localization quality of our approach and the fact that it is virtually drift-free for a rigid object in a sequence over 3000 frames. In the experiments section, we evaluate the method on further sequences from the OTB-2015 and VOT2016 dataset and comment on the performance characteristics of the presented approach.

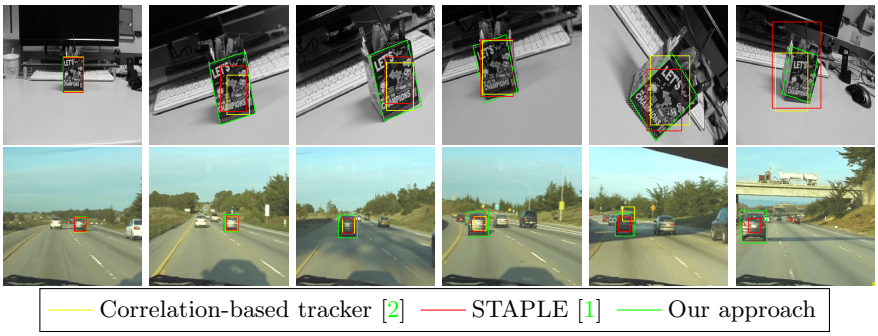


Fig. 1. `Vase` and `Car24` from the OTB-2015 [17] benchmark. We compare our approach to two equally fast, state-of-the-art trackers: STAPLE [1] and a scale adaptive correlation tracker [2]. Our approach is able to accurately determine the position, scale, *and* rotation of the objects, as opposed to just the axis aligned bounding box. The tracker is virtually drift-free in the second sequence (which has over 3000 frames). All three trackers run in real-time at 100 fps on both sequences.

2 Related Work

Within visual object tracking, immense progress has been made in recent years. For example, the best performing tracker in the VOT-2014 challenge was only ranked 35th in the 2016 challenge, with around half of the expected average overlap (AEO) of the VOT2016 winner [7]. The great gain in performance is mostly due to the widespread adoption of discriminative learning methods such as discriminative correlation filters with complex features [3,5,6,18] and deep convolutional neural networks (CNNs) for tracking [4,11,16] (2015 VOT winner and the 2016 baseline for runner up). Although CNN-based trackers show impressive generalization properties and can cope with diverse sequences, most

approaches are restricted to axis-aligned bounding boxes [16] and their accuracy is not on par with their robustness [7]. Furthermore, the computational complexity of many approaches is infeasible and only few approaches are real-time capable with a high performance GPU [4], which is not an option in many industrial applications.

In terms of speed, very robust trackers have emerged in the last few years which are real-time capable. Most of them build on discriminative correlation filters and extensions thereof [1,3,5,6,18]. For example, the STAPLE tracker [1] combines a HOG-based correlation filter with a model based on color statistics and achieved the best real-time performance at the VOT-2016 challenge [7]. Danelljan *et al.* [3] go beyond the ordinary discriminative correlation framework and train continuous convolution filters. Their approach performs on par with trackers based on CNNs and can be extended to subpixel-precise feature point tracking. Unfortunately, all of the mentioned approaches are restricted to axis aligned bounding boxes.

Similar to our approach, Lepetit and Fua [9] present a tracker that estimates the pose of rigid objects. Their keypoint-based recognition system is robust to occlusion and clutter, but requires an extensive offline training phase to generate the tracking model. In contrast to our approach, their tracking is restricted to textured objects that exhibit sufficient keypoints for reliable tracking.

As opposed to the above mentioned approaches, our approach is capable of estimating the position, scale, *and* rotation of an arbitrarily shaped object in real-time and does not require an extensive offline training nor is it restricted to textured objects.

3 Shape-Based Tracking

Our tracking approach builds on the efficient shape-based object recognition technique of Steger [13]. In the first frame, a shape-based model is generated from the arbitrarily shaped ROI of the detected or marked object. The model is used to determine the optimum object pose in the subsequent frames. After each successful tracking step, the model is updated and unstable points are filtered out and new points are added. The three steps (1) model generation, (2) model localization and, (3) model update are explained in more detail in the following section. Furthermore, we describe how the approach is made efficient and able to track most objects in the VOT2016 [7] and OTB-2015 [17] datasets at around 80fps without using the GPU on an IntelCore i7-4810 CPU @2.8GHz with 16GB of RAM with Windows 7 (x64).

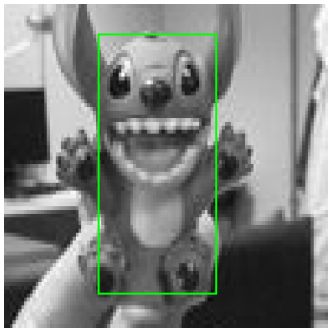
3.1 Model Generation

In the first frame, the tracking model $\mathcal{M}$ is constructed from the ROI of the automatically detected or manually marked object. The model consists of a set of n points $p_i = (x_i, y_i)^T$ and their corresponding direction vectors $d_i = (t_i, u_i)^T$:

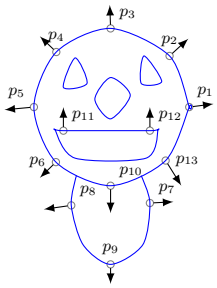
$$\mathcal{M} = \{(p_i, d_i) \in \mathbb{R}^2 \times \mathbb{R}^2, \text{for } i = 1, \ldots, n\}. \tag{1}$$

In a first step, point candidates are extracted by applying a threshold on the Sobel filter edge amplitude of the input ROI. To thin out the number of points, non-maximum suppression is applied with automatically estimated thresholds, see the patent [14] for details. The remaining points can then be refined to subpixel-precision which is described in more detail in Chapter 3.3 of [12]. The coordinates of the model points are all expressed relative to an arbitrary reference point. We use the center of the bounding box for simplicity. An exemplary model is displayed in Fig. 2.

Please note, that we do not have a training phase of our model. The model consists of a single set of points and their directions. The model transformation to different poses is done on the fly in the localization step.



(a) Example Image from the OTB-2015 benchmark [17]



(b) Model Points p_i and their direction vectors

Fig. 2. The model in (b) is generated from the ROI in the first input frame displayed in (a). The model point are generated from non-maximum suppressed edges with a high enough edge amplitude (see [14]). The model in (b) is simplified for better visualization. The real model has over 400 points.

3.2 Model Localization

The model localization essentially amounts to finding the best matching candidate within the target image in a template matching framework. Hence, we compare a transformed model to the target image at a particular location by a similarity measure. By setting a minimal required similarity, we are able to avoid a very large number of computations.

In a first step, we calculate a direction vector for each pixel within the current frame and identify them as $e_{x,y} = (v_{x,y}, w_{x,y})$. We can then evaluate the similarity of the tracking model $\mathcal{M}$ to the current frame at various image locations and for different transformations of the tracking model. The location and transformation parameters with the highest similarities are the most probable

object locations. The similarity transformation of a model point is given by:

$$p'_i = \underbrace{\begin{pmatrix} \sigma\cos\theta & -\sigma\sin\theta \\ \sigma\sin\theta & \sigma\cos\theta \end{pmatrix}}_{T_{\theta,\sigma}} p_i + \begin{pmatrix} x_t \\ y_t \end{pmatrix}, \tag{2}$$

where θ and σ are the rotation and scale parameters, respectively. Similarly, the transformed direction vectors are obtained by

$$d'_i = (T_{\theta,\sigma}^{-1})^T d_i. \tag{3}$$

As similarity measure we use the normalized sum of dot products of the normalized direction vectors of the transformed model and the target image:

$$s(x_t, y_t, \theta, \sigma)_{\mathcal{M}} = \frac{1}{n} \left| \sum_{i=1}^{n} \frac{\langle d'_i, e_{p'_i} \rangle}{\|d'_i\| \cdot \|e_{p'_i}\|} \right|, \tag{4}$$

with $s : \mathbb{R}^4 \to [0, 1]$. The similarity measure is robust to occlusion, clutter, non-linear illumination changes, and a moderate amount of defocusing [13]. The robustness to non-linear illumination change comes from the fact that all direction vectors are scaled to unit-length. The robustness to occlusion comes from the fact that missing points in the target image will, on average, contribute nothing to the sum of (4). Similarly, clutter lines or points in the target image do not only need to coincide with the sparse set of model points, but also need to have a similar direction vectors to contribute to the similarity.

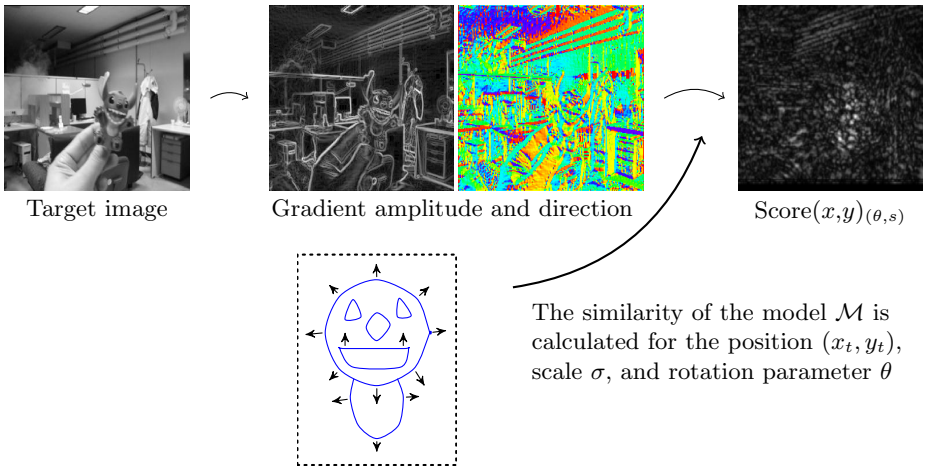


Fig. 3. In a first step, the gradient amplitudes and direction of the target image are calculated. The amplitudes are required for the subpixel-precise refinement of the object position. The maximum similarity of the model $\mathcal{M}$ from Fig. 2. is calculated for the position, scale and angle within the discretized 4d search space.

The localization process is visualized in the flowchart of Fig. 3. At this point the optimal position, angle, and scale are determined with pixel accuracy. In the following subsection, we display how the 4d optima $(\tilde{x}_t, \tilde{y}_t, \tilde{\theta}, \tilde{\sigma})$ is refined to subpixel accuracy.

3.3 Subpixel-precise Refinement

The accuracy of the localization step depends on the chosen discretization of σ and θ as well as the pixel resolution. To refine the match, we apply the concept of the 4d facet model. We approximate the 4d parameter space by calculating a second order Taylor polynomial around the $3 \times 3 \times 3 \times 3$ best match and extracting the maximum of this polynomial [13].

To further improve the localization accuracy and the robustness of the tracking to small model deformations and transformations that cannot be captured by a similarity transformation, we use a modified version of the least squares refinement described in [15]. The least-squares refinement assumes a good initial approximation of the current transformation and improves the global similarity transformation for all points. For each model point p_i, the best point match in the direction of $\pm d'_i$ is determined. The concept is displayed in Fig. 4 and explained in more detail in [15]. In contrast to [15], we do not restrict the length of the search line to 1, but rather allow an arbitrarily long search line. We found that a single least squares iteration was sufficient for most tracking sequences.

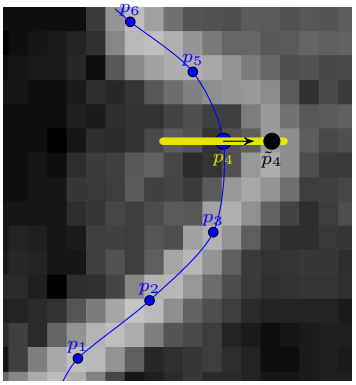


Fig. 4. The model is adapted to transformations that cannot be captured by a similarity transformation of the complete model. After the optimal model position, scale, and rotation have been determined, each point searches for its best match along a 1d search line perpendicular to its image tangent. The length of the search line is variable, but has a significant impact on the runtime.

3.4 Model Update

After we have successfully refined the object pose to subpixel-precision, we conduct one final search for the best corresponding target image point for each model point in the direction of $\pm d_i'$. This time we do not update the global similarity transformation of the model, but rather update the relative positions of the points themselves. This improves how well the model will fit to the target image at future timesteps.

In the example shown in Fig. 4, we shift p_4 towards the best match $\tilde{p}_4$ that is found along the yellow line. We regularize the model update with a parameter λ to be more robust to noisy object deformations. At frame t we update each point p_i^t with its best match $\tilde{p}_i^t$ such that:

$$p_4^{t+1} = p_4^t + \lambda \tilde{p}_4^t. \tag{5}$$

The update step does not only allow our approach to capture small model deformations, but also weakens the restriction of our approach to similarity transformations of the model, consequently projective transformations that increment over time may be captured by locally deforming the model points.

Please note that in tracking settings the model update step always needs to find the balance between keeping the localization accuracy high and generalizing well to new representations of the model. In our approach, too large parameter values of λ may add drift and can lead to a degeneration of the tracking model if no extra care is taken. Nevertheless, since the model transformation is determined with subpixel-precision, even long sequences with over 3000 frames, like the one in the example displayed in Fig. 1., do not drift significantly.

During the tracking process, we further monitor how often every model point is found. This step enables us to identify points that are not significantly contributing to the model localization and to remove them. These points may have either emerged from poorly initialized points in the first frame or by parts of the object becoming occluded or changing in later frames. To prevent us from deleting all points and degenerating the model, we sample new points in sparse areas of the model after a successful tracking step. This allows us to capture newly emerging object edges. The example in Fig. 5. shows how the model update helps to capture deformations of the object.

3.5 Implementation Details

In tracking we do not need to search for the model in the target image exhaustively in each frame. To reduce the workload, we restrict the possible parameter values of $(\theta, \sigma) \in [\theta_c - 0.1, \theta_c + 0.1] \times [\sigma_c - 0.2, \sigma_c + 0.2]$, where θ_c and σ_c refer to the current object rotation and scale. Furthermore, for the translation, we define a circular search region with a radius of $1/2$ of the object diagonal. Although the search space was adequate for all of the test sequences, the size of the search region may be increased for fast moving objects.

If no parameter set of $(x_t, y_t, \theta, \sigma)$ that has a score $> s_{\min}$ is found in a frame, we increase the search region and the parameter ranges of (θ, σ) for the

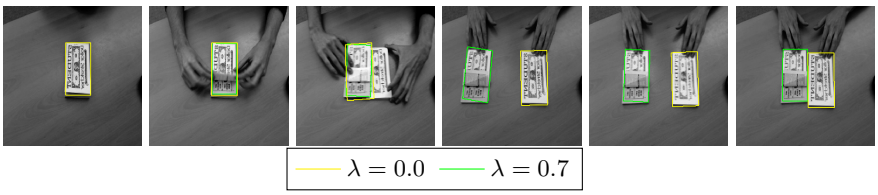


Fig. 5. Sequence `Coupon` from OTB-2015. If no model update is performed ($\lambda = 0.0$) the tracker jumps to the wrong dollar note when the folded top note is moved. If the model update parameter λ is set high enough, the model is updated when the note is folded and the tracker succeeds.

next frames successively. As soon as we find the object again, we reset the search parameters to their initial value. To prevent the workload from becoming too large, we decrease the discretization of the search space when it becomes bigger. Although this decreases the accuracy, it gives us the chance to re-detect lost objects and improve the accuracy in the subsequent frames.

To achieve further speed-ups, we stop calculating a score for a model transformation as soon as it cannot reach a predefined minimal score s_{min} anymore. To obtain an even larger speed-up it is possible to be even stricter, please refer to [15] for further details.

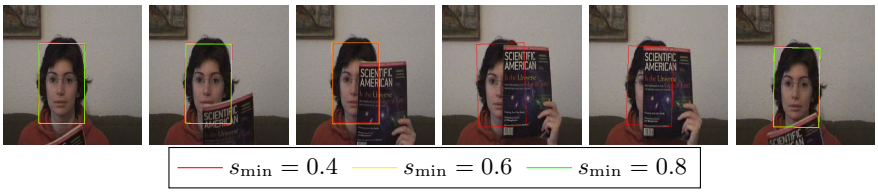


Fig. 6. The results for 3 different parameters of s_{min} are displayed for the `FaceOcc1` sequence from OTB-2015. The value of s_{min} is an indicator of how much the object is allowed to be occluded. Lower values improve the robustness to occlusion but also require more time, since more score values need to be computed. For $s_{\text{min}} = 0.8$, the object is not detected for the third to fifth image, while for $s_{\text{min}} = 0.6$ it is lost for the fourth and fifth image. However, all approaches recover when the occlusion ends.

The parameter s_{min} gives a good estimation of the allowed object occlusion. If half of the model points are occluded in the target image, the maximum score that can be obtained is 0.5. This is displayed in Fig. 6., where the face is essentially occluded by over 50% and hence, only values of $s_{\text{min}} < 0.5$ are able to track the object through all frames. Please note that a low value of s_{min} increases the number of points for which the score needs to be calculated and hence has a negative impact on the runtime.

Further speed-ups can be obtained by only using a fixed number of points for tracking the object. Before each localization step, a random subset of points

is selected from the model $\mathcal{M}$ and used for tracking. Although some accuracy may be lost, the execution time can be reduced.

4 Experiments

The restriction to rigid objects and subpixel-precise localization makes it difficult to compare the approach to the vast majority of existing schemes in general. First of all, the data of existing benchmarks, such as VOT2016 [7] and OTB-2015 [17], are not annotated with sufficient accuracy and focus on robust, generalizable tracking. Hence, we focus our evaluation on selected sequences of rigid objects from both datasets and point out the strengths and weaknesses of the proposed approach.

To get a fair comparison, we compare our method to the state of the art STAPLE tracker [1], which was the best real-time tracker at the VOT-2016 challenge [7], and a very fast correlation filter tracker with scale adaption, based on [2]. We evaluate the average bounding box overlap [17] on a selection of rigid objects from both the OTB-2015 and VOT-2016 datasets in Fig. 7. Please note that the ground truth data is mostly only labeled as axis aligned bounding boxes which puts a heavy bias on the obtained overlaps. For example, for the sequence `Vase` we visually clearly outperform both approaches, as is seen nicely in Fig. 1. Nevertheless, the bounding box abstraction lets the overlap drop very low.

To measure the robustness, we do not use the VOT-2016 measures, but rather evaluate if the tracker is successfully tracking the target in the last few frames (bounding box overlap $> 50\%$). Here STAPLE (22/26) and our approach (21/26) perform equally well, while the correlation based approach drops off (13/26). In the following we will discuss individual sequences in more detail.

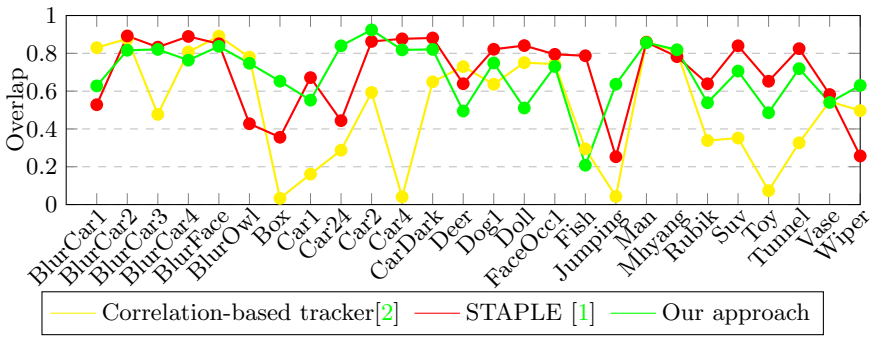


Fig. 7. The average bounding box overlap for rigid sequences within the VOT2016 and OTB-2015 datasets. We compare our overlap scores to STAPLE [1] and a scale adaptive correlation tracker based on [2].

The choice of the similarity measure in (4) makes the tracker inherently robust to nonlinear illumination changes. This is visualized for two sequences in

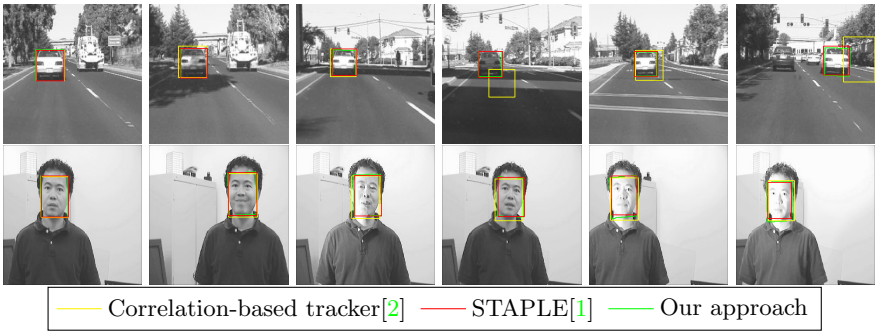


Fig. 8. Our tracking scheme is inherently robust to illumination changes because of the similarity measure we use in the localization step (4). The method performs comparable to STAPLE and outperforms the scale adaptive correlation tracker, which fails in the first sequence, in terms of accuracy.

Fig. 8.. The tracker localization quality is unaffected by the car driving under the bridge or the light being turned on and off in the second sequence. However, very strong changes of the illumination can make it difficult to segment the edge orientation of the target image robustly and lead to tracking failure, which is what happens in the sequence `Fish`.

The fact that our approach is essentially a local object detector with a meaningful score allows our framework to detect tracking failure reliably and recover from complete object occlusion. In Fig. 9, the object is completely occluded in the middle of the sequence and hence the STAPLE and correlation-based tracker start adapting their filters to the foreground. Neither of the approaches is able to recover from the complete occlusion. Our approach, on the other hand, detects the tracking failure and is able to re-detect the object when it reemerges.

The fact that our approach searches for the best similarity transformation between the frames leads to the fact that our approach is weak in sequences where the object has strong local deformations. Furthermore, strong camera motion and image blur can make it fail. In the sequence in Fig. 10, the object is

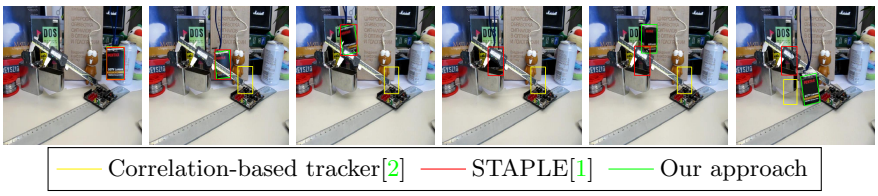


Fig. 9. A further advantage of our approach is the possibility of self-diagnosing when the object is lost. The score is a reliable indicator of how much of the model is visible. In the sequence `Box` from OTB-2015, all three trackers fail when the object is strongly occluded. Nevertheless, our approach recovers when the occluded object reappears.

lost whenever the camera motion is too strong. Fortunately, the object is always re-detected when the camera motion stops and the edges become clear enough. In the respective frame, the STAPLE tracker fails near the end when the camera motion is extremely high.

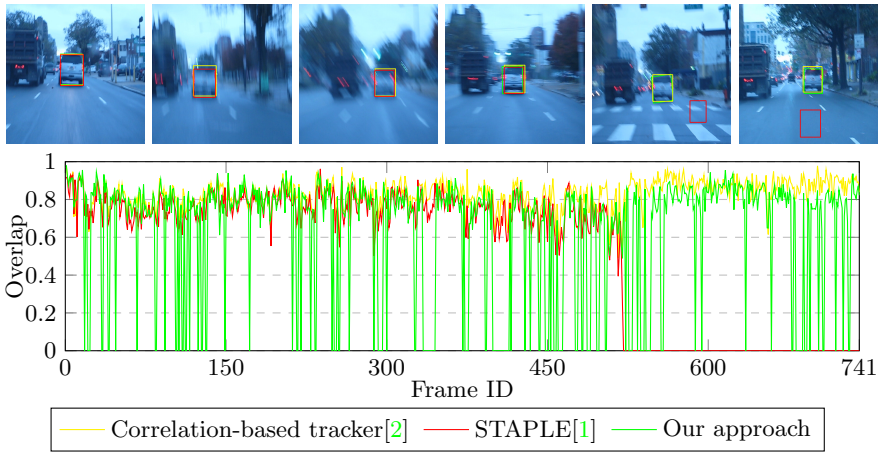


Fig. 10. Edge-based tracking has difficulties when the image blur becomes too large. Neighboring edges may merge into each other or disappear completely. This becomes evident in the sequences `BlurCar1` sequence from OTB-2015 [17]. Our tracker loses the object when the camera motion is too large. Nevertheless, in both sequences the tracker is always able to recover and finishes the sequence with a very good localization. The STAPLE tracker loses the track at frame 543 and does not recover.

5 Conclusion

In this paper, we have proposed an efficient object tracker that is able to determine the position, scale *and* rotation of a rigid objects in various different sequences with high accuracy. We validated our framework on a rigid subset of the VOT-2016 [7] and OTB-2015 [17] datasets and were able to perform on par with real-time state-of-the art approaches in terms of robustness. As opposed to the existing schemes, our approach is more accurate in terms of localization. On the one hand this is due to the subpixel-precise refinement of the object pose and, on the other hand, due to estimating the object rotation.

Unfortunately, the label data of the existing benchmarks is restricted to axis-aligned and oriented bounding boxes, which makes it difficult to quantize the localization gains in the established performance measures. A subpixel-precise tracking dataset and evaluation framework that is not restricted to bounding boxes would be very helpful for future evaluations.

References

1. Luca Bertinetto, Jack Valmadre, Stuart Golodetz, Ondrej Miksik, and Philip H. S. Torr. Staple: Complementary learners for real-time tracking. In *IEEE CVPR*, pages 1401–1409, 2016. 2, 3, 9, 10, 11
2. Martin Danelljan, Gustav Häger, Fahad Shahbaz Khan, and Michael Felsberg. Accurate scale estimation for robust visual tracking. In *BMVC*, 2014. 2, 9, 10, 11
3. Martin Danelljan, Andreas Robinson, Fahad Shahbaz Khan, and Michael Felsberg. Beyond correlation filters: Learning continuous convolution operators for visual tracking. In *ECCV*, pages 472–488, 2016. 1, 2, 3
4. David Held, Sebastian Thrun, and Silvio Savarese. Learning to track at 100 FPS with deep regression networks. In *ECCV*, pages 749–765, 2016. 1, 2, 3
5. 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, 2012. 2, 3
6. João F. Henriques, Rui Caseiro, Pedro Martins, and Jorge Batista. High-speed tracking with kernelized correlation filters. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 37(3):583–596, 2015. 1, 2, 3
7. Matej Kristan, Ales Leonardis, Jiri Matas, Michael Felsberg, Roman P. Pflugfelder, Luka Cehovin, Tomás Vojír, and Gustav Häger. The visual object tracking VOT2016 challenge results. In *ECCV Workshops*, pages 777–823, 2016. 1, 2, 3, 9, 11
8. Matej Kristan, Jiri Matas, Ales Leonardis, Tomás Vojír, Roman P. Pflugfelder, Gustavo Fernández, Georg Nebehay, Fatih Porikli, and Luka Cehovin. A novel performance evaluation methodology for single-target trackers. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 38(11):2137–2155, 2016. 1
9. Vincent Lepetit and Pascal Fua. Keypoint recognition using randomized trees. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(9):1465–1479, 2006. 1, 3
10. Anton Milan, Laura Leal-Taixé, Ian D. Reid, Stefan Roth, and Konrad Schindler. MOT16: A benchmark for multi-object tracking. *CoRR*, abs/1603.00831, 2016. 1
11. Hyeonseob Nam and Bohyung Han. Learning multi-domain convolutional neural networks for visual tracking. In *IEEE CVPR*, pages 4293–4302, 2016. 1, 2
12. Carsten Steger. An unbiased detector of curvilinear structures. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(2):113–125, 1998. 4
13. Carsten Steger. Similarity measures for occlusion, clutter, and illumination invariant object recognition. In *Pattern Recognition, 23rd DAGM-Symposium, Proceedings*, pages 148–154, 2001. 3, 5, 6
14. M. Ulrich and C. Steger. System and methods for automatic parameter determination in machine vision, May 31 2011. US Patent 7,953,290. 4
15. Markus Ulrich and Carsten Steger. Performance evaluation of 2d object recognition techniques. Technical Report PF–2002–01, Lehrstuhl für Photogrammetrie und Fernerkundung, Technische Universität München, 2002. 6, 8
16. Lijun Wang, Wanli Ouyang, Xiaogang Wang, and Huchuan Lu. Visual tracking with fully convolutional networks. In *IEEE ICCV*, pages 3119–3127, 2015. 2
17. Yi Wu, Jongwoo Lim, and Ming-Hsuan Yang. Object tracking benchmark. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 37(9):1834–1848, 2015. 1, 2, 3, 4, 9, 11
18. Kaihua Zhang, Lei Zhang, Qingshan Liu, David Zhang, and Ming-Hsuan Yang. Fast visual tracking via dense spatio-temporal context learning. In *ECCV*, pages 127–141, 2014. 2, 3