AN EFFICIENT OCCLUSION DETECTION METHOD TO IMPROVE OBJECT TRACKERS

Yingkun Xu[*], *Lei Qin*[*], *Guorong Li*[‡], *Qingming Huang*[*‡]

[*]Key Lab of Intel. Inf. Proc., Institute of Computing Technology
Chinese Academy of Sciences, Beijing 100190, China
[‡]University of Chinese Academy of Sciences, Beijing, 100049, China
{yingkun.xu, lei.qin, guorong.li, qingming.huang}@vipl.ict.ac.cn

ABSTRACT

Occlusion is one of the challenging problems in visual tracking. Most of the previous works alleviate this problem by randomly sampled weak features, or analyze it by methods closely related with specific trackers. In this paper, we propose an effective mechanism to detect the occlusion status by random forests, and embed this method into object trackers with a common strategy. We divide the target region into some regular parts, and extract the pairing features within and outside the parts to encode the structure information of the tracked target. The random forests are online trained to discriminate the occlusion status of the parts using occlusion-dependent samples. Several challenging video sequences are used to verify the model, and it proves that our model is capable of recognizing the occlusion status during tracking. The performance of two typical state-of-the-art object trackers is improved by embedding this occlusion detection method.

Index Terms— Visual tracking, occlusion detection, object tracker

1. INTRODUCTION

Visual tracking is an important but difficult topic in computer vision. The purpose of visual tracking is to locate the tracked target and obtain its accurate trajectory. This trajectory acts as one of the middle level features and contains the key motional information for high level analysis. However, visual tracking in real scenario faces many challenging problems, such as illumination changes, pose variations, and occlusions, etc. These problems always cause the template of the target can't describe itself accurately, therefore the object tracker is hard to discriminate the target and distractors in the background.

To cope with above problems, the tracking-by-detection approaches become popular and effective recently [1, 2, 3, 4, 5, 6, 7]. Generally, these trackers generate a basic template when the initial region of the target is selected by the user or an object detector. When the next image arrives, the trackers update this template using online learning algorithms under reasonable assumptions. Commonly, these algorithms implicitly select the representative features which are kept unchanged under complex circumstance, and they works very well when the appearance of target varies gradually. However, the methods do not analyze the status of occlusion explicitly. The appearance will change discontinuously when the serious occlusion takes place, and the trackers which can't discriminate the occlusion and non-occlusion will update the template of target incorrectly. Thus, the erroneous template may cause drift problem. Therefore, it is important to analyze the status of occlusion, which is one of the keys to improve the performance of object trackers as shown in Fig. 1.

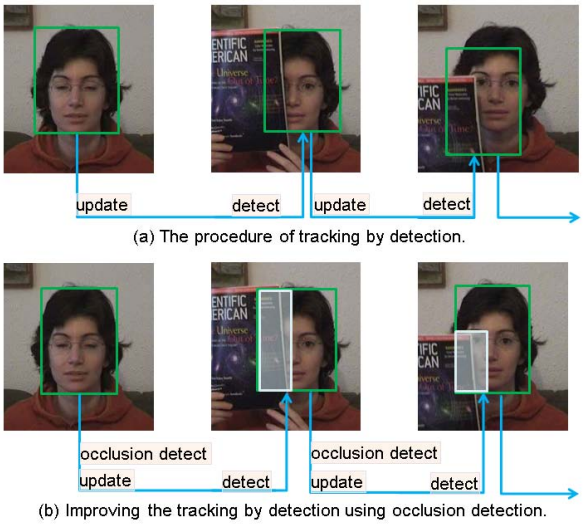


Fig. 1. The general process of tracking by detection, in which occlusion may cause the drift problem if the trackers can't discriminate the status of occlusion and non-occlusion (a), but the tracking can be improved if the occlusion is handled (b).

In this paper, we focus on how to obtain an effective mechanism to detect the occlusion status, and embed the occlusion analysis into the object trackers with a common strategy.

How to handle the occlusion problem is discussed in most of the previous works implicitly or explicitly. For example, the online boosting methods re-weight randomized weak features to emphasize the stable features, while weaken the unreliable ones, such as those in occluded parts [1, 2, 3]. The sparse representation treats the occluded parts as noisy signals in the template [5, 6]. However, serious occlusion always takes place when other objects pass by. In these cases, it is not reasonable to model the occlusion only using implicit reweighting-features or by noisy signals any more. To explicitly analyze how the object is occluded or which parts are occluded during tracking, the generative models and discriminative models are employed [8, 9]. One generative model aims to construct the corresponding between the pixels in the current image with those in previous image, and the occluded pixels are labeled with the identity of the tracked objects [8]. To discriminate whether an object is occluded or not, the target region can be divided into some cell grids, and the occlusion status of each cell is determined by an SVM classifier trained offline [9].

Our proposed method adopts random forests to discriminate the occlusion status for the regular parts of the target. Because most of

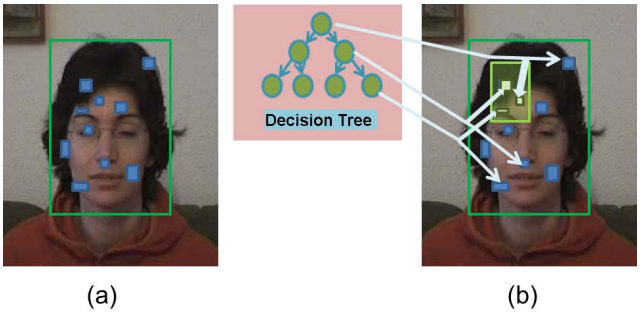


Fig. 2. The pairing Haar-like features to train the decision tree for random forests. (a) The Haar-like representation for tracked target. (b) The Haar-like features to train the random forests to discriminate the occlusion status.

the object trackers (such as compressive tracker [4] and L1-tracker [5, 6]) extract the features from regular target region, they can extract the features from these parts. Thus, our method can be embedded into these object trackers to improve their performance. Although the discriminative model is also used in [9], that method is dependent on patterns of the targets, and closely related with the object tracker, whereas our method has no these limitations.

The remainder of this paper is organized as follows. In section two, we present how to construct the random forests for occlusion analysis, and how to embed the model into object trackers with a common strategy. In section three, we show the performance tuning of occlusion detection and the improvement for the two of typical state-of-the-art tracking methods. Finally, we conclude this paper.

2. OCCLUSION DETECTION AND ITS EMBEDDING

We adopt discriminative model to analyze the occlusion status for the tracked target. To improve the common object trackers with occlusion analysis effectively, we divide the target region into some regular parts, and the status of each part is classified as visible or occluded.

The random forests method is employed by the reason of its high speed, parallelism, and convenience to update. Therefore, we train a random forest for each part and determine which parts are reliable for tracking.

2.1. Learning random forests for occlusion detection

Because a decision tree has low bias but high variance when it is generalized, random forests are proposed to obtain more stable models by reducing the variance of the trees [10]. They average a large of de-correlated trees and obtain the bagging models. Because of its robustness for noise and distracting samples, Random Forests are used successfully for object tracking problem [11, 12].

To construct a random forest, features with different sequences are randomly selected from feature space. The bootstrapped samples are drawn to train the trees separately using these feature sequences. Given one sample (X, y) composed of the feature vector X and its label $y, y \in \{0, 1\}$, the binary test, which makes the sample be classified further, on node i for one tree t is:

$$h_i^t(X, \theta_i^t) = \begin{cases} 1 & g(f_i^t(X, P_a), f_i^t(X, P_b)) > \theta_i^t \\ 0 & \text{otherwise} \end{cases} \quad (1)$$

where g is a comparing function, $f_i^t(X, P), i = 1...M$ extracts the sub-feature set in the feature vector X from position P, and θ_i^t is the threshold. All of the binary tests $h_i^t(X, \theta_i^t)$ compose the classifier $H_t(X, \Theta_t)$ for the tree t [10]. The overall discriminant function of the random forest can be formulated as:

$$H(X) = \frac{1}{T} \sum_{t=1}^{T} H_t(X, \Theta_t) \quad (2)$$

where T is the number of the trees.

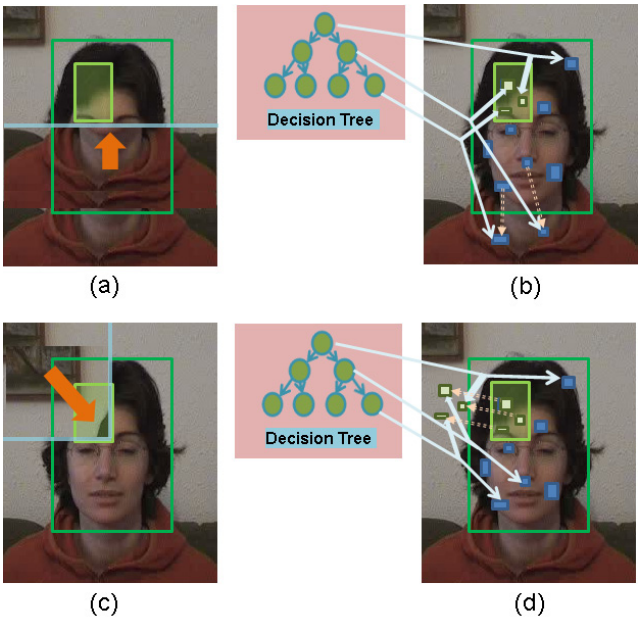


Fig. 3. Contextual Haar-like pairing features for positive and negative samples to discriminate the occlusion status with occlusion simulation. (a) The partial occlusion simulation with the part we concerned visible. (b) The contextual pairing features for the case in (a). (c) The partial occlusion simulation with the part we concerned occluded. (d) The contextual pairing features for the case in (c).

The features and its sub-features extraction for binary test Eq. (1) always compose a weak classifier. For example, in P-N learning, the difference between the intensity values of randomized two points is used to construct the binary test with threshold zero. Unlike that, we adopt the Haar-like combining features because of its effectiveness of implementation, scalability and more robustness than point-based feature. We represent the tracked target by randomly sampled Haar-like pairing features as in Fig. 2(a). However, we should not use any feature pairs without location constraint for our classifier, because our task is to discriminate whether one part is occluded or not. As in Fig. 2(b), to construct one feature pair, we select the first feature of the pair, $f_i^t(X, P_a)$ in Eq. (1), randomly within the concerned part, and the second feature, $f_i^t(X, P_b)$ in Eq. (1), from outside of the part but within the target region. Thus, relative structures between the part and the other region of the target are encoded by the random forests through combination of the inner and outer features. The occlusion status is determined by analyzing whether the relative structures are changed or not.

2.2. Contextual pairing features

To train the random forests for discrimination of occlusion status, we first draw the samples around the tracking result in each frame. However, these samples are not enough for training the classifier to determine the occlusion status. They can be considered as the cases

of full occlusion, and actually are only the subset of what we need to train our classifiers.

In order to obtain more reasonable samples, a straightforward method to construct the samples is to simulate the occlusion situations. For one part which is concerned in the target region, we implement this simulation through copying the surrounding patches out of the target from different directions to cover interior patches of the target region, for example, from bottom of the target as in Fig. 3(a). However, this simulation is constrained and costly. Instead of this straightforward patch copy, we translate one of the feature pair in Eq. (1) from original position to outside of the target region as in Fig. 3(b). These pairing features with one translated are called contextual pairing features.

There are two types of contextual pairing features: the non-occlusion case for concerned part and the occlusion case for it. The former is considered as positive ones and the latter is negative ones. In order to construct the positive contextual pairing features, the translations of four directions are considered: from left, from right, from top and from bottom. Each translation from one direction will swap the outer feature $f_i^t(X, P_b)$ at that side with the feature $\bar{f}_i^t(X, \bar{P}_b)$ out of the target in that direction. For example, Fig. 3(a) and Fig. 3(b) illustrate the example of translation from bottom. The probability of the swap $w(f_i^t(X, P_b), \bar{f}_i^t(X, \bar{P}_b))$ is related with their relative distance, and can be formulated as:

$$p(w(f_i^t(X, P_b), \bar{f}_i^t(X, \bar{P}_b))) \propto \exp(-\lambda||P_b - \bar{P}_b||_2) \quad (3)$$

In order to construct the negative contextual pairing features. The four directions of translation are adopted: from top-left, from top-right, from bottom-right, and from bottom-left. Each translation will overlap the concerned part completely. Thus we swap all of the inner features $f_i^t(X, P_a)$ to the neighbor regions outside of the target in the four different directions. For example, Fig. 3(c) and Fig. 3(d) illustrate the example of translation from top-left. The probability of the swap is similar with Eq. (3).

2.3. Embedding occlusion detection into object trackers

In order to embed the occlusion analysis into the object trackers, we assume that the target region can be represented with rectangle or rectangle after affine transformation. This assumption is established for most of the state-of-the-art methods [1, 2, 3, 4, 5, 6].

In the tracking problem, given each new frame, we want to obtain the optimal position of our target object. Generally, the template $\bar{Q}$ of the target is learned with some features, and the similarity or likelihood $S(\bar{Q}, Q(x))$ for one candidate region $Q(x)$ is calculated.

In our method of occlusion detection, the target region is divided into N regular parts. We can obtain the label y_i for the occlusion status of each part $i, i = 1...N$, where non-occlusion status is labeled as positive. Thus, the overall similarity or likelihood can be calculated for this multi-part model as:

$$S(\bar{Q}, Q(x)) = \frac{\sum_{i=1}^{N} y_i S(\bar{Q}_i, Q_i(x))}{\sum_{i=1}^{N} y_i} \quad (4)$$

In the experiments, we will show this weighting formulation is rational and effective for improving object trackers.

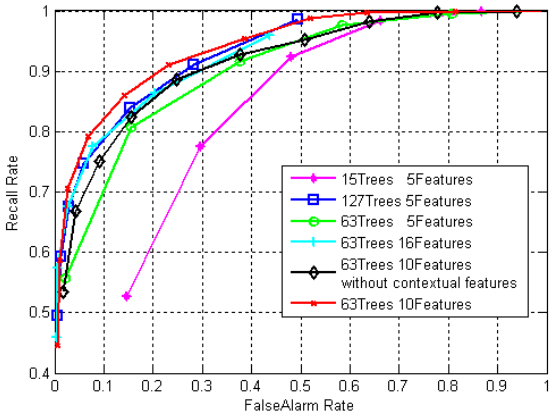


Fig. 4. The ROC curves of performance for occlusion detection with different settings using random forests.

3. EXPERIMENTS

We evaluate our methods with experiments in two aspects. First, we illustrate the effectiveness of occlusion detection based on the tracking ground truth. Second, the two state-of-the-art object trackers are employed and embedded with the occlusion analysis, and their tracking performance is improved manifestly on several changing datasets.

3.1. The evaluation of occlusion detection

In our model for occlusion detection, we utilize the randomized threshold for the binary test of Eq. (1), and thus the extremely randomized forests are adopted [13]. The parameter λ in Eq. (3) is set to 0.06. Because the numbers of decision trees and features are the key settings for random forest, we need to confirm the appropriate values for them.

In this experiment, we use *face1* sequence [14] to verify these configurations, and the tracked target, as in Fig. 1, is divided into regular 3x3 parts. Because the unstable tracking process may impose extra erroneous structure information, we use the ground truth of tracking to evaluate the results of occlusion detection for the parts. The different configuration combinations of 15∼127 trees and 5∼16 pairing features per tree are tested. We omit the contextual pairing features for the best configuration to show the drops of performance, thus, to prove the importance of these simulated features. In Fig. 4, we illustrate ROC curves for the results of some typical configurations.

We can see from the experimental results shown in Fig. 4, although the performance is better when there are more trees, the performance does not increase remarkably when the number of the trees is larger than 63, whereas increasing the number of the features can promote the performance, but some problems, such as overfitting, will arise when the depth of trees is larger than 10. Thus, the configuration of 63 trees and 10 features is a better choice.

3.2. Embedding the occlusion analysis into object trackers

To evaluate the occlusion detection method, we present our experiments on six challenging video sequences. Five of these sequences are chosen from public dataset: *face1* [14], *iLids* [15], *TUD* [16], *face2* [3], *caviar* [17]. The last sequence, *driving*, is downloaded

	OE-CT	CT	OE-APGL1	APGL1	FRAG	MIL
face1	0.127	0.276	0.067	0.068	0.076	0.255
iLids	0.208	0.795	0.096	0.148	0.072	0.222
TUD	0.088	0.424	0.044	0.172	0.055	0.453
face2	0.119	0.177	0.085	0.088	0.096	0.157
caviar	0.255	0.347	0.067	0.281	0.104	0.243
driving	0.256	0.487	0.064	0.140	0.149	0.455

Table 1. The experimental comparison between CT, APGL1, FRAG, MIL and our improved object trackers: OE-CT and OE-APG1 on six video sequences with serious occlusion. The average errors after normalization are utilized to evaluate the tracking results.

from Youku. These sequences contain serious occlusion cases for faces, pedestrians and cars.

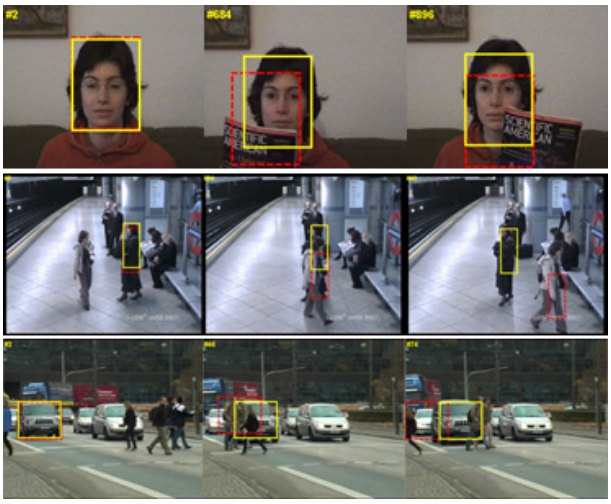
We embed our occlusion analysis method into the two state-of-the-art object trackers: the compressive tracker (CT tracker) [4] and robust L1 tracker using accelerated proximal gradient approach (AGPL1 tracker) [6], which are two of typical state-of-the-art object tracking methods. The concrete embedding methods are referred as Eq. (4). The parameters for the random forests are similar in all experiments as explained previously.

We call the improved object trackers for CT tracker and AGPL1 tracker as OE-CT and OE-APGL1. To illustrate the merits of our method, we compare the methods with online multiple instance learning (MIL) [3] and fragments based tracking (FRAG) [14]. The two methods are robust to occlusion as the authors explained, and we use the codes from their home pages for comparison.

The average tracking errors between the ground truths and tracking results are utilized to evaluate the experimental results. We normalize the average errors using the targets' size of ground truth for fair comparison. The table 1 gives the experimental results.

As the illustration from Table 1, the occlusion embedding methods, OE-CT and OE-APGL1, can improve the tracking results than original two methods respectively. Both of them have more accurate results than that of MIL in most cases, whereas OE-APGL1 is better than OE-CT. The method FRAG adopts the specific mechanism for handling occlusion, and it has more stable results than OE-CT, whereas OE-APGL1 achieves best results except in the sequence of *iLids*, in which color features is outstanding than other features. Fig. 5 shows some tracking examples of OE-CT and OE-APGL1 and their original methods CT and APGL1.

In the Fig. 5, we observe that the serious occlusion leads to disturbance for object trackers. The tracking even drifts for the targets in the sequences of *iLids*, *TUD* and *driving*. The feature reweighting process according to explicit occlusion analysis corrects the erroneous tracking and drifting for these sequences, and improves the tracking results.



(a) Tracking examples using CT and OE-CT

(b) Tracking examples using APGL1 and OE-APGL1

Fig. 5. The examples of tracking results with original object trackers and ones embedding our occlusion detection. (a) The tracking results of CT tracker [4] and its updated version OE-CT on the three video sequences of *face1*, *iLids* and *TUD*. (b) The tracking results of APGL1 [15] and its updated version OE-APGL1 on the three video sequences of *face2*, *caviar* and *driving*. In all six examples, the yellow solid boxes indicate the results of updated trackers, whereas the red dashed boxes indicate the results with original trackers.

4. CONCLUSION

We propose an effective method using random forests model to detection occlusion status. The efficient Haar-like paring features combining with their extensive contextual pairing features are proposed to train the random forests. The experimental results show these features can detect the occlusion status for the parts of target region. We embed the occlusion detection method into two of the state-of-the-art object trackers, and the results are improved manifestly based upon several challenging video sequences, which all have serious occlusion situation. This proves the effectiveness of our occlusion detection method and its improvement for the object trackers.

5. ACKNOWLEDGEMENT

This work was supported in part by National Basic Research Program of China (973 Program): 2012CB316400, in part by National Natural Science Foundation of China: 61025011, 61035001, 61003165 and 61133003, and in part by Beijing Natural Science Foundation: 4111003.

6. REFERENCES

[1] H. Grabner, and H. Bischof. " On-line boosting and vision." In *IEEE CVPR*, pp. 260-267, 2006.

[2] G. Li, Q. Huang, J. Pang, S. Jiang, and L. Qin "Online Selection of the Best k-Feature Subset for Object Tracking." In *JVCIR*, vol. 23, no. 2, pp. 254-263, 2012.

[3] B. Babenko, M. Yang, and S. Belongie "Robust Object Tracking with Online Multiple Instance Learning." In *IEEE TPAMI*, vol. 33, no. 8, pp. 1619-1632, 2011.

[4] K. Zhang, L. Zhang, and M. Yang "Real-time compressive tracking." In *ECCV*, 2012.

[5] X. Mei and H. Ling "Robust visual tracking using l1 minimization." In *IEEE ICCV*, 2009.

[6] C. Bao, Y. Wu, H. Ling, and H. Ji "Real Time Robust L1 Tracker Using Accelerated Proximal Gradient Approach." In *IEEE CVPR*, 2012.

[7] G. Li, L. Qin, Q. Huang, J. Pang, and S. Jiang "Treat samples differently: Object tracking with semi-supervised online CovBoost." In *IEEE ICCV*, 2011.

[8] N. Papadakis, and A. Bugeau "Tracking with Occlusions via Graph Cuts." In *IEEE TPAMI*, vol. 33, no. 1, pp. 144-157, 2011.

[9] S. Kwak, W. Nam, B. Han, and J.H. Han "Learning Occlusion with Likelihoods for Visual Tracking." In *IEEE ICCV*, 2011.

[10] L. Breiman "Random forests." In *Machine Learning*, 2001.

[11] A. Saffari, C. Leistner, J. Santner, M. Godec, and H. Bischof "On-line random forests." In *IEEE Proc. On-line Learning for Computer Vision Workshop, ICCV*, 2009.

[12] Z. Kalal, K. Mikolajczyk, and J. Matas "Tracking-Learning-Detection." In *IEEE TPAMI*, vol. 34, no. 7, pp. 1409 - 1422, 2012.

[13] P.Geurts, D. Ernst, and L.Wehenkel "Extremly randomized trees." In *Machine Learning*, vol. 63, pp. 3-42, 2006.

[14] A. Adam, E. Rivlin, and I. Shimshoni "Robust fragments-based tracking using the integral histogram." In *IEEE CVPR*, 2006.

[15] iLIDS: http://www.elec.qmul.ac.uk/staffinfo/andrea/avss2007_d.html.

[16] M. Andriluka, S. Roth, and B. Schiele. "People-tracking-by-detection and people-detection-by-tracking." In *IEEE CVPR*, 2008.

[17] CAVIAR: Context Aware Vision using Image-based Active Recognition. http://homepages.inf.ed.ac.uk/rbf/CAVIAR/.