An Improved TLD Tracking Method Using Compressive Sensing

Qiang Li[1], Xueshi Ge[2] and Geng Wang[1,*]
[1]School of Software, Shanghai Jiao Tong University, Shanghai, China
[2]Naval Academy of Armament, Shanghai, China
[*]Corresponding author

***Abstract*—Visual Tracking, as an important subject in computer vision, has been widely used in surveillance, space exploration, and human-computer interaction etc. Both tracking-learning-detection (TLD) [1] and compressive tracking (CT) [2] are successful algorithms among those proposed recently. However, TLD suffers from low efficiency and CT overlooks scale change during tracking. In this paper, we propose an improved TLD tracking algorithm by using compressive sensing. The improvements include enhancing the detection method in TLD with CT, employing Kalman filter in detector to estimate the tracking region for improving the detection speed. Besides, adaptive search radius is employed to deal with object disappearance and shielding issue. Lastly, the tracking results of TLD and CT are integrated to estimate the target status and update the classifier. The experiments show that, compared to the original algorithms, the improved algorithm combines the advantages of two algorithms, conducing to accurate tracking precision, faster tracking speed and handling the object extent change.**

Keywords—visual tracking; tracking-learning-detection (TLD); compressive tracking (CT); tracking speed; extent change

I. INTRODUCTION

Object tracking is a classic problem in computer vision, and has wide application prospects in space exploration, intelligent surveillance, unmanned robotics and military usage. Although there have been a number of algorithms proposed, it remains a challenging problem because of the undetermined factors such as pose variation, illumination change, drift, shielding, occlusion and so on. In addition to solving the above problems, a successful tracking algorithm should run in real time, thus has a high requirement on the efficiency of the algorithm.

Object tracking can be achieved from two perspectives, by tracking or by detection. The principle of tracking algorithm is estimating object motion. Trackers only need initialization and will produce smooth trajectories fast, but will accumulate errors (drift) during tracking process and can't handle the case when object disappear from view. On the other hand, detection based algorithms locate the target in every frame independently, but need samples of target to train the classifier.

An effective appearance model is the key for a tracking algorithm, which can be generally categorized as either generative [3, 4] or discriminative [5, 6]. Generative tracking algorithms generally learn a model to represent the target and search the most similar region with the model in next frame. The generative model can be trained offline or update online to adapt to the appearance change during tracking. Generative tracking needs numerous samples to train the model. However, there are typically not enough samples at the outset and it's likely to lose target if appearance changes a lot at the beginning. Secondly, generative tracking algorithms don't use the background information to improve tracking stability and accuracy. Discriminative tracking algorithms take tracking problem as a binary classification to find the decision boundary for separating the target object from the background. Static discriminative trackers train an object classifier before tracking, which can be only used to track know objects. Adaptive discriminative trackers update the classifier during tracking: close neighbors are sampled to train positive examples and distant surroundings are sampled to train negative examples.

Object detection's task is to locate the object in an input frame. Detection typically relies on local image features or a sliding window. Feature-based algorithms usually have three stages: feature detection, feature recognition and model fitting, which needs to know the geometry of the object in advance. Sliding window based algorithms scan the input image by various window size step to detect whether the patch contains the object. There are so many windows in each image, so to achieve high efficiency, a cascaded architecture is employed.

In this paper, an improved TLD tracking algorithm has been proposed, enhancing the detector using compressive sensing. The block diagram is shown in Figure 1. The original object detector, which uses scanning-window grid and cascaded classifier, is replaced with Kalman filter and compressive tracking to improve the detection speed and accuracy. For every input frame, tracker uses median-flow tracking algorithm based on Lucas–Kanade optical flow to estimate the location and extent of the object. Detector uses Kalman filter to estimated position of the target based on historical information, and then does a multi-scale image sampling nearby the estimated position. Based on compressive sensing theory, the feature dimensionality has been dramatically reduced by a sparse measurement matrix. Then the samples will be discriminated by a trained classifier to find out whether the object is in the patches. Integrator will gather the information from tracker and detector to get the final decision on the target position and scale. Learning part uses P-N learning methods, updates the target's positive samples and negative samples.

The remaining part of this paper is organized as follows. Section 2 gives a brief introduction to the related work of TLD and compressive tracking. Section 3 proposes the improved tracking algorithm in detail. Experimental results are reported

in section 4. Finally, a conclusion of this paper is made in section 5.

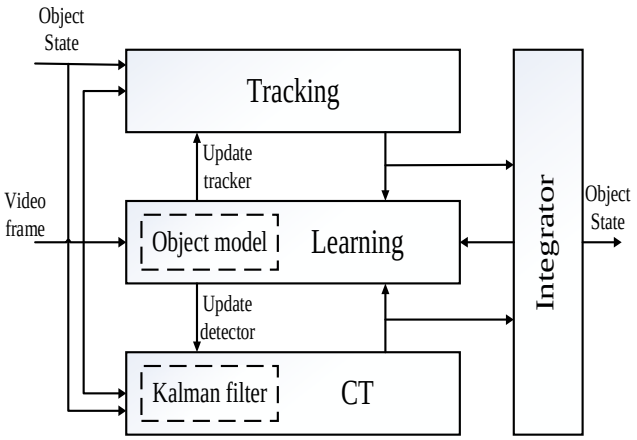


FIGURE I. BLOCK DIAGRAM OF IMPROVED TLD ALGORITHM

II. RELATED WORK

A. TLD Tracking Algorithm

TLD tracking algorithm is a semi-supervised learning framework, which divides tracking task into three parts: tracking, learning and detection, as shown in Figure 2. These three parts are independent subroutines, and can process the input frame simultaneously. Tracker locates the target by estimating the motion frame to frame. Detector captures all traced target and use the samples to correct trackers as much as possible. Learner uses the results from tracker and detector to get knowledge to the target feature, and forms the library of positive and negative samples, which can update detectors to avoid the errors occurring again.

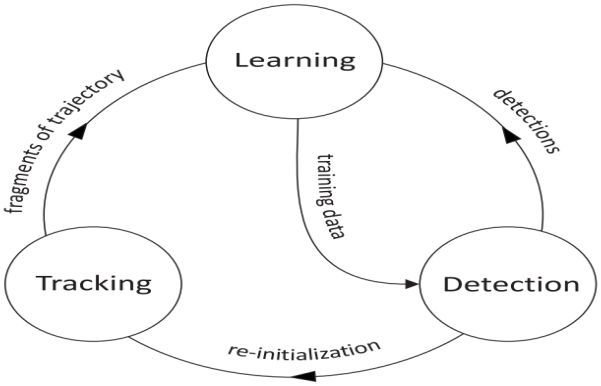


FIGURE II. THE BLOCK DIAGRAM OF TLD FRAMEWORK [1]

During the process of detecting the target in every frame, detector firstly creates index for the entire input frame and divides the input frame into a number of scanning-window grids. Then, for each grid, detector will check it with the all the patches in samples library. Lastly, the grid with most possibility that contains target will be selected as the output of detectors. However, most of the grids checked don't contain the target or contain small part of the object, so these grids are all negative samples and consume a lot of processing time reducing the efficiency of TLD algorithm.

B. Compressive Tracking Algorithm

Compressive tracking algorithm is based on compressive sensing theory. As the general pattern of classification schema, compressive tracking firstly extracts the multi-scale image feature, and then discriminate patch through the trained Bayes classifier. Lastly, classifier is updated through online learning strategy. The main components of compressive tracking algorithm are shown in Figure 3.

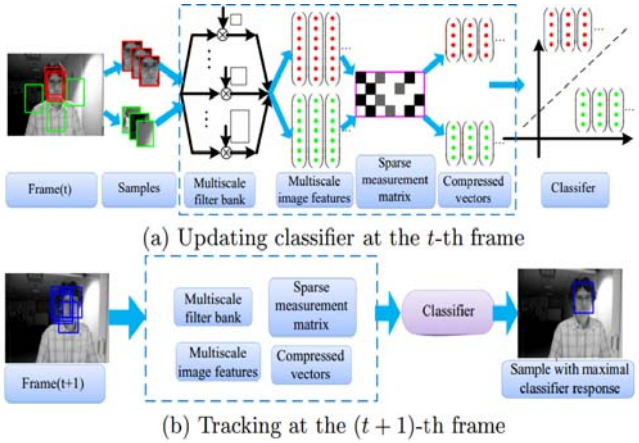


FIGURE III. MAIN COMPONENT OF COMPRESSIVE TRACKING ALGORITHM [2]

A random matrix $R \in \mathbb{R}^{n \times m}$ whose rows have unit length is used to project data from high-dimensional image space $x \in \mathbb{R}^m$ to a lower-dimensional space $v \in \mathbb{R}^n$ where $n \ll m$.

$$v = Rx \tag{1}$$

Ideally, R is expected to provide a stable embedding that approximately preserves all the distance between all pairs of original signals. The Johnson-Lindenstrauss lemma [2] points that the distance between the points in a vector are very likely to be preserved if they are projected onto a randomly selected subspace with suitably high dimensions. Baraniuk et al. proved that the random matrix satisfying the Johnson-Lindenstrauss lemma also holds true for the restricted isometry property (RIP) in compressive sensing. As a result, if the random matrix R in (1) satisfies the Johnson-Lindenstrauss lemma, x can be reconstructed with minimum error from v with minimum error.

Even though compressive tracking algorithm has great advantages in tracking speed, samples are extracted within a predefined search radius, which has great effect on the real-time performance. Moreover, without consideration of motion information, tracking algorithm are very likely to suffer from drift problems. Last but not least, scale change of the target is overlooked when its distance between target and camera or focal length of camera changes. Apparently, when the target is moving away from the camera, more background will be contained in the fixed size window, which would contribute to the unreliability to the classifier.

III. PROPOSED ALGORITHM

In this section, the improved TLD tracking algorithm is proposed. The original detection method in TLD tracking algorithm is replaced by adaptive compressive tracking to improve the detection rate. The process of proposed algorithm is show in Figure 4. At every frame, tracker will estimate the target motion to get its location and scale. Detector will use the scale information to do sampling adaptively nearby the region predicted by Kalman filter. A training process is added to compressive tracking to update the parameters of classifier. Instead of getting only one output from compressive tracking, a set of possible target status will be generated from detector. Integrator will analyze the results from tracker and detector to get the final output. Learning part uses the result to update positive and negative samples.

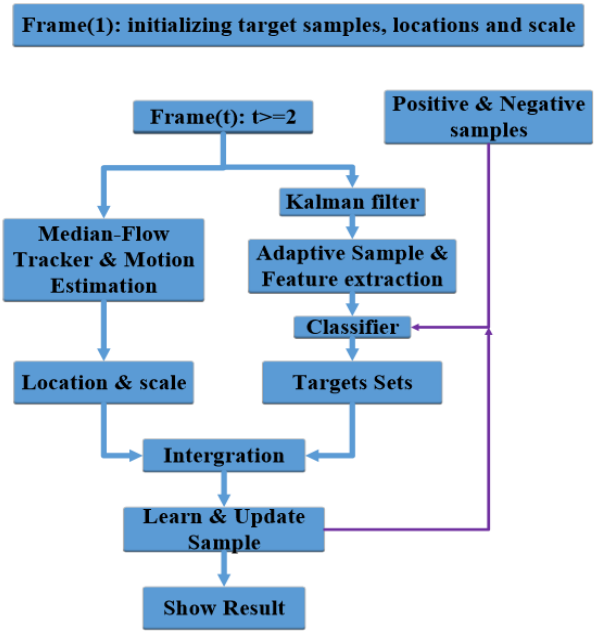


FIGURE IV. PROCESS OF IMPROVED TLD ALGORITHM

A. Median Flow Tracker

In two consecutive frames, a number of feature points, called *pts1*, are sampled uniformly within the target region in *t-1* frame. At *t* frame, the tracker produces a trajectory by tracking these points using Lucas-Kanade optical flow algorithm. Then backward tracking is carried out with the same method from *t* frame to *t-1* frame, getting the same number of points in *t-1* frame, called *pts1'*. Ideally, *pts1* and *pts1'* are identical points if tracker works perfectly. So the difference of two set of points is compared to validate the correctness of tracking. Tracker will filter the invalid tracking points and use the median position and scale difference as the tracking box, as shown in Figure 5.

B. Adaptive Compressive Tracking

Based on compressive sensing, all the elements in low-dimensional representation $v = (\upsilon_1, ..., \upsilon_n)$ are independently distributed and are almost always Gaussian. So for the Bayes classifier.

$$H(\mathrm{v}) = \sum_{i=1}^{n} \log\left(\frac{p(v_i \mid \mathrm{y}=1)}{p(v_i \mid \mathrm{y}=0)}\right) \tag{2}$$

The original compressive tracking, for every input frame, samples a set of image patches within a predefined radius nearby the last tracking location, and use the classifier to get the one with maximal classifier response. Our algorithm improves the process of classifier. In the first frame, the positive samples and negative samples are used to train the classifier getting a threshold of the classifier. As a result, instead of outputting only one result, detector will generate a set of possible positive image patches. Integrator will use these patches from detector and tracker to get the final output. Learning part synthesizes the information from integrator to update the classifier with PN learning method.

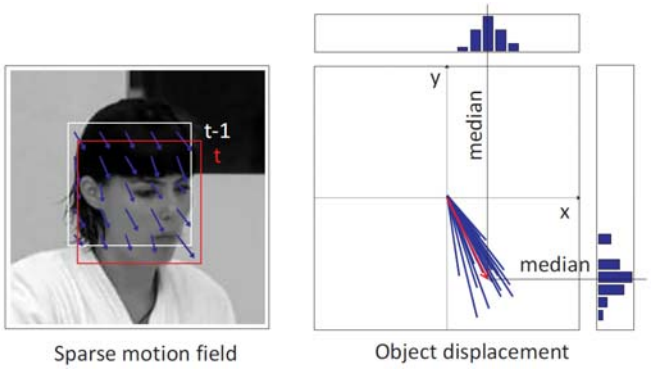


FIGURE V. PRINCIPLE OF MEDIAN FLOW [1]

C. Integration

The results from tracker and detector should be combined to generate the final result. Integrator use the following strategy to integrate the output from tracker and detector.

1) If tracker fails to generate any tracking result, output the detecting box with maximal classifier score.

2) Cluster the box from detector.

3) Find those detecting box that far away from the tracking box and has greater similarity than tracking box. If only one is found, output it as the result.

4) If more than one found, find those detecting box nearby the tracking box, and average them as output result.

IV. EXPERIMENTS

The tracking results of our proposed algorithm are shown in this section. We choose the two basis of our improved algorithm, TLD and compressive tracking, as comparisons. The experiments are performed on 5 public video sequences from http://www.visual-tracking.net, which have the challenging aspects, such as illumination variation, occlusion, scale variation, motion blur and so on. Our tracker is implemented

with C++ and opencv, which runs at 22 frames per second (FPS) one a Core Dura-Core 2.50GHz CPU with 4GB RAM.

Two metrics are used to evaluated the proposed algorithm. The first metric is success rate by evaluating the overlapping ratio between the tracking result and ground truth.

$$score = \frac{area(B_t \cap B_{gt})}{area(B_t \cup B_{gt})} \quad (3)$$

where Bt is the tracking bounding box and Bgt is the ground truth bounding box. If the score is greater than 0.5 in a frame, the tracking result is considered as a success. The other metric is the center location error (CLE) which describes the pixel distance between the tracking result and ground truth. Table 1 and Table 2 propose the quantitative results of the comparison among the three algorithms based on the two metrics.

TABLE I. SUCCESS RATE (SR)(%) OF COMPARISON ALGORITHMS

Sequence	Frames	CT	TLD	TLD with CT
Box	1161	89	92	**92**
David	502	90	98	**99**
Face Occlusion 2	812	100	99	**100**
Girl	452	78	67	**80**
Sylvester	1344	75	94	**95**
Tiger 1	354	80	75	**82**

TABLE II. CENTER ERROR LOCATION(CLE) (PIXELS) OF COMPARIOSN ALGORITHMS

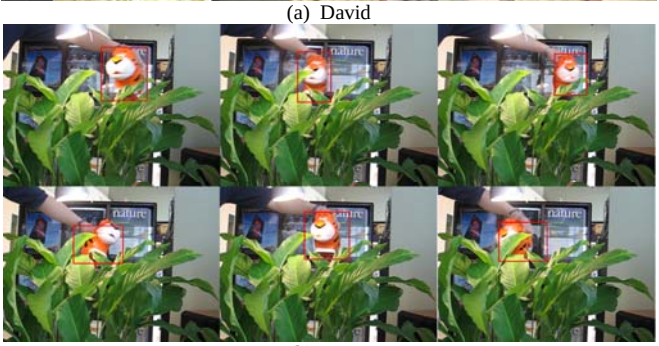
Sequence	Frames	CT	TLD	TLD with CT
Box	1161	14	17	**14**
David	502	16	10	**9**
Face Occlusion 2	812	10	13	**10**
Girl	452	21	18	**17**
Sylvester	1344	9	7	**6**
Tiger 1	354	10	8	**7**

TABLE III. OVERALL PERFORMANCE OF COMPARISON ALGORITHMS

	CT	TLD	TLD with CT
Average CLE	12.4	12.0	10.2
Average FPS	34	12	22

As is shown in Table 1 and Table 2, our algorithm, combining the advantages of TLD and CT, makes up the shortcomings of the two basis algorithms and improves the performance to a certain degree. Besides, our algorithm doesn't sacrifice a lot of efficiency to earn the performance. Table 3 describes overall performance of the three algorithm. Our algorithm greatly improves the FPS of TLD, and can handle the requirement of real-time tracking. Figure 6 gives some snapshots of successful tracking of our proposed algorithm.

(a) David



(b) Tiger

FIGURE VI. SNAPSHOTS OF SOME SAMPLED TRACKING RESULTS

V. CONCLUSION

In this paper, an improved TLD tracking algorithm is proposed by using compressive sensing detection. TLD suffers from shortcoming of inefficiency and CT can't handle the scale change during tracking. By combining the advantages of these two algorithms, employing Kalman filter to estimate the search region and enhancing CT in learning framework, our algorithm makes up the disadvantages of the two algorithms and improves the performance. The experiments show that among the three algorithms, our proposed algorithm has more accurate tracking precision, faster tracking speed and can handle the challenging aspects during tracking.

REFERENCES

[1] Kalal, Zdenek, Krystian Mikolajczyk, and Jiri Matas. "Tracking-learning-detection." Pattern Analysis and Machine Intelligence, IEEE Transactions on 34.7 (2012): 1409-1422.

[2] Zhang, Kaihua, Lei Zhang, and Ming-Hsuan Yang. "Real-time compressive tracking." Computer Vision–ECCV 2012. Springer Berlin Heidelberg, 2012. 864-877.

[3] Ross, David A., et al. "Incremental learning for robust visual tracking." International Journal of Computer Vision 77.1-3 (2008): 125-141.

[4] Li, Hanxi, Chunhua Shen, and Qinfeng Shi. "Real-time visual tracking using compressive sensing." Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on. IEEE, 2011.

[5] Avidan, Shai. "Support vector tracking." Pattern Analysis and Machine Intelligence, IEEE Transactions on 26.8 (2004): 1064-1072.

[6] Grabner, Helmut, Michael Grabner, and Horst Bischof. "Real-Time Tracking via On-line Boosting." BMVC. Vol. 1. No. 5. 2006.