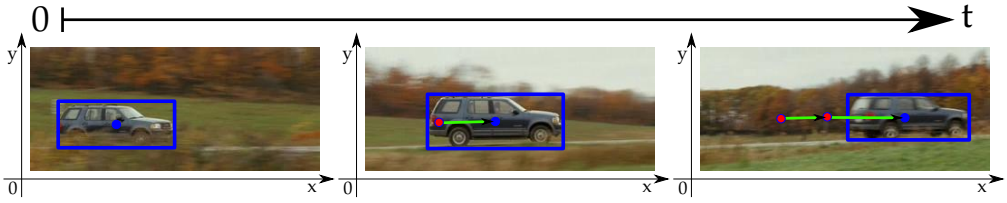
Excel@FIT 2015
http://excel.fit.vutbr.cz

Real-Time Long-Term Visual Object Tracking

Martin Simon*



Abstract
Visual object tracking with focus on occlusion, background clutter, image noise and unsteady camera movements, those all in a long-term domain, remain unsolved despite the popularity it experiences in recent years. This paper summarizes a related work which has been done in trackers field and proposes an object tracking system focused on solving mentioned problems, especially the occlusion, rough camera movements and the long-term task. Therefore, a system combined from three parts is proposed here; the tracker, which is the core part, the detector, to re-initialize tracker after a failure or an occlusion, and a system of adaptive learning to handle long-term task. The tracker uses newly proposed approach of bidirectional tracking of points, which are generally weaker then commonly used keypoints. Outputs of both the tracker and the detector are fused together and the result is also used for the learning part. The proposed solution can handle mentioned problems well and in some areas is even better then the state-of-the-art solutions.

Keywords: object tracking — bidirectional tracking — partial occlusion — long-term tracking — full occlusion — rough camera movement

Supplementary Material: Demonstration Video

*xsimon14@stud.fit.vutbr.cz, *Faculty of Information Technology, Brno University of Technology*

1. Introduction

Despite the fact that a visual object tracking in video has become a popular topic in recent days, it still remains generally unsolved. There is a significant amount of very efficient object tracking systems which are sufficiently accurate and which work in real-time. Unfortunately, many problems, such as occlusion, no matter if full or partial, background clutter, noise or unexpected rough camera movements, are problematic parts of many recent visual object tracking related projects and actually exclude them from daily professional usage.

In this work I would like to focus on recent solutions in the field of tracking and try to build an object tracking system whose efficiency and performance are at least the same as the performance of the state-of-the-art trackers. Also, I would like to improve some critical parts which have serious impacts on accuracy or are too narrowly focused and lack generality.

Further aim of this work is to bring a new challenging evaluation dataset with some highlighted problems, which the standard trackers handle incorrectly. These problems include primarily very small objects of interest or very similar objects in comparison with the background. The new tracking system should be designed in the way to handle both the standard visual object tracking evaluation dataset and the new challenging dataset, ideally without any significant speed or accuracy reduction.

This paper is organized as follows. In section 2 related work is summarized. The section is further di-

vided into a part describing current situation in objects representations (Subsection 2.1), tracking approaches are described in subsection 2.2 and the last subsection provides a short overview of complex tracking solutions presented in recent days.

The core contribution of the paper is in section 3, where the main parts of the proposed solution are described. The experiment is described in section 4, which contains the evaluation as well. In the last section (5) the results are interpreted and a conclusion is given.

2. Related Work

A significant amount of work has been written recently regarding the field of the object tracking in a video. In this section, related parts of tracking based systems are described.

2.1 Object Description

An object representation sets the apriori abilities of every tracking system. The object representation is a way how to treat the object model internally in the system. The system abilities are highly correlated with the type of the object representation.

Possibly the most straightforward way how to represent an object is to use a visual image template [1, 2]. The weakness of this representation can be an inability to cover object appearance changes, which may occur in case of tracking non-rigid objects or during tracking for longer periods.

Another option to represent an object could be a description with a set of good-to-track keypoints. The term keypoint is commonly understood as a significant point like corner or peak with its neighborhood. This method is generally much better than templates, but the quality of the keypoints is critical. There are various widely used descriptors, such as [3, 4, 5, 6].

There are also some other object description methods, like contours [7], which is basically an object geometry, or complex object representations [8], which combine other models together.

2.2 Localization Methods

Localization can be presented as the core of every object tracking system. The used approaches can be divided into two types; frame-to-frame (recursive) tracking and in-frame detection. The main difference is in usage of history of object position.

In the frame-to-frame tracking the history is used. It brings better handling of the missing information, but the processing error can be cumulated over time. On the other side, the in-frame detection does not use the history and searches for the object in every frame separately. The error cannot be cumulated, but it cannot handle the missing information neither. There is also less information in a single frame then in a sequence of multiple frames.

The object detection is performed with a method called *sliding window*, and it is widely used in many trackers and detectors [9, 10, 11]. This is generally a very slow approach, but the performance can be improved by developing some kind of *pyramid* to discard as many true negatives in early stages as possible [10, 12, 13, 14].

The recursive tracking is an approach much related to the object tracking. The reason is clear - the tracking is performed on top of a sequence of frames and it comes with higher density of information than single image. If it is possible, it is a good approach to use this kind of added information.

One way how to use this information is to model object with its movement, very often in form of position and velocity. Then, the future object position can be estimated and therefore the amount of possible object positions is significantly reduced. This is used in systems based on *particle filters* [15, 16] or *optical flow* [1, 17], among others.

2.3 Complex Solutions

The tracking system parts described in previous section are building blocks of complex tracking systems. Such a complex long-term tracking system requires a tracker for the basic tracking function, a detector for handling situations when the object gets lost or the tracker has failed, and some kind of learning system to handle object appearance changes. This does not necessarily require a presence of an object model [18], but the adaptation needs to be present in some way.

A huge number of visual object trackers exists. One often highlighted system is TLD [12, 19]. It is a tracking system with sufficient tracking performance, subtitled *tracking-learning-detection*. A tracking system based on strong keypoints and their voting can be CMT [20], which is considered a good representative of key-point based trackers. Another tracker which is good to mention is Struck [21], which is a tracker solution with adaptive visual templates and online SVM classifier. All of these trackers are considered state-of-the-art.

3. Proposed Solution

The long-term tracking system proposed here is based on OpenTLD [10], an improved C++ implementation of TLD [12]. The main reason is that the TLD consists

of two main parts, the tracker and the detector, which is improved with adaptive learning system. In OpenTLD, there is more improvements, like breaking away from the necessity of strong keypoints.

Therefore, the proposed solution is the OpenTLD with a completely new tracker and a new system of fusion of outputs of both the tracker and the detector.

The particular tracking problem to solve is defined in 3.1. It directly leads into the object representation, which is described in 3.2. The core contribution of this paper, the bidirectional tracking proposal, is described in 3.3. Finally, the results of the detector and the tracker are fused in accordance with 3.4.

3.1 Problem Definition

The problem of visual object tracking in video sequence can be described as follows. The input of the system is a sequence of monochromatic video frames. The output can be generally a four dimensional vector $(x, y, w, h)_t$ symbolizing the object position and the size in every frame. The result can be also a zero vector, if the object is not present in a particular frame.

The algorithm is commonly initialized by the user, after marking the object with bounding rectangle. The object is generally unknown to the system and this user input is the very first information about the object.

So far the tracking is not different from the detection of an object. The difference comes with the usage of history of results to improve result in following frames. Unfortunately, an error introduced in early states leads to bigger error at the end, or eventually, a failure during the process.

3.2 Object Representation

The tracking object can be generally *anything* which exists in the real world. Therefore, every good object tracking system needs to find a proper object representation with respect to other system parts.

The most promising seems to be keypoints model described with usage of keypoints like [3, 4, 5, 6]. The advantages are that these keypoints are easily and quickly found and even if some of them are lost (e.g. due the *partial occlusion*), the rest can still represent the object well. The disadvantage comes with a situation when we are not able to find such strong keypoints on the object; that can happen e.g. due to a limited object size or poor object resolution or contrast.

In the proposed solution the object is internally treated in two different processes. According to that, there are also two different object representations. After the user sets the bounding box of the object, the image *pattern* is stored also as a very first true positive sample for the detector. The pattern is resized and normalized and stored in internal memory besides the true negative, which are taken randomly from the area in the same frame. The frames are marked as true negatives in this initial learning phase if they are disjunctive to the true positive one.

Much more interesting is the object representation used during tracking phase. We propose to use 16 points in their 11×11 neighborhood to describe the object, and to choose these points according to their quality to track [2]. As it was mentioned before, there is a very limited set of such points in our dataset (small object, poor resolution/contrast). Therefore, we select those 16 best points which are possible to find. It is highly expectable that the quality of the majority of these points will be very poor.

One of the problems of visual object tracking is called *background clutter*. It means that the background contains more information than the object itself and, hence, the best points will be chosen from the background rather than from the object, what is obviously a failure. To compensate this problem, a mask of the object is computed before the points are actually searched.

The mask represents generally a difference between the internal bounding box area and the bounding box neighborhood. The mask value is calculated for every single bounding box point. As it was outlined, the value is a sum of weighted differences of the nearest out-box regions. Each region size is set to 5×5 pixels. This is illustrated in figure 1.

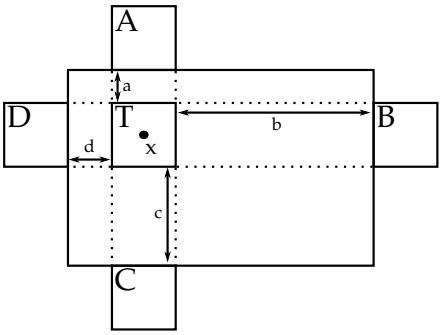


Figure 1. Mask computation illustration. The resulting value is given by a sum of weighted inverted NCC between the point neighborhood and all four nearest out-box regions

The figure 1 is described by equation 1. The symbols are the same as in the figure 1. Symbols A, B, C and D stand for the nearest out-box regions, the T is a neighborhood of the point x, which is the point of the mask. The function $NCC(M,N)$ stands for the *normalized cross correlation* and it is used to compute the similarity/difference between the templates M and N. The symbols a, b, c and d are distances of the

computed region T and corresponding out-box regions. These distances are used to compute weights. The symbols w and h represents the bounding box width, resp. height.

$$\begin{aligned} p \quad = \quad & NCC(T,A) * (1-(a/h)^3) + \\ & NCC(T,B) * (1-(b/w)^3) + \\ & NCC(T,C) * (1-(c/h)^3) + \\ & NCC(T,D) * (1-(d/w)^3) \end{aligned} \tag{1}$$

The mask can be seen in illustration 2. On the left there is the original bounding box used to compute the mask. On the right there is the mask. The white color means the most different regions and therefore *the object*.

Figure 2. Original *bounding box* and its mask. The white color represents the most different areas

This mask is used to select points to track from the object regions rather than from the background. The points placed on the background can lead to serious tracking error.

3.3 Multiple Bidirectional Tracking

Regarding the expected poor quality of points used to track, it has to be improved in tracking process itself. In [19] an approach called *Forward-Backward Error* (FBE) is presented. This basically means that every point is tracked in forward direction, the result is tracked back and the distance between the original point and the returned one is measured and called the *FBE*. Points with too high FBE are then excluded from the tracking process.

This idea brings very good results as it was proved in [19]. Unfortunately, this only excludes wrongly tracked points from the process, but does not improve their individual quality.

In this paper is proposed an improved solution based on multiple tracking in both directions. Every point selected to tracking is tracked in forward direction, which results in 2D histogram (a map) of expected point positions in the next frame, which are measured as the *NCC* between the original point area and the tracking point area.

Instead of taking only the most valuable point to track back, like it is in FBE, we track back 3 most expected positions. As it was observed from experiments, the correct tracking position is not always the most valuable position in the map, but sometimes also the second one or even the third one.

After tracking them back we take into account 2 best candidates of every backward tracking, so for one tracking point we get 3 image candidates and 6 possible backward tracking images. For each of these backtracked images a distance to the original position is measured and the closest point is noted. The starting point for the noted one is then marked as a result.

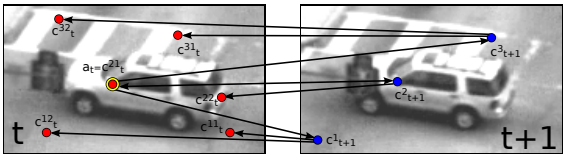


Figure 3. Multiple Bidirectional Tracking (MBT). Taking into account more candidates and tracking them back brings higher chance to find successful tracking path

The situation is better described with an image. In figure 3 can be seen point a_t on the left part (frame in time t) and three of its possible trackings to the frame in time $t+1$. Also, two possible trackings back for every candidate to the original frame can be seen, with some failing results and a successful one.

In the end, the successfully tracked points are those, which end in the same point they came from, with α as a spatial tolerance. Other points are excluded as wrongly tracked, as is in the FBE.

3.4 Detector-Tracker Fusion

The detector is an integral part of a long-term tracking system. It is the only way how to re-initialize tracking process after full occlusion or tracker failure.

The detector used in the proposed solution is taken from [10] unchanged. The detector is very fast, because it is developed with respect to cascade (or pyramidal) principles. The detector cascade consists of 3 stages. A *Variance filter*, which is the first stage with straightforward task to reject as many false positives as possible. The measurement metric is a simple variance. The next stage is an *Ensemble classifier*. This classifier uses method called *random fern classification* [22, 23]. The last stage and the slowest one is a *template matching* with usage of *NCC* as the similarity measurement technique.

With usage of both the detector and the tracker, following rules can be set. If only the detector or the tracker has result, its result is used. When neither the tracker nor the detector has result, then the object is considered as invisible.

In case both the tracker and the detector have their results, then the detector's output is used as an addition to the tracker's one in a fusion originally presented in this paper. All the combinations are for better overview in figure 4.

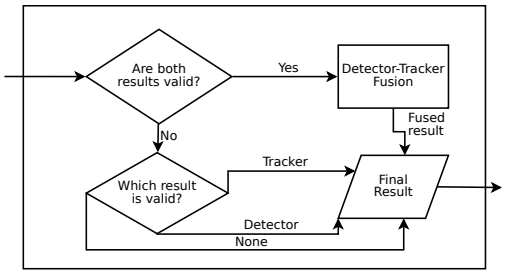


Figure 4. If only one (the detector or the tracker) is successful, its output is used. Otherwise, their outputs are fused

The tracker's and the detector's output is a position and a size of the object. Then, if the results overlap with at least 40%, they can be fused. The fusion result is computed as weighted mean and the weight is considered to be a certainty of the tracker and the detector.

The detector's certainty is computed in its last stage as in equation 2. Then $n \in negativesamples$, and $p \in positivesamples$.

$$\begin{aligned} d_N &= 1 - max(NCC(T, n_i)) \\ d_P &= 1 - max(NCC(T, p_i)) \\ a &= d_N/(d_P + d_N) \end{aligned} \tag{2}$$

The certainty of the tracker is set in accordance with the detector's one. It can be easily done with a small hack; the tracker's result put as an input for the last stage of the detector to get the very same certainty, only for the tracker this time.

The fused result position is illustrated in equations 3 and 4.

$$x = \frac{a_t * x_t + a_d * x_d}{a_t + a_d} \tag{3}$$

$$y = \frac{a_t * y_t + a_d * y_d}{a_t + a_d} \tag{4}$$

In the equations 3 and 4 the x and y are the fused coordinates of the object center, the a_t and a_d are the certainties of the tracker and the detector. The x_d, y_d, x_t and y_t are original outputs from the detector and from the tracker.

This kind of fusion should bring better movement correction in case, e.g. the tracker drifts. On the other hand, an additional error can be entered from the detector, in case of false positives.

4. Experiment and Evaluation

This section consists of two main parts. In the first part (Subsection 4.1), the experiment and evaluation prerequisites are set, including the evaluation dataset. The next part (Subsection 4.2) summarizes results of the evaluation according to the experiment presented in the first part.

4.1 Experiment Description

The dataset used for evaluation of the proposed tracking system consists of several standard video sequences and a few new ones. The standard dataset is represented with sequences known as *ball*, *bicycle*, *car* and *jogging* and it is taken from *VOT2014* [24]. The subset covers problems like object rotation (the ball), background clutter (the bicycle), object scale (the car) and occlusion (jogging).

The originally obtained sequences have work titles *helicopter* and *plane51*. This dataset focuses on small objects of interest (both), low contrast and poor quality resolution (the helicopter), object appearance changes (the plane51) and rough camera movements (both). The plane51 consists from 923 frames and the helicopter from 440 frames. The groundtruth was annotated manually by marking objects' centers.

In figure 5 can be seen representatives from all the datasets. In the top line can be seen the representative images from the standard dataset, in the bottom line are representatives from the original one.

For every frame sequence the proposed tracking solution is evaluated several times to get statistically relevant result. The only result which is measured is a position of the object center on a particular frame. The success rate for every frame is measured according to equation 5.

$$rate = 1 - \frac{\sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2} - \sigma}{\gamma - \sigma} \tag{5}$$

The equation 5 says that the success rate for the particular frame is equal to complement to 1 of normalized Euclidean distance between the correct center position and the position the tracker gives. The equation also contains boundaries due the manual groundtruth marking process. It says that the distances shorter than σ (30 pixels) are considered as 100% correct, and longer that γ (70 pixels) are considered as 100% incorrect.

Other long-term tracking solutions have been already mentioned in section 2.3. Only to summarize it, the compared trackers are OpenTLD [10] and CMT [20]. The reason is mainly that they are considered the state-of-the-art. Another reason is that these trackers have their reference implementation publicly available.

Figure 5. Dataset samples. From the left to the right they are: *ball*, *bicycle*, *jogging*, *helicopter*, *car* and *plane51*

4.2 Results

The proposed solution have been tested in quality domain as was mentioned in the previous subsection. In table 1 are summarized evaluation results for system without the mask computation nor the *MBT*, for both improvements separately and a complete system. Every results is combined from a mean of overall success rate of all sequence frames together in several runs and a standard deviation which symbolizes how unstable the system is (how different the single overall results were).

Table 1. Improvements results. Overall success rate for particular improvement usage and standard deviation. All values are percentages

	None	Mask	MBT	Complete
Ball	66 / 16	66 / 10	73 / 09	71 / 06
Bicycle	70 / 05	72 / 10	71 / 05	79 / 07
Car	73 / 09	79 / 11	78 / 10	79 / 14
Jogging	81 / 10	78 / 13	90 / 03	84 / 13
Helicopter	75 / 07	09 / 01	74 / 04	70 / 32
Plane51	87 / 02	85 / 01	87 / 02	85 / 02

The table shows how the proposed improvements mostly bring an increase of performance in comparison with a system without these improvements. The complete system is almost always better than the basic one.

Table 2 sums success rates for every evaluation dataset and the mentioned trackers to compare, and also the results of system with all proposed improvements. The overall success rate is again a mean from several runs. The most significant numbers (both the best and the worst) are highlighted.

Table 2. Evaluation results. Overall success rate and standard deviation. All values are percentages

	Our	OpenTLD	CMT
Ball	**71** / 35	**99** / 11	98 / 11
Bicycle	**79** / 38	**01** / 09	66 / 46
Car	**79** / 36	64 / 48	**62** / 48
Jogging	**84** / 29	**25** / 43	82 / 37
Helicopter	**70** / 43	**04** / 20	46 / 21
Plane51	**85** / 33	08 / 27	**04** / 17

According to the results, the proposed solution performs better in our dataset. That was expected, as the dataset contains some problematic parts that the other solutions do not solve well. This was also the main reason for the proposed solution.

In the standard dataset the quality performance of the proposed solution is comparable to other solutions. This is considered as a very good result; maybe even better than overperforming the original dataset. It also means that after the generalization to solve our dataset the common performance was not disrupted.

Table 3. Speed evaluation. The M means that MBT is applied, otherwise it is not. All values are means across all datasets.

	64	64/M	256	256/M/C	OpenTLD	CMT
fps	23.2	14.8	2.4	8.8	107.2	45.5

The computational speed is highly affected by an allowed maximal object movement and a number of backtracked points (usage of MBT). In table 3 are shown average values for few configuration with added average values of other trackers, to avoid dependency on machine performance. The values stand for frames-per-second (fps) and the proposed solutions is evaluated with $64px$ and $256px$ as a maximal object movement in all axis directions and by usage of MBT. It is good to mention that the movement by $64px$ is generally much bigger than is needed and much bigger than other solutions offer. The *C* symbolizes a partial CUDA acceleration.

5. Conclusions

In this paper a long-term tracker was introduced. The proposed solution is focused on some particular problems, like small object size and poor image contrast quality. Beside these specific problems, the tracker performance on standard dataset should not be decreased and the tracker should not lose its generality.

As the evaluation showed, the proposed solution overperformed state-of-the-art trackers in specific dataset. On the standard dataset, our tracker performed comparably to others.

The main contribution is the tracking quality measurement called *multiple bidirectional tracking*. The main idea of the method is to move tracking point quality measurement into the tracking process itself.

The system was built on *OpenTLD* background

and therefore it took its structure; the tracker and the detector cooperation. The cooperation was tied closely by newly proposed *Detector-Tracker Fusion*.

The proposed solution is a general object tracker prototype generalized to solve the new problems and able to run in real-time, although it is slower then other solutions. Hence, it can be improved in any conceivable domain. Some of the first steps should be more sophisticated online learning system.

References

[1] Bruce D. Lucas and Takeo Kanade. An iterative image registration technique with an application to stereo vision. In *IJCAI*, volume 2, pages 674–679, San Francisco, CA, USA, 1981. Morgan Kaufmann Publishers Inc.

[2] Jianbo Shi and Carlo Tomasi. Good features to track. In *CVPR*, pages 593–600, 1994.

[3] Herbert Bay, Andreas Ess, Tinne Tuytelaars, and Luc Van Gool. Speeded-up robust features (SURF). *Computer Vision and Image Understanding*, 110(3):346–359, 2008.

[4] Alexandre Alahi, Raphael Ortiz, and Pierre Vandergheynst. FREAK: Fast retina keypoint. In *CVPR*, pages 510–517. IEEE, 2012.

[5] Ethan Rublee, Vincent Rabaud, Kurt Konolige, and Gary Bradski. ORB: An efficient alternative to SIFT or SURF. In *ICCV*, pages 2564–2571. IEEE, 2011.

[6] Stefan Leutenegger, Margarita Chli, and Roland Y. Siegwart. BRISK: Binary robust invariant scalable keypoints. In *ICCV*, pages 2548–2555. IEEE, 2011.

[7] Charles Bibby and Ian Reid. Real-time tracking of multiple occluding objects using level sets. In *CVPR*, pages 1307–1314. IEEE, 2010.

[8] Alper Yilmaz, Xin Li, and Mubarak Shah. Contour-based object tracking with occlusion handling in video acquired using mobile cameras. *TPAMI*, 26(11):1531–1536, 2004.

[9] Christoph H. Lampert, Matthew B. Blaschko, and Thomas Hofmann. Beyond sliding windows: Object localization by efficient subwindow search. In *CVPR*, pages 1–8. IEEE, 2008.

[10] Georg Nebehay. *Robust object tracking based on tracking-learning-detection*. Wien, 2012.

[11] David Forsyth and Jean Ponce. *a modern approach*. Pearson, Boston, 2 edition, 2012.

[12] Zdenek Kalal, Krystian Mikolajczyk, and Jiri Matas. Tracking-learning-detection. *TPAMI*, 34(7):1409–1422, 2012.

[13] Anna Bosch, Andrew Zisserman, and Xavier Munoz. Representing shape with a spatial pyramid kernel. In *CIVR*, pages 401–408. ACM, 2007.

[14] Ondrej Chum and Andrew Zisserman. An exemplar model for learning object classes. In *CVPR*, pages 1–8. IEEE, 2007.

[15] Paul A. Brasnett, Lyudmila Mihaylova, Nishan Canagarajah, and David Bull. Particle filtering with multiple cues for object tracking in video sequences. In *Electronic Imaging*, pages 430–441. SPIE, 2005.

[16] Xin Sun, Hongxun Yao, and Shengping Zhang. Contour tracking via on-line discriminative appearance modeling based level sets. In *ICIP*, pages 2317–2320. IEEE, 2011.

[17] Jean-Yves Bouguet. Pyramidal implementation of the affine lucas kanade feature tracker description of the algorithm. *Intel Corporation*, 5, 2001.

[18] Lu Zhang and LJP van der Maaten. Preserving structure in model-free tracking. *TPAMI*, 36(4):756–769, 2014.

[19] Zdenek Kalal, Krystian Mikolajczyk, and Jiri Matas. Forward-backward error: Automatic detection of tracking failures. In *ICPR*, pages 2756–2759, 2010.

[20] Georg Nebehay and Roman Pflugfelder. Consensus-based matching and tracking of keypoints for object tracking. In *Winter Conference on Applications of Computer Vision*. IEEE, IEEE, 2014.

[21] Sam Hare, Amir Saffari, and Philip H. S. Torr. Struck: Structured output tracking with kernels. In *ICCV*, pages 263–270. IEEE, IEEE, 2011.

[22] Mustafa Ozuysal, Pascal Fua, and Vincent Lepetit. Fast keypoint recognition in ten lines of code. In *CVPR*, pages 1–8. IEEE, 2007.

[23] Mustafa Ozuysal, Michael Calonder, Vincent Lepetit, and Pascal Fua. Fast keypoint recognition using random ferns. *TPAMI*, 32(3):448–461, 2010.

[24] VOT2014. [online]. [cit. 2015-04-14].