2014 22nd International Conference on Pattern Recognition

Improving object tracking with voting from false positive detections

Vassileios Balntas
University of Surrey
Guildford, UK
v.balntas@surrey.ac.uk

Lilian Tang
University of Surrey
Guildford, UK
h.tang@surrey.ac.uk

Krystian Mikolajczyk
University of Surrey
Guildford, UK
k.mikolajczyk@surrey.ac.uk

***Abstract*—Context provides additional information in detection and tracking and several works proposed online trained trackers that make use of the context. However, the context is usually considered during tracking as items with motion patterns significantly correlated with the target. We propose a new approach that exploits context in tracking-by-detection and makes use of persistent false positive detections. True detection as well as repeated false positives act as pointers to the location of the target. This is implemented with a generalised Hough voting and incorporated into a state-of-the art online learning framework. The proposed method presents good performance in both speed and accuracy and it improves the current state of the art results in a challenging benchmark.**

I. Introduction

Context is very important in object tracking and many recent approaches make explicite use of the context to improve the tracking results [1], [2], [4], [5]. Exploiting the spatial arrangements of the context in a scene as well as correlated motion of neighbouring objects can help localise the position of the tracked object with greater accuracy. Furthermore, it can act as a predictor when the object is absent from the scene as in [1]. Recent work in [6] emphasises the role of the context in a scanning window based detection, and demonstrates that the position of the target can be predicted faster and more accurately by using the context information.

Our work builds upon the methods that add context to the tracking process but also on the idea of Implicit Shape Model (ISM) [17], [18]. ISM is a very successful offline trained object detector based on interest point descriptors, codebook model, and generalised Hough voting. Generic interest point detectors and descriptors [3] as well as matching limits the efficiency of this approach and makes it difficult to adapt to online learning problems. However, an efficient and discriminatively trained detector can provide a few but more reliable candidates for object locations than a generic interest point detector. Moreover, explicit modelling of various context objects [1], [4], [5] adds to the complexity of the online learning and detection. Our goal is to avoid explicit modelling of the context regions, and to exploit persistent detections resulting from an online learnt target model as predictors of the target position regardless their motion pattern. Furthermore, the idea is to use true positives as well as false positive detections in the Hough voting framework to improve the detection of the object of interest. This use of false positives is an alternative way of improving the classifier compared to directly using false positives as negative appearance examples during online training. Some false positives are extremely similar to the target and may significantly contribute to overfitting the classifier that may become too discriminative. In other cases an efficient classifier may not be sufficiently discriminative and generates false positives despite good training examples. The proposed approach is addressing both problems in online trained tracking-by-detection system. One of the very well performing online learnt detectors/trackers in the recent evaluation [7] is tracking-learning-detection (TLD) from [9]. We adopt this learning approach in our method and implement the ideas mentioned above within TLD framework.

In contrast to the other tracking methods that exploit context, we consider all detections (true and false), provided by a classifier trained for a single object, as pointers to the true location of the target. In our approach a discriminative detector trained on the target examples acts as an object of interest extractor which gives less regions to process than in case of interest points but more likely to correspond to the target. The main contributions of the proposed method are:

- We propose a new approach that exploits locations of all candidate detections including false positives to indicate the location of the true object, thus all detections are used as predictors of the target position.
- We propose an efficient implementation based on the generalised Hough transform which can be used to extend any other existing online detector.
- We incorporate the proposed method into the tracking-learning-detection framework [9] and show that it significantly improves the results on a number of challenging sequences.

A. Related work

One of the first approaches that discussed the role of context in improving robustness of a tracker was the work by Cerman et. al. [2]. The basic idea is to automatically identify any part of the background (i.e. regions that are not parts of the target) that moves in a consistent way with the object and lies at the boundary space between the target object and the background. In other words, the object is constantly expanded until it converges to a larger region that moves in a consistent way. For example, if the goal is to track a head, the model may extend to the region around the shoulders of the person as it is often moving in a correlated way with the target.

1051-4651/14 $31.00 © 2014 IEEE
DOI 10.1109/ICPR.2014.337

Grabner et al. [1] extend this idea by devising a framework based on keypoints rather than identifying neighbouring areas by correlated movements. They introduce the so called *supporters* which can be defined as keypoints that move in a consistent manner with the target object, and they can be located on the object itself, or arbitrarily far away from the object. With a simple model that uses the generalised Hough transform to localise the object from individual votes of the supporters, they are able to identify the global maxima of the hough voting space and infer the position of the object, even in the extreme cases when it is fully occluded or it moves outside the scene boundaries.

Ba Dinh et al. [5] propose a different method of exploiting context during object tracking by making use of two semantic object categories in a scene. One of the categories are the supporters, similarly to [1] which are defined as keypoints around the object that move in a similar way. However, they also point out that there exists a specific class of objects during on-line detection and tracking, which have similar appearance to the target object and are called distracters. In any tracking by detection system, there is a very high probability that drifting may occur towards the distracters, because of their significant similarity to the target. To address this problem, the authors track the distracters simultaneously with the target, thus are able to prevent drifting to one of the distracters.

Another approach based on motion consistency is the work presented by Stalder et al. [10], where a fast motion segmentation method is used in order to group similar moving keypoints into clusters that represent objects based on the notion of dynamic objectness. Those clusters are later classified as either belonging to the object or to the background. By growing the keypoint based model for both the object and the background, they are able to identify new clusters and adapt to new appearances of the object on-line.

Sun et al. [11] use GLAD method [12] for integrating labels from annotators with unknown expertise in order to use the context from neighbouring objects and specific parts of the target to simultaneously track both the object as well as the context. The spatial configuration of the context models is then used to infer the final position of the target. However, those supporting objects (helpers) are not discovered automatically, and have to be manually annotated at the beginning of the process.

Finally, Yang et al. [13] propose a method to automatically discover auxiliary objects during tracking, which co-occur and present consistent motion correlation with the target. They later simultaneously track the auxiliary objects in order to verify the target position more efficiently.

B. Overview

The proposed method is inspired by the recent advances in the context based detectors and trackers outlined in several works above. We make use of an efficient classifier which exhibits very high recall but low precision. We use fern classifier similar to the one proposed in [14] that was successfully used in an online trained detector in TLD [9]. The fern classifier usually returns a large number of candidates of the object location that have to be validated by a less efficient but more discriminative classifier in TLD. Some of the candidate detections persistently return in subsequent frames despite adding these examples to the negative training set during online learning. Labelling these hard negatives is challenging due to their high similarity to the target in the object representation space. In [5] such objects are called distracters and are modelled and tracked together with the target object, which eliminates them from the list of candidates and prevents drifting. However, in contrast to their similar appearance their location is different from the true object. We propose to incorporate the location of these candidates with respect to the target location into the detection process. Moreover, in our work, we do not differentiate between the target and the distracters, and thus we consider all the candidates that are returned by the classifier as pointers. An illustration of the proposed method, together with the two closely related approaches from [1] and [5], are presented in Figure 1. We propose to infer the position of the target based on the configuration of the pointers positions and we incorporate this idea into the TLD approach. Each of the pointers casts a vote for the target position learnt from previous frames, and the accumulated votes in the voting space indicate the position of the target in the current frame.

II. Proposed approach

In this section we present our proposed approach that includes the initialisation, detection and update of the model.

A. Model initialisation

The first step of the process is to initialise the object location in the first frame and to train the classifier. In principle, any classifier that is efficient, and can provide many detection candidates is suitable. The objective is to use a classifier with high recall, which may come at the expense of low precision. We use the fern based classifier from TLD. This classifier evaluates a large number of sliding windows from every single frame, and returns a set of candidates to the subsequent detection process. However, further validation of the returned candidates is crucial for the tracker's accuracy. TLD uses a nearest neighbour classifier based on normalised cross-correlation of patches to identify true positives and discard the hard negatives. Instead, we propose to associates a pointing function to each of the candidates. The role of this function is to estimate a displacement vector that points to the position of the target. Note that the position of the target is known in the first frame only and in the subsequent ones it is estimated from the pointers.

Let x_o, y_o denote the centre of the bounding box that represents the target in the first frame. We train and evaluate the fern detector in the first frame to get a list of detection candidates. We also lower the classification threshold, in order to obtain a list of detections that include the target's bounding boxes as well as a set of false positive (non-target related) bounding boxes. Each candidate is represented by a tuple $P = \{\boldsymbol{\beta}, x, y, \Delta x, \Delta y, f\}$ where $\boldsymbol{\beta}$ is a descriptor extracted from the detected bounding box, x and y are the coordinates of its centre, Δx and Δy represent the distance in both dimensions between the detection and the target $\Delta x = x_o - x$, $\Delta y = y_0 - y$ and f is the frame number in which this specific bounding box was seen. Any descriptor can be used, but in our implementation we use BRIEF [16] due its efficiency and

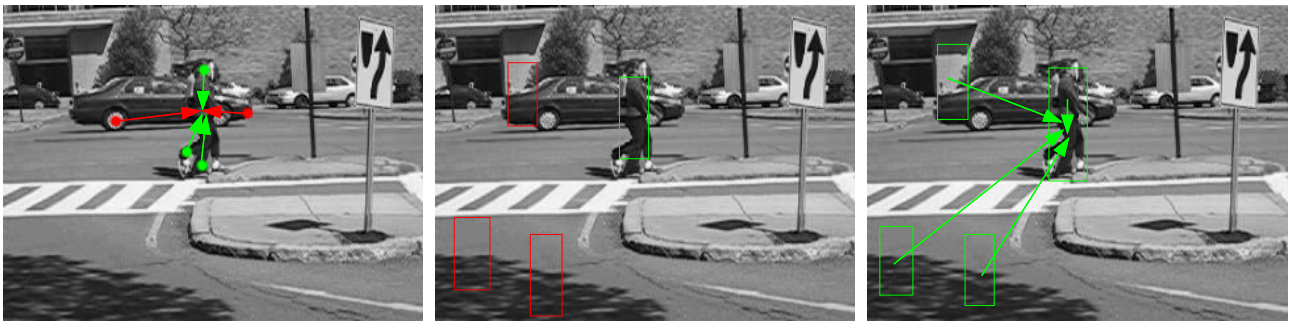

Fig. 1: Grabner et al. [1] (left) use a set of keypoints that move in a significantly correlated way with the target as supporters in the tracking process. Note that the supporters can be either on the target object itself (green points) or on other objects (red points). Ba Dinh et al. [5] (middle) track objects with similar appearance (red boxes) to the target (green box) to prevent drifting during tracking. Our approach (right), uses all the detections from a classifier to point at the position of the target. Note that those detections can be from the target itself as well as from arbitrary scene objects with similar visual characteristics.

matching accuracy that was recently demonstrated in [3]. Note that we do not label the candidates, but instead we store the $\Delta x, \Delta y$ values that point to the target. The pointers with low values of $\Delta x, \Delta y$ are more likely to represent the object than pointers with large values.

B. Detection process

Once the model is initialised in the first frame, there exist a set of pointers in the database that can be used for the detection in the following frames. In the next frame, the fern classifier provides a set of candidates. BRIEF descriptor is extracted for each candidate $P_* = \{\boldsymbol{\beta}_*, x_*, y_*, \Delta x_*, \Delta y_*, f_*\}$ and matched against the pointers from the database to find its nearest neighbour. The list of tuples is processed sequentially by first considering the neighbours in the spatial coordinates and then by comparing the BRIEF descriptors:

$$\mathcal{RDB} = \{p, ||(x_*, y_*) - (x_p, y_p)||_E < T_c, \ \forall p \in \mathcal{DB}\} \quad (1)$$
$$P_{NN} = \underset{p \in \mathcal{RDB}}{\arg\min} \left(||\boldsymbol{\beta}_p - \boldsymbol{\beta}_*||_{\mathcal{H}}\right)$$

where T_c is a spatial distance threshold, $||\cdot||_E$ is the Euclidean distance between box centres and $||\cdot||_{\mathcal{H}}$ represents Hamming distance between BRIEF descriptors. The use of two stage nearest neighbour search significantly accelerates the processing and enforces the temporal consistency of the pointers. The parameters of the nearest neighbour match P_{NN} are used to calculate the coordinates of the bin in a discretised voting space $\mathcal{V}$ which is incremented:

$$\mathcal{V}(x_* + \Delta x_{NN}, y_* + \Delta y_{NN}) := \quad (2)$$
$$\mathcal{V}(x_* + \Delta x_{NN}, y_* + \Delta y_{NN}) + \frac{\tau}{\theta(f_* - f_{NN})}$$

where τ is a scalar, and $\theta(\cdot)$ is a monotonically decreasing weighting function. $\theta(\cdot)$ gives lower weights to votes that come from pointers that were observed earlier in the sequence. Intuitively, a pointer that was observed a long time before the current frame is less important than the ones observed in the previous frame. A more sophisticated approach that employs Gaussian based weighting similar to the one used in [1] can be adopted. Finally, once all the candidates cast their votes, the maximum in the voting space $\mathcal{V}$ is detected. We consider valid target detections as the ones that overlap with this maximum, and we estimate the target location as the average of the valid bounding boxes. This formulation also allows the case where the target is not found in the frame. In this case there are typically no bounding boxes overlapping with the global maximum of $\mathcal{V}$.

C. Updating the model

Once the detection described above has returned an estimate of the current position of the target, similarly to the initialisation process, we form a set of tuples $P = \{\boldsymbol{\beta}, x, y, \Delta x, \Delta y, f\}$, one for each of the observed pointers, and store them in the database. We keep the previously collected pointers for a fixed number of frames. If a pointer representing the same object appears in consecutive frames, the most recently collected one has the largest contribution to the voting process based on function $\theta(\cdot)$. Similarly to [9], in order to prevent drifting we do not update the model if the observed descriptor differs significantly from the previously seen target descriptors.

D. Implementation details

The database is used to accumulate the past history of the sequence but its size has a negative effect on the speed of the nearest neighbour search in the descriptor space, we therefore limit it to improve the efficiency. We limit the size a queue that holds the pointer tuples, and when the queue is full, we discard the oldest pointers from the model. Using this technique, once the queue is filled, our method performs at constant speed.

Another parameter that affects the speed is the number of pointers per frame. As one can expect increasing the number of pointers per frame increases the tracking performance, but reduces the speed of tracking. Thus in our implementation we set this number to be 100, and the maximum size of the pointer queue to 2000. This allows for the system to run with a speed of $4 - 11$ fps depending on the size of the original image. This speed is with Matlab implementation, which leaves much scope for improvements.

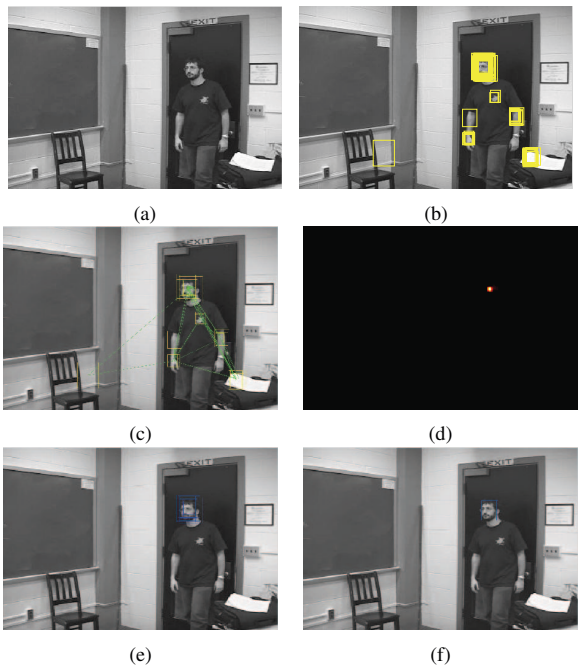

Fig. 2: Outline of the proposed method. (a) Input frame (b) Candidates provided by the classifier (c) Each candidate casts a vote indicating the target's position (d) The resulting voting space $\mathcal{V}$ (e) Valid detections that overlap with the global maximum of the voting space (f) Final result as the average of the valid detections.

A simplified outline of the method is presented in Algorithm 1 and an illustration of the detection steps is in Figure 2. As it can be seen from Figure 2 (c) there are some candidates that cast their votes in incorrect locations, but these are not detected in the voting space as the local maxima in such cases are very low.

III. Results

In this section we present qualitative and quantitative results to demonstrate the performance of the proposed method. We use the benchmark data from [7] which contains a large set of annotated video sequences. We compare to two other state-of the art methods and discuss the results.

A. Evolution of pointers

Figure 3 shows the evolution of the pointers over time on the test sequence `CarScale`. In the first frame of the sequence, where the model is not yet initialised, all the strong pointers are from detections that represent the actual object. In the next few frames, the bounding boxes start to act as pointers and are incorporated into the detection process. The green lines indicate the locations for which the bounding boxes vote. All pointers are visually similar to each other and they are selected based on nearest neighbour search by the binary BRIEF descriptors, therefore there are many mismatches that lead to incorrect voting locations. The boxes pointing to incorrect locations are represented by the red lines in Figure 3. However,

Algorithm 1 Pointer tracking

1: **if** first frame **then**
-Train the classifier from initial bounding box and random negatives.
-Collect *pointers*: All detections from the classifier.
-Initialize $\mathcal{DB}$: For each pointer, extract and store the representing tuple $P = \{\boldsymbol{\beta}, x, y, \Delta x, \Delta y, f\}$.
2: **end if**
3: **for** next frames **do**
- Clear voting space $\mathcal{V}$.
- Get candidates from the classifier.
4: **for all** candidates **do**
- Extract the tuple $\{\boldsymbol{\beta}_*, x_*, y_*, f_*\}$
- Update $\mathcal{V}$ using $(\Delta x, \Delta y)$ from the P_{NN} match.
5: **end for**
- Find global maximum m in $\mathcal{V}$
- Estimate target bounding box from candidates that overlap with m
- Update the target centre x_o, y_o
6: **for all** candidates **do**
- Evaluate new $\Delta x_* = x_o - x_*$ and $\Delta y_* = y_o - y_*$
- Form pointer tuple from candidate P_* and store in $\mathcal{DB}$.
7: **end for**
8: **end for**

since the number of pointers per frame is large, enough votes accumulate in the correct location and the maximum in the voting space can be estimated with good accuracy.

B. Recall-Precision results

To assess the performance of the proposed method, we use challenging videos from the recently released dataset used in a large scale comparison of online trackers [7]. In this comparison two approaches demonstrated particularly high performance, Struck [15] and TLD [9]. We focus on the sequences where either of these two failed to track through the entire sequence. The precision and recall scores are calculated based on the overlap criterion [7]. A detection is considered valid, if the overlap between the ground truth bounding box B_{gt} and the tracker result B_{res} is greater than 0.5, where the overlap o is defined as $o = B_{gt} \cap B_{res} / B_{gt} \cup B_{res}$.

The results presented in Table I show that the proposed system leads to a better performance than TLD approach [9] and Struck [15]. Our method outperforms TLD in 11 out of 13 sequences and Struck in 6 sequences in terms of recall, and in 5 sequences in terms of precision.

C. Discussion

The main strength of the proposed method is that all detections, both true and false positives, are used to localise the object. This makes the tracker more robust to drifting, and leads to higher average precision and recall than for the other two trackers. For example, in the `Crossing` sequence, both TLD and Struck drift due to significant illumination change on the boundary of the shadow as it is shown in the top row of Figure 4. It happens early in the sequence which significantly lowers the performance scores of both trackers. A comparison

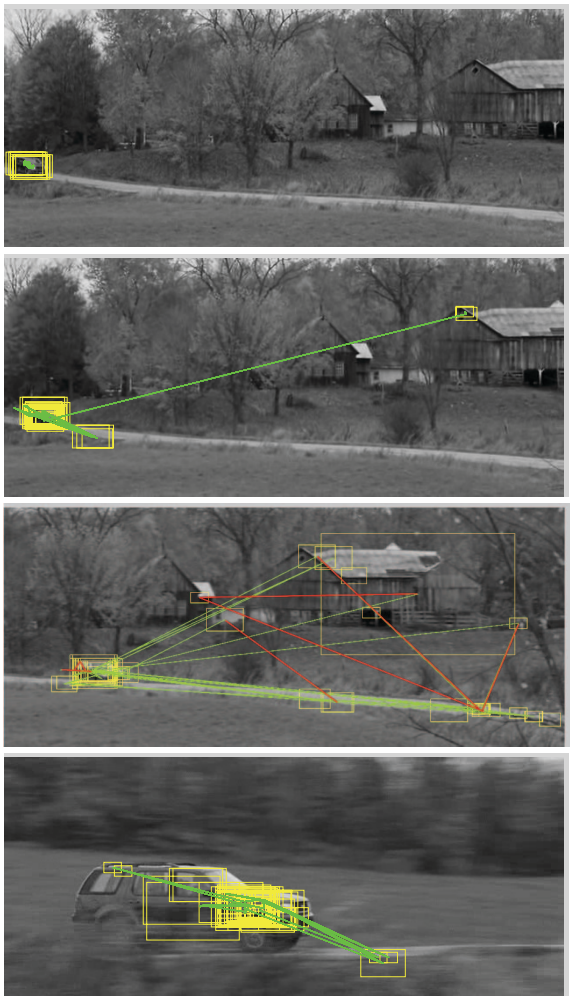

Fig. 3: The evolution of the pointers over time. In the initialisation of the model (top image), all the pointers originate from bounding boxes representing the object to be tracked. However, during the update process, we see that several new pointers appear and contribute to the detection process.

with Struck shows that while in general Struck gives excellent results in many videos, it does not adapt to different scales of the objects, and assumes that the object is always present in the scene thus not suitable for sequences where objects frequently disappears.

A comparison of our method to TLD [9] is essential, since both methods have similar detection processes. In both systems, a set of object candidates that are returned from the fern classifier are evaluated in order to either localise the object, or to identify a full occlusion/ absence of the object. The difference between the two methods is that our approach does not need to validate or assign a label to each of the candidates provided by the fern detector. Any error in such validation leads to either lower recall or a drift, due to the presence of negative candidates that are very similar to the object of interest. Instead, we consider every candidate as a potential supporter that can point to the true location of the object. Thus several pointers can aggregate around a specific position in the hough space. This can be observed in the `Freeman4` sequence with a face as the object of interest, presented in the second row of Figure 4. While Struck has drifted to a dissimilar object, TLD drifts to other faces, which are hard negatives in this scene and are validated as true positives by the nearest neighbour normalised cross-correlation classifier in TLD. However, since in our representation those false detections are recorded as pointers, together with the distance vector to the target, it is unlikely that a significant local maximum will be accumulated in the voting space around such candidate as it will not be supported by other pointers. This example illustrates the benefit of using the hard negatives as pointers, since their re-occurrence amongst the detection results makes it possible to use them for spatial voting rather than for improving the decision boundary of a discriminative classifier.

TABLE I: Precision-recall results

Sequence Name	TLD [9] Recall/ Precision	Struck [15] Recall/ Precision	Ours Recall/ Precision
Car4	0.50 / 0.60	**1.00 / 1.00**	0.97 / **1.00**
CarDark	0.72 / 0.72	**1.00 / 1.00**	0.69 / 0.69
CarScale	0.80 / **1.00**	0.79 / 0.79	**0.99** / 0.99
Couple	0.58 / **1.00**	0.88 / 0.88	**0.96** / 0.99
Crossing	0.70 / 0.70	0.39 / 0.39	**1.00 / 1.00**
David3	0.29 / 0.32	**1.00 / 1.00**	0.92 / 0.94
Deer	0.80 / **1.00**	**1.00 / 1.00**	0.75 / 0.91
FaceOcc2	0.45 / **1.00**	**1.00 / 1.00**	0.83 / 0.83
Fish	0.60 / 0.87	0.87 / **1.00**	**1.00 / 1.00**
Football	0.79 / **1.00**	**1.00** / 0.94	0.80 / 0.80
Freeman1	0.30 / 0.52	0.52 / 0.93	**0.95 / 0.96**
Freeman4	0.18 / 0.28	0.47 / 0.47	**0.71 / 0.74**
MountainBike	0.37 / 0.61	**1.00 / 1.00**	0.93 / 0.93
Average	0.55 / 0.74	0.87 / 0.87	**0.88 / 0.90**

IV. Conclusions

We have presented a novel idea of using true and false positive detections in tracking to localise the object of interest. Our method is based on a fast fern classifier, and a new representation of candidate detections which we call pointers. We demonstrated that false positive detections can be used differently than for refining a decision boundary of an appearance based classifier. Our approach is applicable to any tracking by detection system with online learning. It leads to improved results in particular when there are many similar objects in the scene or the classifier has a high false positive rate. This is a frequent problem in application scenarios where a high classification accuracy has to be sacrificed for high efficiency of the system. We incorporated this approach into TLD tracker which significantly improved its performance. Our evaluation results on 13 challenging sequences show improved precision and recall over TLD as well as Struck which are considered the state of the art tracking approaches. This is a significant achievement considering the simplicity of the proposed method.

Acknowledgement. This work was supported by EU Chist-Era EPSRC EP/K01904X/1.

Fig. 4: Comparison of the results from TLD [9] (green), Struck [15] (blue), and the proposed method (red). Unlike the other methods, in all cases our approach localises the object successfully, due to the support from the pointers. Top to bottom: `Crossing`, `Freeman4`, `Carscale`, and `Freeman1`.

REFERENCES

[1] H. Grabner, J. Matas, L. J. V. Gool, and P. C. Cattin, "Tracking the invisible: Learning where the object might be", *CVPR*, 2010.

[2] L. Cerman, J. Matas, and V. Hlavac, "Sputnik Tracker: Having a Companion Improves Robustness of the Tracker", *Image Analysis*, vol. 5575, pp. 291–300, 2009.

[3] O. Miksik, K. Mikolajczyk, "Evaluation of local detectors and descriptors for fast feature matching", *ICPR*, 2012.

[4] L. Cerman and V. Hlavc, "Tracking with context as a semi-supervised learning and labeling problem", *ICPR*, 2012.

[5] T. B. Dinh, N. Vo, and G. Medioni, "Context tracker: Exploring supporters and distracters in unconstrained environments", *CVPR*, 2011.

[6] B. Alexe, N. Heess, Y. W. Teh, and V. Ferrari, "Searching for objects driven by context", *NIPS*, 2012.

[7] Y. Wu, J. Lim, and M.-H. Yang, "Online object tracking: A benchmark", *CVPR*, 2013.

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

[9] Z. Kalal, K. Mikolajczyk, and J. Matas, "P-N Learning: Bootstrapping Binary Classifiers by Structural Constraints", *CVPR*, 2010.

[10] S. Stalder, H. Grabner, and L. J. V. Gool, "Dynamic objectness for adaptive tracking", *ACCV*, 2012.

[11] Z. Sun, H. Yao, S. Zhang, and X. Sun, "Robust visual tracking via context objects computing", *ICIP*, 2011.

[12] J. Whitehill, P. Ruvolo, T. Wu, J. Bergsma, and J. R. Movellan, "Whose vote should count more: Optimal integration of labels from labelers of unknown expertise", *NIPS*, 2009.

[13] M. Yang, Y. Wu, and G. Hua, "Context-aware visual tracking", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 31, no. 7, pp. 1195–1209, 2009.

[14] M. Ozuysal, M. Calonder, V. Lepetit, and P. Fua, "Fast keypoint recognition using random ferns", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 32, no. 3, pp. 448–461, 2010.

[15] S. Hare, A. Saffari, and P. H. S. Torr, "Struck: Structured output tracking with kernels", *ICCV*, 2011.

[16] M. Calonder, V. Lepetit, C. Strecha, and P. Fua, "Brief: binary robust independent elementary features", *ECCV*, 2010.

[17] B. Leibe, A. Leonardis and B. Schiele, "Robust object detection with interleaved categorization and segmentation", *IJCV*, vol. 77, no. 3, pp. 259–289, 2008.

[18] K. Mikolajczyk, H, Uemura, "Action recognition with appearance-motion features and fast search trees", *CVIU* , vol. 115, no. 3, pp. 426–438, 2010.