World Academy of Science, Engineering and Technology
International Journal of Computer and Information Engineering
Vol:15, No:7, 2021

Fast and Robust Long-term Tracking with Effective Searching Model

Thang V. Kieu, Long P. Nguyen

***Abstract*—**Kernelized Correlation Filter (KCF) based trackers have gained a lot of attention recently because of their accuracy and fast calculation speed. However, this algorithm is not robust in cases where the object is lost by a sudden change of direction, being obscured or going out of view. In order to improve KCF performance in long-term tracking, this paper proposes an anomaly detection method for target loss warning by analyzing the response map of each frame, and a classification algorithm for reliable target re-locating mechanism by using Random fern. Being tested with Visual Tracker Benchmark and Visual Object Tracking datasets, the experimental results indicated that the precision and success rate of the proposed algorithm were 2.92 and 2.61 times higher than that of the original KCF algorithm, respectively. Moreover, the proposed tracker handles occlusion better than many state-of-the-art long-term tracking methods while running at 60 frames per second.

***Keywords*—**Correlation filter, long-term tracking, random fern, real-time tracking.

I. Introduction

VISUAL object tracking is one of the most interesting research topics in the field of computer vision because of its wide applications in both civilian and military areas such as traffic monitoring, augmented reality, human-computer interaction, automated surveillance, etc. In recent years, researchers have paid much attention to the correlation filter-based tracking algorithm [1]-[3]. This algorithm has two main advantages: (i) the computational efficiency by using the characteristics of the cyclic matrix to perform the correlation process in the Fourier transform domain and (ii) the target information constantly updated thanks to the online training process at every frame. However, this type of algorithm - KCF [2] - is not robust in cases where the object is lost by a sudden change of direction, being obscured or going out of view. Once the target is lost, it is very difficult to recover the target tracking.

Target re-detection can be divided into two main research directions based on searching area criteria: local or global searching on the recent image where target lost. Idea of Shin et al. [4] is based on the assumption that the lost target will reappear in the vicinity of the last target position, to construct a 3x3 pixel grid to capture the target which has the maximum response. Moreover, Liu et al. [5] construct the grid over the entire image and then move the grid a quarter of its size to increase the chances of hitting the target. However, this method has a high computation cost and is not suitable for real-time applications. Another approach uses saliency detection [6] to relocate the target. The range of the searching area will increase over time since the target lost track. However, this solution only works well in case of thermal images, where the saliency objects have a high contrast compared to the background.

This paper proposes (i) an anomaly detection method for target loss warning by analyzing the response map of each frame and (ii) a classification algorithm for reliable target re-locating mechanism by using Random fern to improve KCF tracker so that it can recover the lost target.

II. KCF Tracking Algorithm

We denote X as the feature matrix obtained from tracking window and each row of X is one sample x_i. We consider y_i as the desired correlation output corresponding to a given sample x_i. The goal of training is to find a function $f(z) = q^T z$ that minimizes the squared error between samples x_i and their regression targets y_i:

$$\underset{q}{\operatorname{argmin}} \sum_i (f(x_i) - y_i)^2 + \lambda \|q\| \tag{1}$$

The λ is a regularization parameter that controls overfitting in linear regression. Because the large data set can lead to the relationship between them being non-linear, it is necessary to map the data set to a high-dimensional space using the kernel trick:

$$f(z) = q^T z = \left(\sum_{i=1} \alpha_i \phi(x_i)\right)^T z = \sum_{i=1} \alpha_i \phi^T(x_i) \tag{2}$$

Now, the problem has changed from finding the optimal coefficient q to finding α. Using a linear kernel trick to simplify the calculation, the solution of the problem can be expressed as:

$$\alpha_F = \frac{Y_F}{\overline{X_F} * X_F + \lambda} \tag{3}$$

where X_F, Y_F, α_F denote the Fourier Transform of X, Y, α, respectively, and $\overline{X_F}$ is the complex-conjugate of X_F.

The response r from the KCF algorithm represents the similarity of the sample at the current frame and the Z_F sample of the next frame:

$$r = max\left(F^{-1}(\alpha_F * X_F * Z_F)\right) \tag{4}$$

Equation (4) indicates that the response value contains information about the target in two consecutive frames. First,

Thang V. Kieu and Long P. Nguyen are with the Viettel High Technology Industries Corporation, Hanoi, 100000 Vietnam (e-mail: thangkv@viettel.com.vn, longnp91@ viettel.com.vn).

we use this value to determine when the target has lost track, and at that same time, a global searching mechanism is performed to relocate the target.

III. Improved Tracking Algorithm

The proposed algorithm is shown in the block diagram in Fig. 1 where the blue area is the original KCF algorithm, the orange area corresponds to the target loss warning block, and the green area corresponds to the target re-detection block.

A. Target Loss Warning

The KCF algorithm locates the target in the next frame by choosing the position with the maximum response *r*. Therefore, analyzing this value is an efficient approach to predict whether the target loses track or not.

Fig. 1 Flowchart of the proposed tracking algorithm

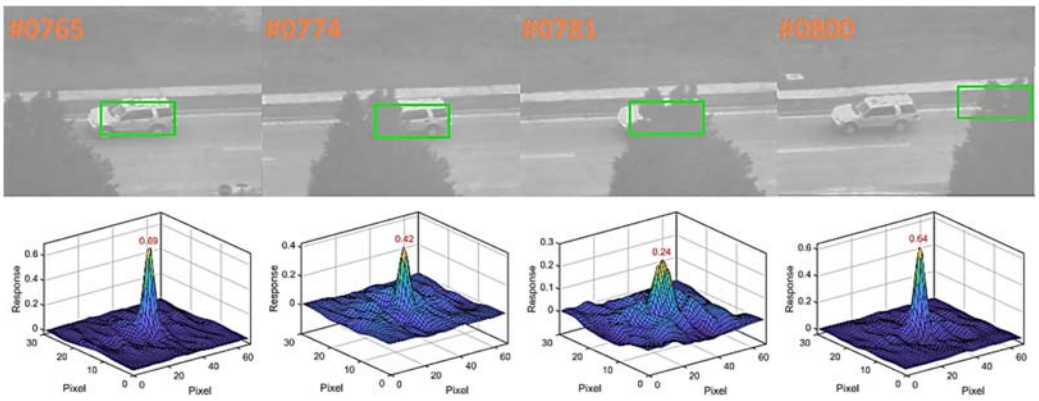

Fig. 2 Response value distribution of video sequence

The graph of response in Fig. 2 shows that from the moment the target is partially obscure (frame #0774) until completely lost (frame #0781), the response peak *r* drops sharply, and the response matrix also exhibited more non-uniformity. In the next frame, the response peak and the response matrix recover and return to the original state (frame #0800 compared to #0765). Note that even though the object has lost its track completely (#0781), the algorithm still holds onto the tree. When the target passes through the tree, the algorithm still runs the training process and updates continuously on the tracking model. Therefore, if the training time is large enough, the algorithm will recognize the tree as the target.

The characteristic of the response matrix is modeled by:

$$peakratio_{frame} = \frac{|r_{frame} - mean_{frame}|}{std_{frame}} \quad (5)$$

where r_{frame}, $mean_{frame}$, std_{frame} are the maximum response value, the average response value, and the standard deviation of the response matrix in the current frame, respectively. We analyze the $peakratio_{frame}$ value from *n* consecutive frames as a basis for making a decision whether the subject is lost in the current frame or not. The selection of *n*-value must be large enough to increase the accuracy of outlier peak detection in the global area. Then, we remove the noise on the sequence of values obtained from the $(t - n)^{th}$ to t^{th} frame with a Gaussian filter. This step aims at eliminating small noisy peaks that are not really the points where the target loses track:

$$u = imgaussfiltu \quad (6)$$

where $u = peakratio_{frame(t-n:t)}$.

When the target loss occurs, r_{frame} value decreases drastically while $mean_{frame}$ and std_{frame} tend to increase. Therefore, as a consequence, $peakratio_{frame}$ decreases rapidly in the occlusion case, compared with the ordinary values. We amplify the difference between these values using the exponential:

$$u = exp(-u) \quad (7)$$

In (7), the negative sign is used to reverse the graph. Thus, the $peakratio_{frame}$ values in the normal state will be approximately 0, otherwise much higher than 0 in the occlusion case. We use sample Suv in data set TB [9] to check the response characteristics of the target in case of target loss occurs. The results are shown in Fig. 3.

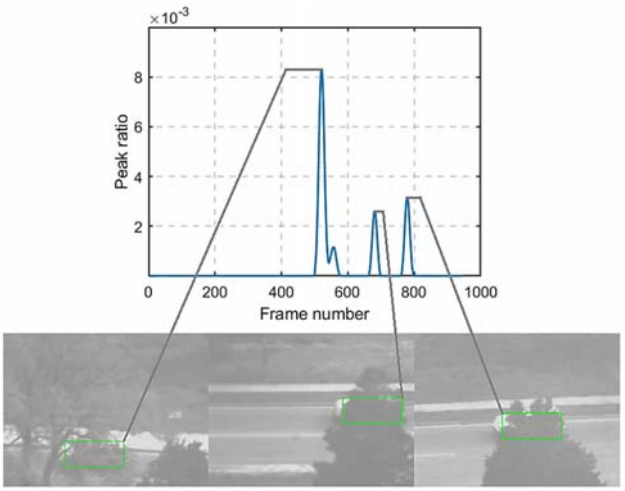

Fig. 3 Flowchart of the proposed tracking algorithm

The results indicate that when the object is obscured, the graph has a completely different form than the rest. We use the Median Absolute Deviations definition to detect that outliers peak. The values are evaluated as an abnormal peak if the following conditions are met:

$$|peakratio_{frame} - median(u)| \geq k * MAD \quad (8)$$

where *median(u)* is the median value on the frame sequence, *k* is the threshold factor and MAD is determined by:

$$MAD = \frac{-median|u - median(u)|}{\sqrt{2} * erfcinv\left(\frac{3}{2}\right)} \quad (9)$$

Using such an evaluation method, we determined three times when the target obscured corresponding to three abnormal peaks as shown in the graph in Fig. 3.

B. Target Re-detection

After determining when the target has lost track, it is necessary to search the target in the entire image. The idea here is to use a classifier to divide the image into two groups: the group contains the target, and the other is the background. The training for the classifier will take place at the same time as the tracking process: pixels of the target inside the tracking window are marked as positive sample, and the rest of the image is considered as negative sample. With a sufficiently large number of training samples, the classifier learns the target in many rotation states, different shapes, or changing luminous intensity. Thus, when given to the classifier unspecified image, the classifier is able to find the target if it appears in the current frame.

Based on such an idea, we propose to use the Random fern classifier, which is more efficient when compared with equivalent classifiers such as Decision tree, Random Forest [7]. Moreover, it effortlessly balances between accuracy and computation time for real-time applications rather than other classifiers such as Support vector machine, K nearest neighbors, Artificial Neural Network, Expectation-Maximization, etc.

We use the sliding window to sample the entire image as input for the Random fern classifier. The idea of computation on the grid is inspired by the facial recognition algorithm using the Haar-like feature [8]. These cells in the grid map have overlapping areas, as shown in Fig. 4.

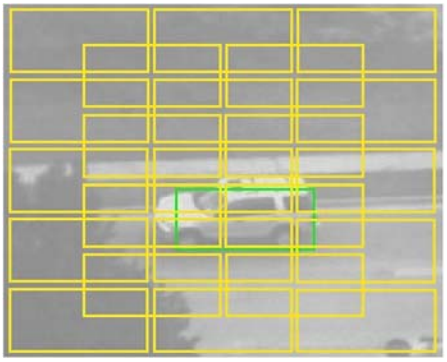

Fig. 4 Illustration of a scanning-window grid

Input data of Random fern are built based on the binary test method: First, label the sample, then randomly take on it *N* pairs of points and determine the f_i features based on the *pixel comparisons*. If the digital number difference between these points is higher than 0, we label that sample as $fi = 1$ and vice versa $f_i = 0$. Converting the binary string to a decimal value will result as the input of the classifier, called the posterior - an array of 2^s values initialized by the value 0, where *s* is the number of trees, then these values will be updated at every training phase. The feature extraction process is shown in Fig. 5.

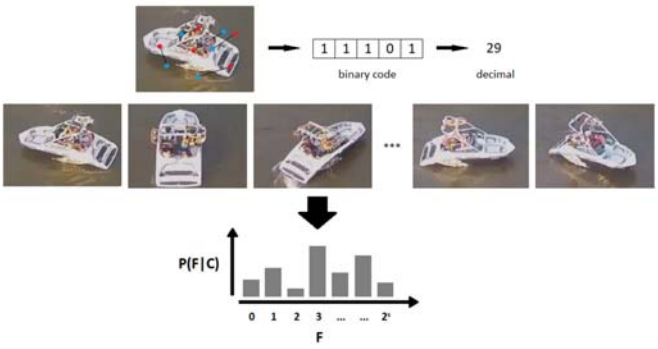

Fig. 5 Working principle of Random fern

We consider *C = {"positive", "negative"}* is a set of classes to classify. The label of each sample is obtained from:

$$\underset{C_i}{\operatorname{argmax}}\{\, P(C_i|f_1, f_2, f_3, \dots, f_N)\,\} \tag{10}$$

Using Bayes' theorem, which can be expressed as:

$$P(C_i|f_1, f_2, \dots, f_N) = \frac{P(f_1, f_2, \dots, f_N|C_i)*P(C_i)}{P(f_1, f_2, \dots, f_N)} \tag{11}$$

To collect and store a large number of features, we divide features into M groups of size $S = N/M$. Assuming that the groups F_k are independent of each other, the conditional probability becomes:

$$P(f_1, f_2, f_3, \dots, f_N|C_i) = \prod_{k=1}^{M} P(F_k|C_i) \tag{12}$$

The sample with the highest probability evaluated by Random fern will be calculated response r against the last sample of the target before loss track. If the response characteristic satisfies the inverse condition of (8), the target is successfully re-initialized:

$$|peak\_ratio_{frame} - median(u)| \geq k * MAD \tag{13}$$

IV. Experimental Results and Discussion

We evaluate the proposed method on 12 challenging sequences from TB [9] and VOT [10] benchmark dataset. Then we compare our algorithm with other six state-of-the-art trackers which are designed for long-term tracking purpose [11]: MIL [12], CSRT [13], BOOSTING [14], TLD [15], MOSSE [16] and KCF [17]. Algorithms like BOOSTING and MIL are based on the tracking by detection principle, using a classifier that separates the target from the background for training, then identify the target at the next frame by selecting the position with the highest probability within the vicinity. TLD uses the results from the tracker to provide training data to update the classifier, and the classifier re-initializes the tracker when it fails.

A. Parameters and Setting Environment

The size of the search window for moving estimation is set to 4 times of target size. The target loss warning and target re-detection mechanism are set as follows: $n = 1000$, $\sigma = 2$, $k = 3$ and $M = 10$, $S = 10$, sliding step = 0.3 times tracking window size. We use the same setting of parameter for all sequences. Our implementation runs at about 60 frames per second on a computer with an Intel Core i7-8700K, the main frequency is 3.7 GHz, and the memory is 16 GB.

B. Assessment Metrics

In this paper, two methods [18] are used as the performance evaluation criteria of all tracking algorithm: (i) distance precision, which shows the percentage of frames whose estimated location is within the given threshold distance of the ground truth, and (ii) overlap success rate, which is defined as the percentage of frames where the bounding box overlap surpasses a threshold.

C. Performance Analysis

Fig. 6 shows the overall precision and success plot of seven tracking algorithms. From the graph, it is easy to see that the performance of the proposed algorithm ranks first and outperforms the other state-of-the-art-trackers. The precision and success rate of the proposed algorithm are 1.33 and 1.25 times higher than the second algorithm (MIL), and the performance is improved by 2.92 times and 2.61 times, respectively, compared to the traditional KCF algorithm.

Fig. 7 is a representative tracking result of several challenging sequences. The results in Fig. 7 explain the outperform tracking performance of the proposed algorithm. In the *Car chase* (frame #0700) and *Suv* case (frame #0580), after the targets are completely occluded and then reappear, only the proposed and TLD, or BOOSTING algorithm successfully track the targets; in the Coke case (frame #0280) only the proposed and CSRT algorithm can relocate the targets. However, the TLD and BOOSTING track the wrong targets in Box case (frame # 0420), Coke (frame # 0220) before it is occluded. The CSRT performs well in handling targets successfully in all cases before it is occluded, but it drifts when the targets undergo heavy occlusion in Box (frame #0500), Car chase (frame #0700) and Suv case (frame #0580). Meanwhile, the proposed algorithm successfully tracks in both partially obscured cases such as Box and completely obscured such as Car chase, Coke, Suv because it adds the target loss re-detection mechanism, and shows strong anti-occlusion ability.

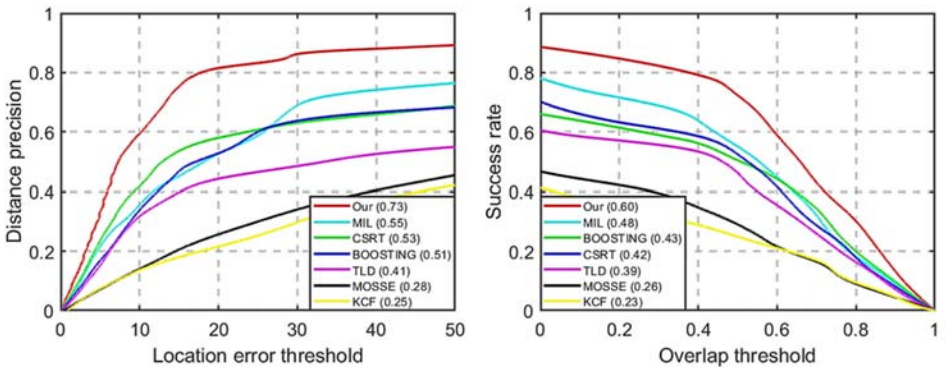

Fig. 6 Precision plot and success plot qualitative analysis

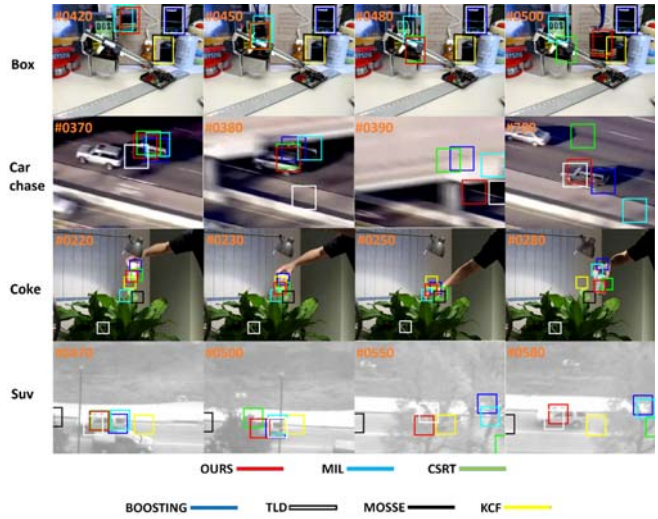

Fig. 7 Partial tracking results with target occlusion

V. Conclusion

In this paper, we propose an effective target loss re-detection mechanism to improve KCF tracker so that it can recover the lost target. By analyzing response map of each frame, we develop an efficient method to detect the loss track using the principle of outlier detection. We further develop a robust online classifier to re-detect target in case of tracking failure. Experimental results show that the proposed algorithm works as a promising alternative for long-term tracking tasks and performs favorably against state-of-the-art methods in terms of efficiency, accuracy, and robustness while still running at about 60 frames per second.

References

[1] D. S. Bolme, J. R. Beveridge, B. A. Draper and Y. M. Lui, "Visual object tracking using adaptive correlation filters," *Proceedings of IEEE conference on computer vision and pattern recognition*, pages 2544-2550, 2010.

[2] J.F. Henriques, R. Caseiro, P. Martins and J. Batista, "Exploiting the circulant structure of tracking-by-detection with kernels," *European Conference on Computer Vision*, Springer: Berlin/Heidelberg, Germany, pages 702–715, 2012.

[3] J. F. Henriques, R. Caseiro, P. Martins and J. Batista, "High-speed tracking with kernelized correlation filters," *IEEE Trans. Pattern Anal. Mach. Intell*, volume 37, pages 583–596, 2014.

[4] J. Shin, H. Kim, D. Kim and J. Paik, "Fast and Robust Object Tracking Using Tracking Failure Detection in Kernelized Correlation Filter," *Applied Sciences*, 2020.

[5] Y. Liu, Y. He, Q. Tian and J. Yang, "KCF Tracking Algorithm Based on Outlier Detection," in *Springer*, vol 752, 2019.

[6] X. Xue, Y. Li, H. Dong and Q. Shen, "Robust Correlation Tracking for UAV Videos via Feature Fusion and Saliency Proposals," *Remote Sensing*, 2018.

[7] M. Özuysal, P. Fua, V. Lepetit, "Fast Keypoint Recognition in Ten Lines of Code," *IEEE*, 2007.

[8] P. Viola, M. Jones, "Rapid Object Detection using a Boosted Cascade of Simple Features," *IEEE*, 2001.

[9] http://cvlab.hanyang.ac.kr/tracker_benchmark/datasets.html, accessed on 9/2020.

[10] https://www.votchallenge.net/vot2020/dataset.html, accessed on 9/2020.

[11] https://docs.opencv.org/4.2.0/d0/d0a/classcv_1_1Tracker.html, accessed on 9/2020.

[12] B. Babenko, M. H. Yang, and S. Belongie, "Visual tracking with online multiple instance learning," *Computer Vision and Pattern Recognition*, 2009, pages 983–990, 2009.

[13] A. Lukezic, T. Voj'ir, L. C. Zajc, J. Matas, and M. Kristan, "Discriminative correlation filter tracker with channel and spatial reliability," *International Journal of Computer Vision*, 2018.

[14] H. Grabner, M. Grabner, and H. Bischof, "Real-time tracking via on-line boosting," in *BMVC*, volume 1, page 6, 2006.

[15] Z. Kalal, K. Mikolajczyk, and J. Matas, "Tracking-learning-detection," *IEEE Trans. Pattern Anal. Mach. Intell*, volume 6, January 2010.

[16] D. S. Bolme, J. R. Beveridge, B. A. Draper, and M. L. Yui, "Visual object tracking using adaptive correlation filters," in *Conference on Computer Vision and Pattern Recognition*, 2010.

[17] M. Danelljan, F. S. Khan, M. Felsberg, and J. Weijer, "Adaptive color attributes for real-time visual tracking," in *Computer Vision and Pattern Recognition*, pages 1090–1097, June 2014.

[18] Y. Wu, J. Lim, M. H. Yang, "Online object tracking: A benchmark sC]," *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 2411–2418, 2013.