ATLANTIS PRESS

Advances in Computer Science Research, volume 80

International Conference on Computer Science, Electronics and Communication Engineering (CSECE 2018)

An Improved TLD Tracking Algorithm for Fast-moving Object

Shijie Zhou, Yuanxi Peng*, Kecheng Gong and Leizhi Shu

College of Computer, National University of Defense Technology, Changsha, Hunan, 410073, China

*Corresponding author

***Abstract*—Traditional object tracking is easily affected by deformation, scale changes, illumination changes, partial occlusions and so on. TLD(Tracking-Learning-Detection) is a classic effective algorithm in long-term tracking which can solve these problems well. Meanwhile, the real-time performance of the system should be taken into account while in the actual situation. An improved fast-moving object tracking algorithm based on TLD is proposed in this paper. In the paper, a method of narrowing the region of detection is proposed to effectively minimize the consumption of time, the method is combined with self-prediction of motion direction to ensure the accuracy of detection. To compensate for the possible missing and false detections caused by the reduction of detection region and the changing background, the variance threshold is updated dynamically to let more possible correct bounding boxes pass the variance classifier. Experiments have been conducted to verify the improved TLD algorithm, the results show that our algorithm ensures the accuracy of object tracking and has a good performance on the real-time.**

Keywords—object tracking; TLD, real-time; narrowing region; variance threshold

I. INTRODUCTION

Visual based object tracking has been one of the core content in the field of computer vision, which is defined as estimating the trajectory of a moving object in an image sequence or a video with fixed or changing background. As the vision-based object tracking technology has been widely used in military and civil fields such as military guidance, visual navigation, automated security monitoring, artificial intelligence and so on, it is also faced with a lot of challenges as a practical matter such as deformation, scale changes, illumination changes, partial occlusions and so on. Especially in terms of real-time performance[1] for fast-moving object, due to the tremendous computing workload, the object can't be tracked timely and effectively. Meanwhile, the precision and robustness in the process of long-term tracking can't be guaranteed all the time. Thus, researchers from all over the world have done a lot of research work and proposed a series of objects tracking algorithms.

Unlike traditional object tracking algorithm, the latest object tracking algorithm tend to adopt an online learning mechanism to improve the tracking effect. The online learning mechanism means that the tracker needs to collect a variety of samples of the object to learn and train constantly, besides, the samples should cover various kinds of deformations, rotations, scales, attitudes and illumination, in other words, the sample selection is quite considerable for the performance of the tracker. Among the long-term tracking algorithms based on online learning and detection for single object, TLD proposed by Zdenek Kalal[2] is an adaptive and reliable tracking technique which combines traditional tracking algorithm and traditional detection algorithm. And it updates the key feature points of tracking module and the object model and relevant parameters constantly by using an improved online learning mechanism[3,4] thus it can enhance the stability and reliability of tracking and basically achieves the goal of long-term tracking. However, there is still a defect when it comes to real-time capability. Because of the enormous computation burden, the running speed is less than ideal. In order to improve the efficiency of the tracking algorithm and track the fast-moving object accurately and timely, an improved algorithm based on TLD is proposed in this paper. On the basis of the original TLD, the paper makes improvements from two aspects. Firstly, we narrow the region of detection by predicting the position of the object which proves to improve real-time performance, and secondly, the reduction of detection area and the changing background will inevitably incur missing and false detections. Thus, we update the variance threshold[5] of variance classifier dynamically so that it suits the actual situation well.

II. ANALYSIS OF TLD ALGORITHM

A. Introduction of TLD Algorithm

In simple terms, TLD algorithm is made up of three modules: tracking module, learning module and detection module. The three capital letters: T, L and D represent the three modules and they are grouped together according to a certain formula so that detection module and tracking module can work in parallel with each other under the supervision of the learning module. The structure of TLD is shown in Figure 1.

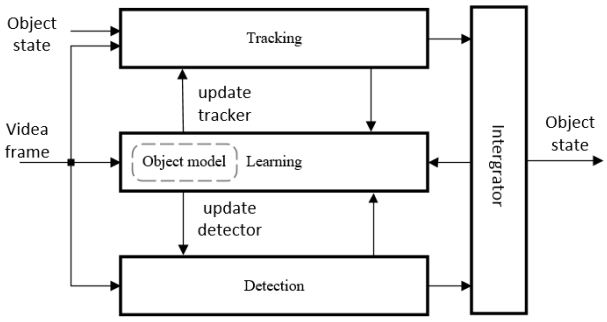

FIGURE I. THE STRUCTURE OF TLD ALGORITHM

Each module plays an indispensable role in the whole process of object tracking. They work like this: the tracker and detector execute in parallel. When the tracker fails to track the target in the previous frame, the tracking result in the current frame should take the result of the detector in the current frame. Otherwise, the tracker executes in the current frame, if the target fails to be tracked, the result of the current frame should also take the result of the detector. But if the target is tracked, we should compare the results of the tracker and the detector. When the overlap of the tracking result and the clustering result of the detector is less than 0.5 and the detector has higher confidence, the result is classified as trusted detections, if there is only one result, as before, the result depends on the detector. We should initialize the tracker with new position relocated by the detector. However, if the overlap of the tracking result and the clustering result of the detector is higher than 0.5 or the tracker has higher confidence, the tracking result is up to both the tracker and the detector. In this case, if the result of the tracker is credible, the learning module starts to work and the online learning model is updated. But we only take the detecting results of detector of which the overlap of the tracker and the detector is higher than 0.7. To get the final tracking result, take the abscissa as an example, a weighted mechanism is adopted here and the formula is as follows.

$$bbnextx = \frac{10*tbbx+\sum_{i=1}^{close\_detections} dbbx_i}{10+close\_detections} \quad (1)$$

Where bbnextx, tbbx and dbbx represent the abscissa of the final result, the tracking result and the satisfied detection result. Similarly, the ordinate, width and height can be computed by the mechanism.

Here are the core principles of each module:

1) Tracking Module

TLD adopts the strategy of overlapping blocks and detection is added to it based on median flow tracker algorithm[6] if tracking fails. Before tracking, the target is marked by the rectangle. In the rectangle, points are generated evenly with an equal interval. And then we estimate the forward and backward motion trajectory of the target between the previous frame and the current frame, the two frames are consecutive. For instance, we select a point A within the rectangle in the previous frame, according to A, a point B is tracked in the current frame. After that, a new point C is tracked in the previous frame according to the point B. Finally, we calculate the displacement from A to B, if the displacement is below a certain threshold, we regard the forward tracking as correct. In this way, we select all the eligible points within the rectangle but only half of them displacements of which are lower than the median are retained as final feature points. Meanwhile, d_i stands for the displacement of the i th feature point while d_m stands for the mean value of all the displacements, $|d_i - d_m|$ stands for the residual, if the residual is higher than 10 pixels, we regard the tracking as a failure. Otherwise, the tracking is successful and we select the minimum rectangle that can contain all the feature points in the current frame as final tracking result.

2) Detection Module

The detection module adopts scanning windows of 21 different scales. Based on the size of target rectangle box, we take 10 incrementing scales and 10 diminishing scales using a fixed scaling factor. The size of window is limited by a minimum of 15*15 pixels and those that don't meet the criteria would be removed. Then each scanning window will scan the whole frame, the step lengths along the x-axis and y-axis of scanning are ten percent of the width and the height of the window. In this way, we get about 50 thousand image patches for each picture. Then all the patches would be put in the cascade classifier which is comprised of Variance classifier, Random Forest classifier and Nearest Neighbor classifier. The structure of cascade classifier is shown in Figure 2.

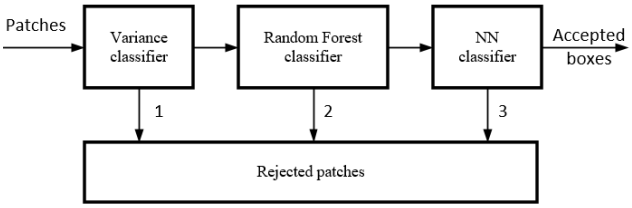

FIGURE II. THE STRUCTURE OF CASCADE CLASSIFIER

After screening by the three classifiers, the patches that fully satisfy the judgment of the three classifiers would be retained as the final result of detection. Otherwise, they would be filtered out. The mechanism of cascade classifier enhances the detection accuracy.

●Variance Classifier

This classifier needs to calculate the variance of gray value of each image patch. The formula of calculating is as follows.

$$D(p) = E(p^2) - E^2(P) \quad (2)$$

Where $D(p)$ is the variance of gray value of image patch p, $E(p)$ is the mean value of gray value of image patch p, $E(p^2)$ is the mean value of gray value in the square of image patch p. The image patch p is considered to contain the target if $D(p)$ is above a certain threshold. Usually we take $\text{var} = 0.5 * D_1$ as the threshold and D_1 is the variance of the initial patch.

●Random Forest Classifier

This classifier is essentially an ensemble classifier consists of 10 base classifiers[7]. Each classifier is like a tree and ten trees make up the forest. Each tree contains 13 nodes and each node performs a comparison of a pair of pixels, besides, each tree's 13 pairs of pixels are different from each other but for each tree, 13 pairs of pixels are fixed and arranged in fixed order so that each tree will generate a 13-bit binary code x for each patch. Then, the posterior probability of $p_i(y|x)$ is tallied, where y$\in$ (0,1) that indicate whether the i th patch contains the target. To ensure the accuracy, we need to calculate the mean value of p_1, p_2, ..., p_9 and p_{10}. If the mean value is over 50%, the patch can be classified as the target or non-target.

●Nearest Neighbor Classifier

This classifier is the final stage of the detector. The process is like online Template Matching Algorithm. The classifier selects the image patch with the highest similarity compared with the original target model as the final test results. Meanwhile,

the online template is updated dynamically and the threshold that is used to train the NN classifier[8] is constantly corrected. The similarity between image patch p_i and p_j is as follows.

$$S(p_i, p_j) = \frac{1}{2}(NCC(p_i, p_j) + 1) \quad (3)$$

Where NCC $\in$[-1,1] is a normalized correlation coefficient.

3) Learning Module

The learning module is based on an online model and the function of the module is to synthesize the results of the tracker and the detector. In addition, it is in charge of the growth and pruning of the online model of the NN classifier and the updating of the posterior probability of $p_i(y|x)$. In other words, learning module reorganizes positive and negative samples to retrain the random forest classifier and the NN classifier.

B. Existing Defects and Possible Solutions

As mentioned in the previous section, for each frame, about 50 thousand image patches would be generated to be judged by the variance classifier. However most of them don't contain the target. Thus, it will take much time unnecessarily and lead to low efficiency of the algorithm. When tracking a fast-moving target, the processing speed of the algorithm is particularly important. To improve real-time performance, a concept of ROI is proposed in [9] which means that detection is only executed within the region of interest. For instance, [10] use Calman filter to predict the next region of detection, [11] adopts Pyramid LK optical flow method and [12] adds Meanshift algorithm to TLD to get the region of interest. All the method above can accelerate the process of the tracker.

III. IMPROVED TLD ALGORITHM

According to the above analysis, the processing time of each frame is greatly reduced. However, due to some potential defects and problems of the method of predicting the next region of detection, some correct samples will be inevitably missed. With the accumulation of error, the online model will be more and more inaccurate and the tracking accuracy will be significantly reduced which will ultimately lead to the failure of tracking. For the purpose of giving attention to both the real-time performance and the precision of tracking, we proposed an improved algorithm based on TLD.

A. Narrowing the Region of Detection

The main purpose is to reduce the number of image patches waiting to be detected so that the computation is eased and the running speed can be raised obviously. The specific method is as follows.

(1) According to the target box in the last frame, the tracker predicts a new box in the current frame.

(2) Compare the positions of the two boxes and we will get four possible directions of motion: the top left, the top right, the lower left and the lower right. Here we take the top left as an example and continue with the following steps.

(3) Based on the lower right corner of the target box, we will get a point after a displacement. The displacement vector is (X, Y) where the value of X is half of the value of the width of the target box and the value of Y is half of the value of the height of the target box.

(4) Via (3), we get a point. Based on the point, we take the whole upper left area as the final region of interest. All patches that overlap with the region should be send to the detector.

It should be noted that not every frame is processed in this way, if the tracking of the last frame is a failure, then we should detect the whole image. The diagram of narrowing the region of detection is shown in Figure 3.

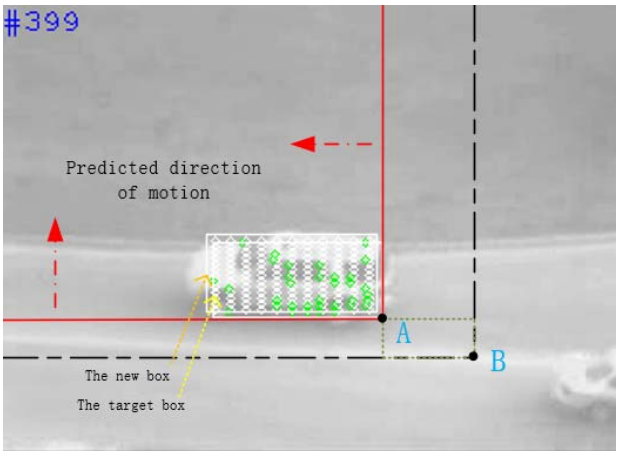

FIGURE III. THE DIAGRAM OF NARROWING THE REGION OF DETECTION

Most of the time, the results of the tracking module are credible if the target is not obscured and therefore the region near the location of the box predicted by the tracker is likely to contain the target in the current frame. As is shown in Figure 3, the region at the top left of point B is finally what we want.

B. Updating the Variance Threshold Dynamically

In the detection module, the variance classifier is the first to screen all the patches and it's particularly important for the accuracy. We already know from the previous text that the variance threshold is a fixed value which equals half of the initial target variance. Only the image patches of which the variance is greater than the threshold can pass through the classifier, and thus it's rather important to select a suitable threshold.

With the increase of video sequences, the shape and size of the object will change inevitably and the image information would be affected by the similar color, occlusion, light and other factors. As a result, the variance of the target will change more or less. If we still use the original fixed threshold, the correct patches will probably be filtered out and the subsequent tracking will fail.

To solve this problem, we adopt a mechanism of updating the variance dynamically. Under this mechanism, we give full consideration to the information of correct patches of all processed frames. With the increase of video frames, the new characteristics of the target tracked are obtained continuously and the variances of every target tracked are stored which can reflect the changes. D_i is the variance of the i th target tracked.

Each time the variance classifier executes, we take the minimum value of the existing top N variances as D_{min} and the threshold $\text{var} = 0.5 * D_{min}$. In this way, any possible correct patch won't be missed.

IV. EXPERIMENTAL RESULTS

Our experimental platform is Ubuntu14.04 and OpenCV 2.4.9. The processor is Intel(R) Core(TM) i5-4590 CPU @3.30GHz with 8GB RAM. The original version of TLD is C++ version by Alan Torres.

This paper selects different datasets from a benchmark and evaluates the proposed improved TLD by comparing it with the original TLD.

A. Datasets Introduction

The video sequences for testing the algorithm are from a benchmark. We compare the improved TLD with the original TLD under different scenes. There are ten open testing video sequences, the effect of tracking is as follows.

FIGURE IV. THE EFFECT OF TRACKING

The video sequences are named as David, Jumping, Pedestran1, Pedestran2, Pedestran3, Car, Motocross, Volkswagen, Carchase and Panda. Among them, Jumping, Car, Motocross, Volkswagen and Carchase reflect a situation of fast motion.

B. Results and Analysis

In theory, the cost of time is greatly reduced benefited from the decreasing numbers of the image patches of most of the frames. Take the Car as an example, the comparison of the numbers of image patches of the two algorithm is shown in Figure 5.

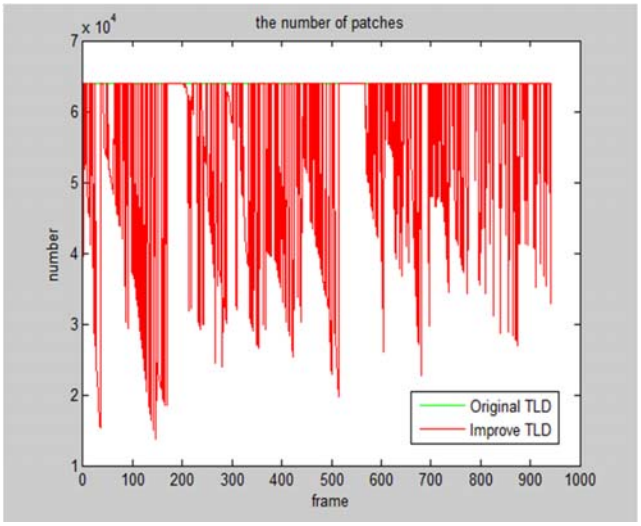

FIGURE V. THE COMPARISON OF ORIGINAL TLD AND IMPROVED TLD

From Figure 5, we know that for original TLD, all of the image patches generated by the sliding windows will be sent to the detection module. As a result, the detecting time for each frame is more than that of improved TLD.

The comparison of the time of detection each frame between the original TLD algorithm and the improved TLD algorithm is shown in Figure 6.

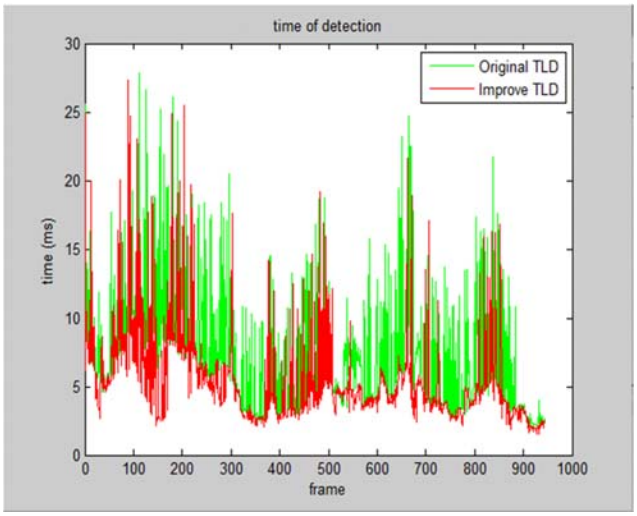

FIGURE VI. THE COMPARISON OF DETECTING TIME

We can know from the above figure that the detecting time for each frame of our improved TLD algorithm is obviously shorter than that of the original TLD. Though the time saved is negligible, with the accumulation of them, the final improvement is significant.

Next, let's have a look at the overall running time for each video sequence. Here is a comparison of the original TLD and the improved TLD shown in Table I.

TABLE I. THE OVERALL RUNNING TIMES

Name	Frames	Running time of original TLD/ms	Running time of improved TLD/ms
1.david	761	49316.7	46601
2.jumping	313	20284.8	19167
3.pedestrian1	140	11620.8	10147
4.pedestrian2	338	17257	17201.6
5.pedestrian3	184	15273.2	12433.5
6.car	945	57808.8	51669.5
7.motocross	2665	166372	147924
8.volkswagen	8576	660542	640314
9.carchase	9928	665135	563881
10.panda	3000	184243	179393

From the table, we can draw a conclusion that the improved algorithm can improve the real-time performance of tracking.

But a good performance in real time does not mean that our algorithm is better, we should also take its accuracy into consideration. Here we select Jumping, Car, Motocross, Volkswagen and Carchase which reflect the situation of fast motion to test our algorithm. Here we adopt the evaluation of Precision, Recall and F-measure. The result of the evaluation is shown in Table II.

TABLE II. THE RESULT OF THE EVALUATION

Name	Original TLD			Improved TLD		
	P	R	F	P	R	F
Jumping	1.000	0.613	0.760	1.000	0.732	0.845
Car	0.962	0.982	0.972	0.978	0.990	0.984
Motocross	0.704	0.470	0.564	0.935	0.493	0.646
Volkswagen	0.865	0.245	0.381	0.916	0.365	0.513
Carchase	0.981	0.147	0.263	0.997	0.159	0.274

As the is shown in the Table II, the precision, recall and f-measure are improved. A P-value indicates the proportion of tracked precise bounding boxes to tracked bounding boxes, an R-value indicates the proportion of tracked precise bounding boxes to all precise bounding boxes, and a F-value is the harmonic mean of P and R.

V. CONCLUSION

As the real-time performance of traditional tracking for fast-moving object is not ideal enough. This paper has presented an improved object tracking algorithm based on TLD to achieve long-term target tracking with ideal real-time performance. Based on TLD, we have introduced a new mechanism of generating the region of interest. The experimental results show that the proposed algorithm has faster running speed and higher tracking accuracy for fast-moving object.

The main direction of future work is to further speed up the operation of the system. Some modules such as the time-consuming NN classifier have much room for improvement.

REFERENCES

[1] H. Grabner, M. Grabner, and H. Bischof, "Real-time tracking via on-line boosting," *Br. Mach. Vis. Conf.*, vol. 1, pp. 47–56, 2006.

[2] Z. Kalal, K. Mikolajczyk, and J. Matas, "Tracking-Learning-Detection," vol. 34, no. 7, pp. 1409–1422, 2012.

[3] Z. Kalal, J. Matas, and K. Mikolajczyk, "Online learning of robust object detectors during unstable tracking," *2009 IEEE 12th Int. Conf. Comput. Vis. Work. ICCV Work. 2009*, pp. 1417–1424, 2009.

[4] Z. Kalal, J. Matas, and K. Mikolajczyk, "P-N learning: Bootstrapping binary classifiers by structural constraints," *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, pp. 49–56, 2010.

[5] T. Xu, C. Huang, Q. He, G. Guan, and Y. Zhang, "An improved TLD target tracking algorithm," *2016 IEEE Int. Conf. Inf. Autom. IEEE ICIA 2016*, no. August, pp. 2051–2055, 2017.

[6] Z. Kalal, K. Mikolajczyk, and J. Matas, "Forward-backward error: Automatic detection of tracking failures," *Proc. - Int. Conf. Pattern Recognit.*, pp. 2756–2759, 2010.

[7] B. L, "Random forest," *Mach. Learn.*, vol. 45, no. 1, pp. 5–32, 2001.

[8] Y. Liao and V. R. Vemuri, "Use of k-nearest neighbor classifier for intrusion detection," *Comput. Secur.*, vol. 21, no. 5, pp. 439–448, 2002.

[9] B. Nemade and V. A. Bharadi, "Adaptive automatic tracking, learning and detection of any real time object in the video stream," *Proc. 5th Int. Conf. Conflu. 2014 Next Gener. Inf. Technol. Summit*, pp. 569–575, 2014.

[10] C. Sun, S. Zhu, and J. Liu, "Fusing Kalman filter with TLD algorithm for target tracking," *Chinese Control Conf. CCC*, vol. 2015–Septe, pp. 3736–3741, 2015.

[11] B. Ahn, Y. Han, and I. S. Kweon, "Real-time facial landmarks tracking using active shape model and LK optical flow," *2012 9th Int. Conf. Ubiquitous Robot. Ambient Intell. URAI 2012*, no. Urai, pp. 541–543, 2012.

[12] N. Fan, N. Huang, F. Yu, Y. Song, S. Zhao, and Z. Tan, "An Improved Target Tracking Scheme via Integrating Mean-shift with TLD Algorithm," pp. 7877–7882, 2017.