ROBUST VISUAL TRACKING VIA ADAPTIVE OCCLUSION DETECTION

Yueyang GU, Xiaoguang NIU, Yu QIAO⋆

Intelligence Learning Laboratory, Department of Automation, Shanghai Jiao Tong University, China
Key Laboratory of System Control and Information Processing, Ministry of Education, Shanghai
{guyueyang, 2012657, qiaoyu}@sjtu.edu.cn

ABSTRACT

Occlusion is a special challenge in visual tracking, which may cause target template corrupted by background information. In this paper, we propose an adaptive occlusion detection framework for robust tracking against occlusion. The framework consists of a patch tracker, an occlusion detector, a template updater and a search window predictor. The patch tracker applies KCF-based method to track background patch individually, which may occlude target. The occlusion detector searches for background patches occluding target with an adaptive threshold. The template updater evaluates the occlusion state and applies appropriate target template update strategy. The search window predictor adaptively rescales the size of search window based on occlusion state. Experiments in OTB50 demonstrate that our tracker achieves comparable performance compared with other state-of-art trackers and outperforms them in cases of occlusion.

Index Terms— Visual tracking, occlusion detection, occlusion patch candidate, occlusion mask, adaptive search window

1. INTRODUCTION

Visual tracking is one of the most appealing fields in computer vision [1, 2, 3], etc.. Numerous tracking algorithms have been proposed. Correlation filter (CF) was used in visual tracking [4, 5, 6, 7, 8] and achieved good tracing results. They may well handle slight occlusion with low learning rate, such as slight partial occlusion in one or two frames. But they may fail in heavy or complete occlusion, such as complete occlusion in sequences "Coke" and "Jogging". Because these algorithms have no capacity to identify occlusion and treat occlusion as appearance variation. They update target template with background information in complete occlusion. Consequently, trackers may drift from target or even fail to track since target template has already been corrupted by background information. Part-based trackers [9, 10, 11, 12] divide target into several individual parts and track them simultaneously. The template will be partial updated if tracking confidence of the part is high. However, both appearance variation and occlusion may lead to the drop of tracking confidence. Part-based trackers can not handle fast appearance variation because they stop template update in cases of appearance variation due to low tracking confidence.

In fact, the template update strategy for all tracking methods is totally different in cases of occlusion and appearance variation. The template requires fast update in condition of appearance variation such that the tracker can capture the target variation. On the contrary, the template requires to stop update in condition of occlusion such that the target template can be prevented from corruption of background information. Therefore it is an inevitable problem for all tracking methods to distinguish occlusion and target appearance variation.

In order to deal with occlusion, Niu and Qiao [13] proposed the Context-based Occlusion Detection framework (COD), which is designed for occlusion detection. COD is a KCF-based method only used to track background patches surrounding target. These background patches are treated as context information and may be candidates that occlude target in following frames. Therefore COD is not used for target tracking but provided occlusion information to target tracker for appropriate template update strategy. If the occlusion is detected by COD, the target tracker can stop updating template. COD can be integrated with various tracking approaches [14, 15] and improved their robustness against occlusion. However, there are some drawbacks in COD. Firstly, a fixed number of background patches are randomly selected in current frame, which may cause lack or redundancy of sampled background patches (Fig. 2). Secondly, COD adopted a pre-defined thresholds to evaluate the occlusion patches, which may not work well in various cases. Thirdly, COD selects template update strategy based on the amount of occlusion patches. It may not work if these occlusion patches may overlap at the same location tracked based on different background patches. The slight occlusion may be falsely treated as heavy occlusion.

In this paper, we propose an adaptive occlusion detection framework. It consists of a patch tracker, an occlusion detector, a template updater and a search window predictor. The patch tracker utilizes KCF-based approach to track all background patches that may occlude target. The occlusion

⋆Corresponding author: Yu QIAO (qiaoyu@sjtu.edu.cn). This research is partly supported by NSFC (No: 61375048).

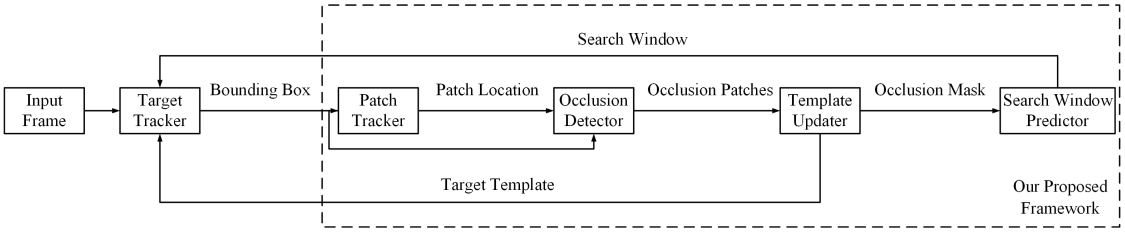
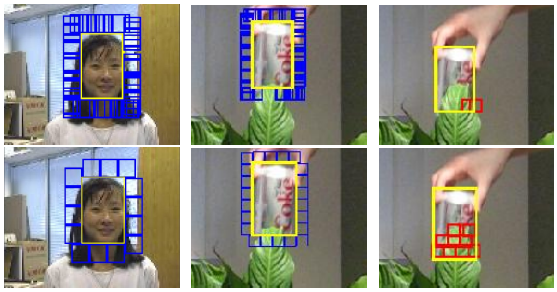


Fig. 1. Adaptive occlusion detection framework.

detector evaluates occlusion patch candidates with an adaptive threshold. The template updater establishes an occlusion mask according to the results of occlusion detection. The target template is updated in cases of non or slight occlusion and stop if heavy or complete occlusion occurs. The search window predictor introduces an adaptive search window by rescaling the search window size in cases of occlusion. In this way, the occluded target may be located in a larger search window when it appears again in the image.

The contributions of this paper are as follow: (1) an occlusion mask is proposed to illustrate occlusion state; (2) the size of search window is adaptively rescaled; (3) an adaptive threshold is used for occlusion patch evaluation; 4) an adaptive strategy is applied to select background patches.



(a) (b) (c)

Fig. 2. Visualized results of patch trackers (first row: COD [13], second row: our method): (a) background patch redundancy in COD; (b) the lack of patches in COD; (c) the occlusion detection results. Yellow: bounding box, blue: background patch, red: occlusion patch.

2. PROPOSED FRAMEWORK

In this section, we present our proposed adaptive occlusion detection framework (Fig. 1) in detailes. In our framework, background patches surrounding target bounding box are treated as context information. The occlusion patch is defined as the background patch obtained in previous frames that occludes the target. Obviously, both occlusion patches and current background patches are key context information for occlusion detection.

An occlusion patch candidate (OPC) set is introduced for occlusion detection in current frame. The current OPC set is regarded as key context information collected from last frame. It consists of two kinds of patches: (1) background patches surrounding bounding box in last frame; (2) occlusion patches of last frame. Obviously, occlusion patches are actually background patches obtained in previous frames. All patches in current OPC set may be the candidates to occlude the target in current frame. Therefore our framework is an adaptive context-based occlusion detection method. After occlusion detection, current OPC set is updated with background and occlusion patches in current frame.

2.1. Patch Tracker

In our framework, the patch tracker is designed to track all patches in current OPC set. These tracked patches are context information collected from last frame. We apply linear KCF tracker [16] as the patch tracker with linear kernel:

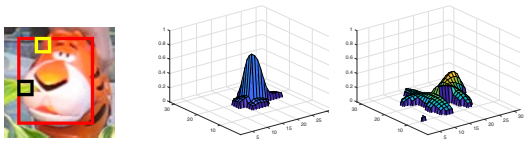
$$k^{xx'} = F^{-1}(\hat{x}^* \odot \hat{x'}), \tag{1}$$

where $\hat{}$ means DFT, F^{-1} means inverse DFT, * means complex conjugation. The patch tracker outputs the location of individual patch of the OPC set in current frame.

2.2. Occlusion Detector

Occlusion indicates that some background patches come into the bounding box. The location of all patches in current OPC set are examined by the occlusion detector. There are two kinds of patches located within the bounding box: (1) the patch occluding target; (2) the patch occluded by target. Fig. 3 presents two kinds of patches in the bounding box and their response map.

Peak-to-Sidelobe Ratio (PSR) [17] is used as the criterion to distinguish the above mentioned patches. This criterion represents not only the intensity of response map but also sharpness of the peak. We set an adaptive threshold for every patches exploiting k historic PSRs of patches locating at the same relative position about target:



(a) (b) (c)

Fig. 3. (a) Two kinds of patch in bounding box, red: bounding box, black: patch occluding target, yellow: patch occluded by target; (b) response map of the patch occluding target; (c) response map of the patch occluded by target.

$$T_i^{(t+k)} = \sum_{j=0}^{k-1} w_j * f(R_i^{t+j}) \tag{2}$$

$$w_j = \frac{e^j}{\sum_{j=0}^{k-1} e^j} \tag{3}$$

where T_i^{t+k} denotes the threshold for ith patch in $t+k$th frame, $f(\cdot)$ means calculating PSR of input map, R_i^{t+j} is the response map of ith patch in $t+j$th frame, w_j is the weight for PSR in $t+j$th frame. In fact, the T_i^{t+k} is the weighted mean of $f(R_i^{t+j})$.

Those patches located within the bounding box are evaluated with PSR and classified into occlusion patch and patches occluded by target. The current OPC set is updated with current context information. Only the patches occluding target is included in updated OPC set. Other patches in previous OPC set are discarded. These patches may be occluded by target or locate outside current bounding box.

The updated OPC set also includes new context information of current frame, which are new background patches surrounding current bounding box. In our framework, the number of background patches to be tracked is determined by the size of current bounding box. The number of each horizontal and vertical side can be obtained by:

$$N_h = \left\lceil \frac{w}{a} \right\rceil \qquad N_v = \left\lceil \frac{h}{a} \right\rceil \tag{4}$$

where $\lceil \cdot \rceil$ means the rounding up operation, w, h are the width and height of bounding box, a is the side length of patch. After N_h, N_v is obtained, all patches are located surrounding the bounding box such that they do not intersect each other and leave no space along four sides of the bounding box. The second row in Fig. 2(a) and (b) shows two examples of background patches. Fig. 2(c) presents the results of occlusion patch detection with COD and our method respectively.

2.3. Template Updater

Before tracking a new frame, the template updater first establishes an update mask as the same size as the input image with *1* denoting target pixels and *0* denoting others. When the occlusion detector outputs the location of occlusion patches, the corresponding elements are set to be *0*. The proportion of non-occluded area is used to evaluate occlusion state and calculated by

$$\varphi = \frac{Sum(M)}{Are(M)} \tag{5}$$

where M is the update mask, $Sum(M)$ is the number of elements *1* in M and $Area(M)$ is the number of all elements in M. The occlusion state is classified into 3 cases with φ: (1) non-occlusion; (2) partial occlusion; (3) heavy occlusion.

Then the template in frame t can be updated by

$$x_t = x_{t-1} * (1-\alpha) + x_{t-1} * \alpha * \delta(\varphi_1 - \varphi) + x_c * \alpha * (1 - \delta(\varphi_1 - \varphi)) \tag{6}$$

where x_{t-1} is the template in frame $t-1$, x_c is the image in current bounding box, α is the learning rate, φ_1 is the threshold to distinguish the states of occlusion and non-occlusion, $\delta(\cdot)$ is unit step function. If the target is occluded, then $\delta(\varphi_1 - \varphi) = 1$, $x_t = x_{t-1}$, the template keeps unchanged. Otherwise, the target template is updated with the learning rate α, $x_t = x_{t-1} * (1-\alpha) + x_c * \alpha$.

2.4. Search Window Predictor

In case of occlusion state (3) (heavy occlusion), tracking results may drift from ground-truth since the appearance of target is covered by background information. Our proposed search window predictor adaptively rescales search window size in order to capture the feature of target when bounding box drifts from ground-truth due to heavy or complete occlusion.

In the first frame, we train n correlation filters exploiting images with different sizes centered on the target. The sizes are selected in the following size pool:

$$S = a_i * S_0 \qquad a_i \in a_1, a_2, ..., a_n \tag{7}$$

where a_i is size coefficients and S_0 is basic size. The search window predictor expands the size of search window in cases of heavy or complete occlusion. It recovers the original window size if occlusion is not observed. Here, we use η_t to denote the temporal state of occlusion in tth frame:

$$\eta_t = \begin{cases} \eta_{t-1} + 1 & \varphi < \varphi_2 \\ max(\eta_{t-1} - 1, 1) & \varphi > \varphi_2 \end{cases} \tag{8}$$

where φ can be obtained by Eq. 5, φ_2 is the threshold to distinguish states of partial and heavy occlusion.The sigmoid function is utilized to calculate the window size index:

$$i = [\frac{n}{1 + e^{0.5*(\eta - b)}}] \tag{9}$$

where $[\cdot]$ means rounding operation, n is the number of sizes in size pool, b is bias which translate the function along the X axis and i is the index of a. Therefore, the size of search window in next frame is $a_i * S_0$.

3. EXPERIMENT AND DISCUSSION

3.1. Set up

To validate the performance of proposed framework, we run our tracker and other state-of-art trackers on OTB50 [1]. In experiments, the size of background patches is 13*13, the critical value φ_1 and φ_2 are 0.75 and 0.25, the length of size pool is 7, bias b is 5. Other parameters are same as the KCF tracker [16].

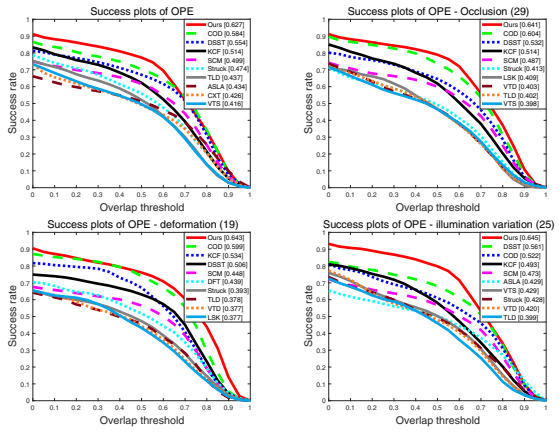


Fig. 4. Successful plot on OTB50. The AUCs of each tracker are presented on the top right corner of plot.

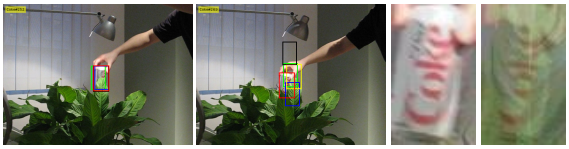
3.2. Quantitative Evaluation

Center Location Error (CLE) is not suitable for different sizes of targets. We select Overlap Ratio (OR) as the metric and draw Success Plot with threshold from 0 to 1. The performance of trackers can by obtained by Area Under Curve (AUC). We conduct One Pass Evaluation (OPE) with our tracker and 32 others trackers (KCF [16], DSST [18], COD [13] and 29 trackers evaluated in [1]) on OTB50. Here we present the top-10 results in Fig. 4. Our tracker achieves the best results and improved 21.98% and 7.36% compared with KCF and COD on overall performance. Attribute-based evaluation (Fig. 4) shows that our tracker also presents competitive performance due to distinguishing occlusion from appearance variation caused by deformation or illumination variation. It is worthy to note that other trackers without occlusion detection module get poor performance on occlusion sequences while our tracker can run better results.

3.3. Qualitative Evaluation

Fig. 5 is a typical example to demonstrate that our occlusion detection framework can avoid template corruption by occlusion object effectively. The coke can moves behind leaves and then move up to leave green leaves. Our template keeps unchanged during occlusion (Fig. 5(c)) while templates of other KCF-based trackers have been corrupted by leaves (Fig. 5(d)). Therefore,only our tracker can track successfully in this case.

Fig. 6 is another example to show the effect of search window predictor. The jogging woman is occluded by the telegraph pole. Other trackers fail to track it after she is not occluded (Fig. 6(d)). Even though the target appears again, they can not re-capture since the target is not in their search window (Fig. 6(c)). Our search window predictor expands the size of search window to ensure that the target may be included in the expanded search window (Fig. 6(c)). In this way, the search window predictor avoid the problem caused by the fixed search window during occlusion.



(a) (b) (c) (d)

Fig. 5. "Coke" sequence: (a) tracking results; (b) tracking results; (c) template of ours; (d) template of KCF. Green: CSK, blue: KCF, black: DSST, yellow: COD, red: our method:

(a) (b) (c) (d)

Fig. 6. "Jogging" sequence: (a) tracking results before occlusion, green: CSK, blue: KCF, black: DSST, yellow: COD, red: our method; (b) occlusion detection results, blue: occlusion patches; (c) search window of trackers; (d) tracking results after occlusion, green: CSK, blue: KCF, black: DSST, yellow: COD, red: our method.

4. CONCLUSION

In this paper we propose an adaptive occlusion detection framework. Our framework utilizes an adaptive threshold to detect occlusion patches. The template will not be corrupted by background information based on occlusion state evaluation. The target will not drift from adaptive search window selected by search window predictor. Experimental results show that our tracker achieve competitive performance against other state-of-the-art trackers and improve the robustness against occlusion.

5. REFERENCES

[1] Yi Wu, Jongwoo Lim, and Ming Hsuan Yang, "Online object tracking: A benchmark," in *Computer Vision and Pattern Recognition*, 2013, pp. 2411–2418.

[2] Yi Wu, Jongwoo Lim, and Ming Hsuan Yang, "Object tracking benchmark," *IEEE Transactions on Pattern Analysis & Machine Intelligence*, vol. 37, no. 9, pp. 1834–1848, 2015.

[3] Naiyan Wang, Jianping Shi, Dit Yan Yeung, and Jiaya Jia, "Understanding and diagnosing visual tracking systems," in *IEEE International Conference on Computer Vision*, 2015, pp. 3101–3109.

[4] João F Henriques, Rui Caseiro, Pedro Martins, and Jorge Batista, "Exploiting the circulant structure of tracking-by-detection with kernels," in *European Conference on Computer Vision (ECCV)*. Springer, 2012, pp. 702–715.

[5] Martin Danelljan, Fahad Shahbaz Khan, Michael Felsberg, and Joost Van De Weijer, "Adaptive color attributes for real-time visual tracking," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2014, pp. 1090–1097.

[6] Yang Li and Jianke Zhu, "A scale adaptive kernel correlation filter tracker with feature integration," in *European Conference on Computer Vision (ECCV)*. Springer, 2014, pp. 254–265.

[7] Chao Ma, Xiaokang Yang, Chongyang Zhang, and Ming Hsuan Yang, "Long-term correlation tracking," in *Computer Vision and Pattern Recognition*, 2015, pp. 5388–5396.

[8] Kunqi Gu, Tao Zhou, Fanghui Liu, Jie Yang, and Yu Qiao, "Correlation filter tracking via bootstrap learning," in *IEEE International Conference on Image Processing*, 2016, pp. 459–463.

[9] Ting Liu, Gang Wang, and Qingxiong Yang, "Real-time part-based visual tracking via adaptive correlation filters," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 4902–4912.

[10] Ting Liu, Gang Wang, Qingxiong Yang, and Li Wang, "Part-based tracking via discriminative correlation filters," *IEEE Transactions on Circuits & Systems for Video Technology*, vol. PP, no. 99, pp. 1–1, 2016.

[11] Si Liu, Tianzhu Zhang, Xiaochun Cao, and Changsheng Xu, "Structural correlation filter for robust visual tracking," in *Computer Vision and Pattern Recognition*, 2016, pp. 4312–4320.

[12] Kunqi Gu, Mingna Liu, Tao Zhou, Fanghui Liu, Xiangjian He, Jie Yang, and Yu Qiao, "Patch-based object tracking via locality-constrained linear coding," in *China Control Conference*, 2016, pp. 7015–7020.

[13] Xiaoguang Niu and Yu Qiao, "Context-based occlusion detection for robust visual tracking," in *IEEE International Conference on Image Processing*, 2017, pp. 3655–3659.

[14] Xiaoguang Niu, Zhipeng Cui, Shijie Geng, Jie Yang, and Yu Qiao, "Robust visual tracking via occlusion detection based on depth-layer information," in *International Conference on Neural Information Processing*, 2017, pp. 44–53.

[15] Xiaoguang Niu, Xingqi Fang, and Yu Qiao, "Robust visual tracking via occlusion detection based on staple algorithm," in *Asian Control Conference*, 2018, pp. 1051–1056.

[16] J. F. Henriques, R Caseiro, P Martins, and J Batista, "High-speed tracking with kernelized correlation filters," *IEEE Transactions on Pattern Analysis & Machine Intelligence*, vol. 37, no. 3, pp. 583–596, 2015.

[17] David S. Bolme, J. Ross Beveridge, Bruce A. Draper, and Yui Man Lui, "Visual object tracking using adaptive correlation filters," in *Computer Vision and Pattern Recognition*, 2010, pp. 2544–2550.

[18] Martin Danelljan, Gustav Häger, Fahad Khan, and Michael Felsberg, "Accurate scale estimation for robust visual tracking," in *British Machine Vision Conference (BMVC), Nottingham, September 1-5, 2014*. BMVA Press, 2014.