ISSN (Print) : 0974-6846
**Indian Journal of Science and Technology,** *Vol 8(26), DOI: 10.17485/ijst/2015/v8i26/80983, October 2015* ISSN (Online) : 0974-5645

# An Enhanced Model-based Tracking Algorithm with Dynamic Adjustment of Learning Parameters according to Online Performance Evaluation

**Yehong Chen[1], Pil Seong Park[2*] and Qian Gao[1]**

[1]School of Information, Qilu University of Technology, Jinan - 250353, China;
chenyh@qlu.edu.cn, gq@qlu.edu.cn
[2]Department of Computer Science, University of Suwon, Hwasung, Gyeonggi-do - 445743, Korea;
pspark@suwon.ac.kr

## Abstract

In online object tracking, the ability to identify possible track loss without ground truths is important. Template Inverse Matching (TIM) is one such performance evaluation method that does not require ground truths, but it is only applicable to template-based methods. We extend the idea of TIM to a more general method so that it is applicable to model-based trackers. The idea can be introduce to any model-based trackers to adjust model learning parameters whenever possible track loss is identified by this technique. As an application, we introduced the idea to the famous Real-time Compressive Tracking (CT for short) algorithm and the results are compared with that of the original CT. The experimental results show better performance and stability with negligibly small additional running time.

**Keywords:** Model Inverse Matching, Object Tracking, Online Performance Evaluation, Real-Time Compressive Tracking, Template Inverse Matching

## 1. Introduction

In the past decades, researchers gained a lot of progress in real-time object tracking, and the result is the advent of many excellent algorithms like Particle filter[1], HOG plus SVM tracker[2], online AdaBoost tracker[3], L1-tracker[4], and MIL tracker[5], etc. However, none of them perfectly works in all conditions due to complex situations, e.g., cluttered background, illumination changes and all kinds of occlusions. Most of state-of-the-art tracking algorithms can give satisfactory tracking results in certain particular scenes but give poor results in other real world situations. Even though some observation has been reported on which algorithm is suitable for which kind of videos, but there is still no general rule.

To improve stability, some strategies to evaluate the performance of a tracking algorithm are crucial. Many types of metrics which are dependent on ground truth data have been proposed in order to compare tracking algorithms by their results. Chau et al.[6] used a set of features to compute the confidence of trajectories and also the precision of tracking results. Wu and Zheng[7] proposed an algorithm based on measuring appearance similarity and tracking uncertainty

Whenever a tracker runs on a video sequence in a real-world tracking task, system parameters are set offline in advance hoping good performance. However, as the situation in the video changes, it is not guaranteed to work well until the end, and tracking sometimes ends up with a drift. There are two main reasons for it - One is because of inaccuracy of the current tracking algorithm, i.e., the accumulated error in every frame will finally result in a drift. The other is due to the fixed parameters used in the tracking algorithm. In fact, most running parameters are sensitive to different video characteristics. When the situation in a video changes, the parameters which were optimally chosen in some situation may not work anymore in some other situations. Hence, the running

* *Author for correspondence*

parameters should be adjustable according to situation changes, especially for videos showing dynamic changes.

Online determination functionality of track loss can help improve performance by making them adjust or reset themselves automatically, and it has received more and more attention in computer vision community. SanMiguel et al.[8] presented a comparative evaluation method of online quality estimators for visual object tracking, showing that the observation likelihood measure is appropriate for overall tracking performance evaluation, while TIM measure[9] is appropriate to detect the start and end instants of tracking failures. We are especially interested in TIM, which is the inverse of the common process of Template Matching (TM), and is theoretically proved by Stable Marriage (SM) problem theory. TIM is algorithm-related rather than trajectory character related. Hence, it is helpful to link algorithm parameters with evaluation measurements.

In this paper, we first look at TIM and the famous state-of-art algorithm CT[10], which we will improve by adopting our parameter adjustment strategy based on online performance evaluation to detect possible track loss. However, we cannot use TIM as is since CT is not template-based. Similarly to TIM, we devise a more general Model Inverse Matching (MIM) idea and propose our MIM-aided CT algorithm. Experimental results comparing the performance of the two algorithms are given at the end.

## 2. Related Works

### 2.1 Template Inverse Matching

Ground truth based performance evaluation methods are not appropriate for online tracking. Liu et al.[10] introduced a performance evaluation method called TIM. Under template-based tracking context, to position a target, a common way is finding the region that matches the current template as closely as possible, i.e., by template matching. As an online performance evaluation method, TIM is the inverse process of TM. After capturing the target in the new frame, TIM uses it as the template to match the target in the previous frame. The resulting location by TIM is then compared with the known previous target position in the previous frame. The difference between these two positions can be considered as a measure of tracking accuracy.

Let $I_t$ be the $t$-th frame in a video sequence, and $Z_t$ be the target captured in $I_n$ for some $t$. The TM process can be described as

$$Z_t = T(I_t, Z_{t-1}) \tag{1}$$

Where $T(\,)$ is some tracking algorithm which uses $Z_{t-1}$ as a template and looks for a matched target in the image $I_t$. On the other hand, TIM is the inverse process of TM:

$$Z'_{t-1} = T(I_{t-1}, Z_t) \tag{2}$$

Where $Z'_{t-1}$ is the result in the previous frame by TIM. Note that the same method $T(\,)$ is used in both TM and TIM. Track loss is determined by the distance measure $d_t$ between the centers of $Z_{t-1}$ and $Z'_{t-1}$

$$d_t = \sqrt{\left(\frac{C_x(Z_{t-1}) - C_x(Z'_{t-1})}{W(Z_{t-1})}\right)^2 + \left(\frac{C_y(Z_{t-1}) - C_y(Z'_{t-1})}{H(Z_{t-1})}\right)^2} \tag{3}$$

where $[C_x(),C_y()]$ is the center coordinate, and $H()$ and $W()$ are respectively the height and width of the target.

If $d_t$ is larger than some pre-defined threshold $d_T$, it means the tracking has been disturbed and a possible drift might have taken place.

### 2.2 Real-Time Compressive Tracking

Zhang et al.[10] proposed a highly efficient state-of-the-art real-time compressive tracking algorithm. The object representation used in CT is a compressed feature vector in a generalized Haar-like feature space. For each image sample $I$, its low-dimensional representation is $\mathbf{v} = (v_1,\ldots,v_m)$, where $m = 50$ is used in CT. CT assumes all elements in $\mathbf{v}$ are independently distributed and models them with a Naive Bayes (NB) classifier

$$F(v) = \sum_{i=1}^{n} \log\left(\frac{p(v_i \mid y = 1)}{p(v_i \mid y = 0)}\right) \tag{4}$$

Conditional distribution is assumed to be Gaussian with four parameters $p^t = (\mu^1, \sigma^1, \mu^0, \sigma^0)$, where $\mu^1$, $\sigma^1$ or $\mu^0$, $\sigma^0$ respectively are the mean and standard deviation of the positive sample set $X^1$ or the negative sample set $X^0$.

At the $t$-th frame, the newly learned model parameter $p^t$ is incrementally updated as a combination of these Gaussian parameters by the weighted average of the current and incremental values by

$$\sigma_t^1 \leftarrow \sqrt{\lambda(\sigma_{t-1}^1)^2 + (1-\lambda)(\sigma^1)^2 + \lambda(1-\lambda)(\mu_{t-1}^1 - \mu^1)^2} \tag{5}$$

$$\mu_t^1 \leftarrow \lambda\mu_{t-1}^1 + (1-\lambda)\mu^1 \tag{6}$$

where $\lambda$ is the learning parameter (0.85 in CT).

Let the previously captured target sample be $z^{t-1} = (x,y,w,h)$, where $x^{t-1} = (x,y)$ is its coordinate, and $w$ and $h$ are its width and height. Let $V^t = \{v_1^t, ..., v_j^t : j = 1, ..., n\}$ e the representation of the current detector samples which are centered at $x^{t-1}$ within a certain radius. Then the captured target will be chosen as the one with a maximal value of the classifier function $F(\ )$, i.e.,

$$J = \arg\max_j F(v_j^t) \qquad j = 1, 2, .....n.. \tag{7}$$

## 3. Our Proposed Algorithm

When running the CT algorithm on a video, some parameters need to be carefully tuned beforehand. Some of them are the learning parameter $\lambda$, the search window size, the number of features to be extracted, etc. We focus on just one parameter $\lambda$, keeping the rest fixed throughout. We'd like to improve the performance of the famous CT algorithm by introducing an additional mechanism to adjust the parameter $\lambda$ whenever possible track loss is detected. From now, we will use $\lambda_t$ instead of $\lambda$ since we do not use a fixed value for it.

### 3.1 Detecting Possible Track Loss

To detect possible track loss, we use a similar idea to TIM. However, we cannot introduce TIM to CT as is since CT is not template-based but model-based[9,10]. Similarly to TIM, we introduce model inverse matching as follows.

Let the Model Matching (MM) process in CT be represented by

$$Z_t = M(I_t, Z_{t-1}) \tag{8}$$

where $I_t$, $Z_t$, and $Z_{t-1}$ are as explained in the previous section. $M(\ )$ is the tracking process that consists of two stages, i.e., a learning stage and a detection stage. In CT, most of computation time is spent during the learning stage, and the detection stage takes negligibly small amount of time compared with the learning stage. We define the MIM process by

$$Z'_{t-1} = N(I_{t-1}, Z_t) \tag{9}$$

where $N(\ )$ is similar to $M(\ )$ but takes much less time by adopting only the detection stage.

We would like to improve the performance of the original CT by embedding the MIM process at the end of the detection step in CT, as a way to identify some possible track loss during tracking. That is, at every frame, after capturing the new position of the target, the algorithm triggers the MIM process to inversely estimate the previous position in the previous frame, as shown in Figure 1.

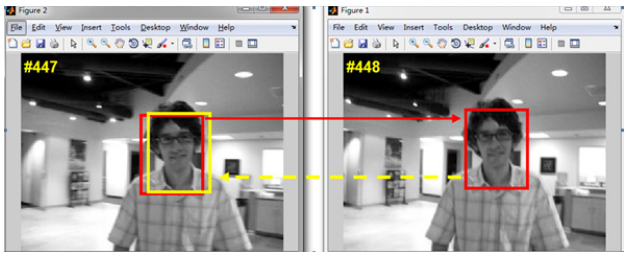


**Figure 1.** An example showing inverse matching by MIM.

The red rectangles are the tracking results by the original CT in two consecutive frames #447 and #448. The yellow one (in frame #447) is the inversely estimated result by MIM using the new position (in frame #448). The difference in center coordinates of the two rectangles in the previous frame #447 is estimated by the measure $d_t$ in (3). If $d_t$ is larger than some threshold $d_T$, then we may conclude that the newly estimated position (in frame #448) by the MM process may not be correct and some track loss might have occurred. Then we discard the newly estimated position (in frame #448) and go back to the previous frame and re-compute the new position after adjusting the learning parameter $\lambda_t$.

### 3.2 Adjusting the Learning Parameter $\lambda_t$

In template-based tracking algorithms, the matched template is kept by trackers and needs online update according to target's appearance variation. However CT is not template-based, and CT does not keep templates but keeps an incrementally learned NB classifier. The main difference between template-based tracking and model-based tracking is that the former updates templates according to template clustering while the latter online updates classifier or other model parameters whenever track loss is detected.

Note that CT uses a fixed value for the cumulative model learning parameter $\lambda$, but we would like to enhance the adaptability of the algorithm by adjusting the value of $\lambda$ all the time depending on situations by considering the value of $d_t$. Note that $\lambda$ is the model learning parameter. Hence increase of $\lambda$ means that model update is dependent more on the previous tracking results, and the model is rather closer to a fixed classifier. On the other hand,

decrease of $\lambda$ means that model update is dependent more on new tracking results and the model is rather closer to a local model.

Hence we update the value of $\lambda$ based on the value $d_t$ at every frame by

$$\lambda_t = \lambda_{t-1} / (1+d_t) \quad (10)$$

However, if some track loss is detected by the result of MIM, it means the strategy no longer works, and we need to roll back to the previous frame and start over, resetting the value of $\lambda_t$ to some predetermined value $\lambda_R$.

### 3.3 Proposed Algorithm

Similarly to Renno et al.[11], a schematic diagram showing how the MIM process helps CT by detecting possible loss is given in Figure 2.

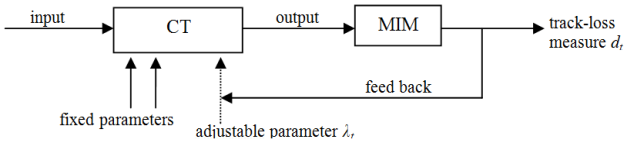


**Figure 2.** A schematic diagram of our proposed algorithm.

**Algorithm (MIM-aided CT)**

Let $x_1$ : the initial target position;
$d_T$ : threshold of track loss;
$\lambda_t$ : incremental learning parameter.
$\lambda_R$ : reset value of λt

1. Collect positive and negative training sets around z1. Extract features in reduced dimensionality. Learn parameters of the NB classifier $F_1(\ )$.
2. For frame index from $t$ =1 to $n$-1,

1) Sample a set of image patches around xt. Extract features in reduced dimensionality. Apply F$t$ ( ) to each sample vector to find the target candidate $x_{t+1}$ as in (3).
2) Collect new training samples around $x_{t+1}$.
3) Incrementally update the NB classifier to $F_{t+1}(\ )$ by λt.
4) Apply the MIM process using Ft ( ) as in (5) to find the target candidate $x_{t+1}$ by (2).
5) Compute track loss measure dt by (3).
6) If $d_t >= d_T$,
Roll $F_t(\ )$ back to $F_{t-1}(\ )$.
$\lambda_t=\lambda_R$
else
$\lambda_t = \lambda_{t-1} / (1+d_t)$

**Figure 3.** Our proposed algorithm.

After each normal CT step to detect the new position of a target in the next frame, we apply the MIM process to check if the newly obtained position is valid. If $d_t$ is smaller than some threshold $d_T$, then the result is considered to be satisfactory and we update $\lambda_t$ by (10) and continue to the next frame. Otherwise, some track loss might have occurred and we discard the result and go back to the previous frame, setting $\lambda_t = \lambda_R$ where $\lambda_R$ is some predefined value (We used 0.85).

The whole algorithm is given in Figure 3.

## 4. Results and Discussion

The original source code of CT and all the videos used for performance tests were obtained from[12]. The experiments were performed on a PC equipped with Intel® Core™ i3–3110M CPU running at 2.40 GHz and having 4 GB of main memory. The original CT tracker and our proposed MIM-aided CT tracker are both implemented using MATLAB running on 64-bit Windows 7, that runs around 30 frames per second. We manually measured the ground truth data on the videos frame by frame three times and averaged to obtain the ground truth for reference.

### 4.1 Test Run on the Video "David indoor"

To see the performance of CT depending on the learning parameter $\lambda$, we first ran CT 5 times on the video at different fixed values of $\lambda$ and obtained the average of the position deviation from the ground truths (which were manually obtained by marking the center on the screen). As shown in Table 1, the best result was obtained when $\lambda$ = 0.95.

**Table 1.** Performance of CT depending on the learning parameter

| Fixed learning parameter λ | Mean deviation from ground truths (pixels) |
|---|---|
| 0.75 | 19 |
| 0.80 | 22 |
| 0.85 | 20 |
| 0.90 | 21 |
| 0.95 | 17 |

Next, we ran CT with the most optimal value $\lambda$ = 0.95 on the video to see CT's performance and to identify which part of video may show possible track loss. The result is shown at the top of Figure 4. The horizontal axis is the frame number, and the vertical axis is the distance

measure $d_t$ in (3). The performance is stable with relatively small values of $d_t \approx 0.01$ throughout, except the beginning and near frames around 400.

Now we ran our MIM-aided CT with $\lambda_R = 0.95$ and $d_T = 0.02$ (Note that we continually change $\lambda_t$ throughout, and $\lambda_R$ is just the reset value when we start over in case of track loss.) and compared the result with that by CT with $\lambda = 0.95$. The results are shown at the bottom of Figure 4, the vertical axis being the deviation from ground truths in pixels. We can see that our MIM-aided CT (in blue) works better than the original CT (in red).

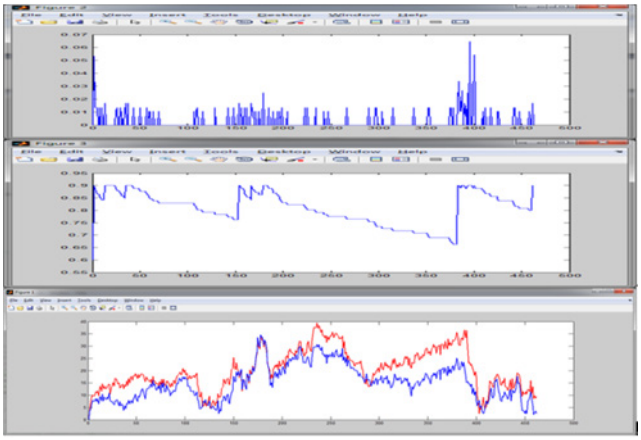

**Figure 4.** Test results using the video "David indoor".

The middle graph in Figure 4 shows how $\lambda_t$ changes. At most frames, the value of $\lambda_t$ changes slightly, but we can see two big and several small abrupt changes. Such abrupt changes in $\lambda_t$ mean that track loss was detected and its value was reset to $\lambda_R = 0.95$.

## 4.2 Test Run on the Video "Bolt"

We ran both the original CT and our MIM-aided CT on the video "Bolt"[12]. The video has 7 race runners and the situation in the video changes much faster and more dynamically compared with the previous video "David indoor". As before, we ran CT to track 3 runners at different values of $\lambda$, as we did in Table 1, and found that $\lambda = 0.85$ gives the best performance. Hence, for MIM-aided CT, we set $\lambda_R = 0.85$, and started with the initial value $\lambda_0 = 0.65$.

Figures 5, 6, and 7 show the tracking results of the race runners 1, 2, and 3, respectively. In each figure, the captured images in the upper row (in red) are by CT, while those in the lower row (in blue) are by MIM-aided CT. In all cases, MIM-aided CT was successful in tracking correct objects until the end, while the original CT always ended up with failure on the way.

In Figure 5, CT begins to lose the runner 1 from frame

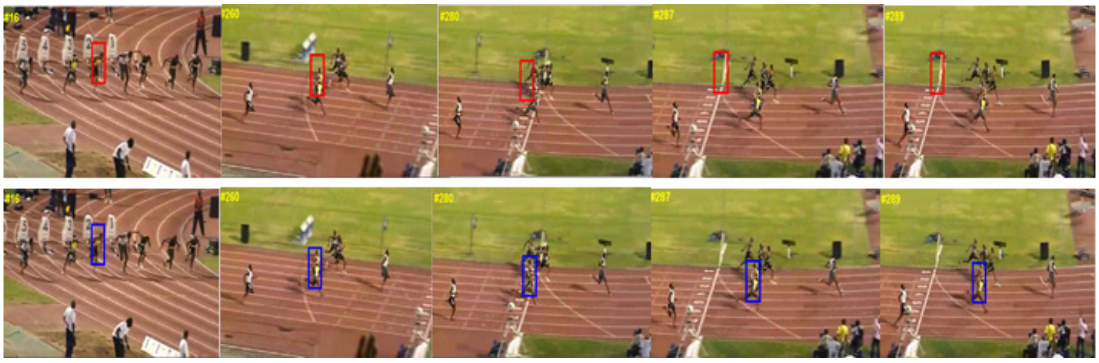

**Figure 5.** Tracking runner 1.

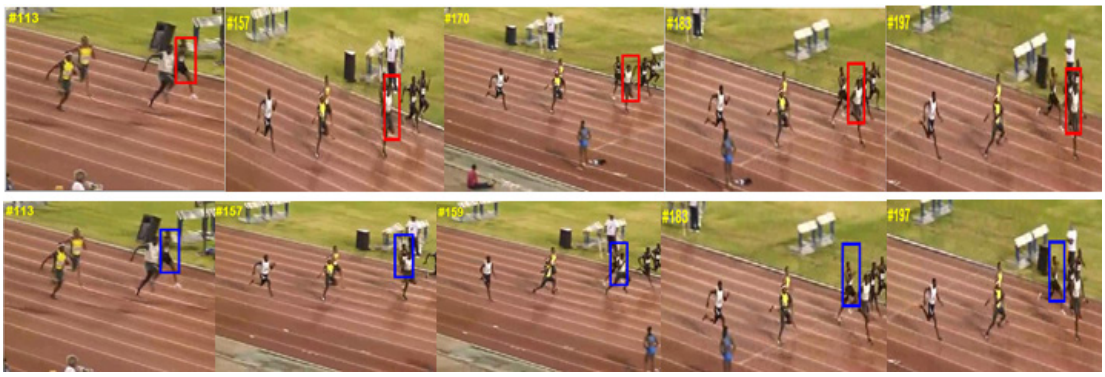

**Figure 6.** Tracking runner 2.

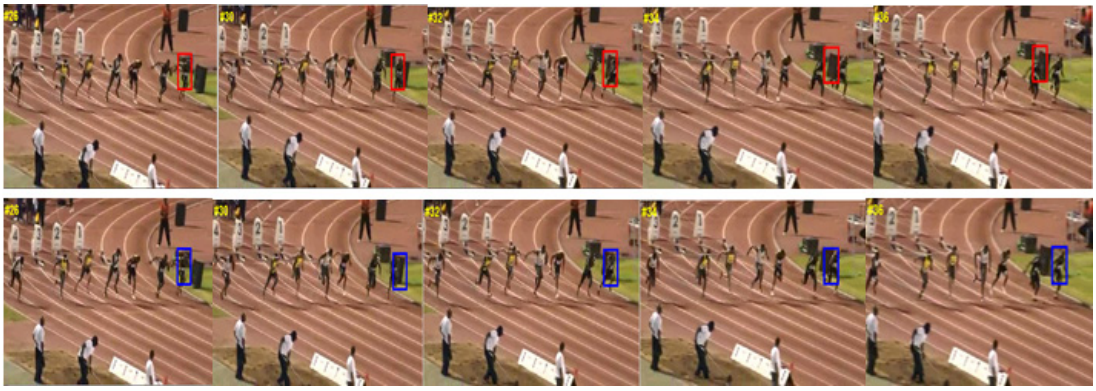

**Figure 7.** Tracking runner 3.

#280 and follows a wrong object until the end, while our MIM-aided CT with $d_T$=0.02 successfully follows the runner until the end.

In Figure 6, the runner 2 begins to be occluded by another runner (in white) from frame #113. CT begins to track the wrong runner from frame #157 while MIM-aided CT with $d_T$=0.03 succeeds to follow the correct runner until the end.

It is somewhat disappointing that CT failed to track the runners 1 and 2, following either the completely wrong object (i.e., the white line in Figure 5) or the runner in a completely different color (in Figure 6).

In Figure 7, CT loses the runner 3 and captures some wrong unmoving object in a similar color from frame #34. However MIM-aided CT with $d_T$= 0.03 succeeds to follow the correct runner until the end.

## 5. Acknowledgment

The authors would like to thank Dr. K. Zhang for making their source codes and videos publicly available with no restriction. This work was supported by the GRRC program of Gyeonggi province, Korea [GRRC SUWON 2015-B1, Cooperative Recognition and Response System based on Tension Sensing].

## 6. References

1. Zhang X, Hu W, Maybank S, Li X, Zhu M. Sequential particle swarm optimization for visual tracking. IEEE Conference on Proceedings of CVPR; 2008. p. 1–8.
2. Pang Y, Yuan Y, Li X, Pan J. Efficient HOG Human Detection. Signal Processing. 2011; 91(4):773–81.
3. Grabner H, Grabner M, Bischof H. Real-time tracking via on-line boosting. Proceedings of British Machine Vision Conference; 2006. p. 6.1–6.10.
4. Mei X, Ling H, Wu Y, Blasch E, Bai L. Minimum error bounded efficient ℓ-1 tracker with occlusion detection. IEEE Conference on Proceedings of CVPR; 2011. p. 1257–64.
5. Babenko B, Yang M.-H, Belongie S. Robust object tracking with online multiple instance learning. IEEE Trans Pattern Anal Mach Intell. 2011; 33(8):1619–32.
6. Chau DP, Bremond F, Thonnat M. Online evaluation of tracking algorithm performance. Proceedings of ICDP; 2009. p. 1–6.
7. Wu H, Zheng Q. Self-evaluation for video tracking systems. Center for Automation Research. College Park; 2004.
8. SanMiguel JC, Cavallaro A, Martinez JM. Evaluation of on-line quality estimators for object tracking, Proceedings of 17th IEEE International Conference on Image Processing; 2010. p. 825–8.
9. Liu R, Li SZ, Yuan X, He R. Online determination of track loss using template inverse matching. Proceedings of Eighth International Workshop on Visual Surveillance–VS; Marseille, France. 2008 Oct.
10. Zhang K, Zhang L, Yang MH. Real-time compressive tracking. Proceedings of 12th European Conference on Computer Vision – ECCV; 2012. p. 864–77.
11. Renno J, Lazarevic-McManus N, Makris D, Jones GA. Evaluating motion detection algorithms: Issues and results. Proceedings of 6th IEEE International Workshop on Visual Surveillance; 2006. p. 97–104.
12. A Real-time compressive tracker. Available from: http://www4.comp.polyu.edu.hk/~cslzhang/CT/CT.htm