Real-Time Compressive Tracking with a Particle Filter Framework

Xuan Yao and Yue Zhou

Institute of Image Processing and Pattern Recognition, Shanghai Jiao Tong University
yaoskd@sjtu.edu.cn

Abstract. Recently a real-time compressive tracking was proposed and achieved relative good results in terms of efficiency, accuracy and robustness. It belongs to the "tracking by detection" method. Slight inaccuracies in the tracker can lead to incorrectly labeled training examples in these algorithms, which degrade the classifier and usually cause drift. In this paper, we incorporate the motion model into the traditional compressive tracking where we utilize the particle filter. Therefore, our algorithm can handle drifting problem to some extent. Meanwhile, in order to improve the discriminative power of the classifier to relieve drifting problem radically, a modified naive Bayes classifier is proposed. The proposed algorithm performs favorably against state-of-the-art algorithms on some challenging video sequences.

Keywords: Compressive tracking, particle filter, naive Bayes classifier, tracking by detection.

1 Introduction

Object tracking is a well-studied problem in computer vision and has many applications. However, there is no single algorithm which will handle all circumstances, due to the complexity of the object and the environment, such as illumination change, pose and scale variation, occlusion.

As is stated in [1], a typical tracking system consists of three components:

- An appearance model, which can evaluate the likelihood that the object of interest is at some particular location.
- A motion model, which maintains the distribution of the locations of the object overtime.
- A search strategy for finding the most likely location in the current time.

Readers are recommended to [2] for a thorough overview of above three components.

Particle filter theory, also known as the bootstrap filter or sequential Monte Carlo filter, was first proposed from the field of signal processing[3], computer vision [4], and statistics[5], for solving the non-Gaussian and non-linear problems. It uses a set of weighted particles to approximate the location of the object. During tracking, particle filter maintains a distribution of the target's location and is characterized by the motion model.

C.K. Loo et al. (Eds.): ICONIP 2014, Part III, LNCS 8836, pp. 242–249, 2014.

Recently a large number of algorithms, known as the "tracking by detection", has thrived which focuses on the appearance model represented by the online learned classifier. Considering the successful application of Adaboost in the object detection [6], an on-line boosting algorithm [7][8][9] has been applied to the object tracking. In [7], a novel on-line Adaboost feature selection algorithm, known as OAB, is proposed for tracking. But it is sensitive to the noise and easy to drift. Later an online multiple instance learning (MIL) method [1] was proposed where samples are presented in sets, often called "bags" and labels are provided for the bags rather than the individual instances. The proposed method relieves the drifting problems and improves the accuracy of detection. The other relative method [10] uses a kernelized structured output support vector machine that learned online to provide adaptive tracking. Most of these above algorithms are suffered from heavy computational load that make it hard for real-time tracking and drift problems. Recently a novel tracking framework [11] was proposed that explicitly decomposes the tracking task into tracking, learning and detection. The PN learning constantly estimates the detector's error so that the algorithm can be applied to long-term tracking.

Compressive tracking [12], known as CT, adopts a very sparse measurement matrix to efficiently extract the features for the appearance model which makes it possible for real-time tracking and achieves relatively good results on some challenging video sequences. However, these algorithms often suffer from the drifting problem and the naive Bayes classifier used in the compressive tracking is quite simple. In this paper, to solve these problems, the key contribution of this work can be summarized as follows.

- We incorporate the particle filter into the traditional compressive tracking as the motion model to maintain the distribution of the location of the object. When drifting problem happens, the proposed algorithm will have a big chance to detect the object again.
- We calculate the correct rate of the classifier on each feature. The correct rate is then used as the weight of each feature's classifier. When the classifier classifies most of samples correctly, it is reasonable to believe the reliability of the classifier. Then the overall naive Bayes classifier is formulated as the weighted combination of each feature's classifier.

The paper is organized as follows: Section 2 briefly introduces the real-time compressive tracking proposed in [12]. Our method combining the above algorithms with particle filter is presented in Section 3. Then experimental results are shown in Section 4. Section 5 concludes the paper.

2 Compressive Tracking

A real-time compressive tracking method was proposed in [12]. In the paper, a very sparse measurement matrix was adopted to efficiently compress features from the foreground samples and background ones. The compression of feature is shown in Figure 1.

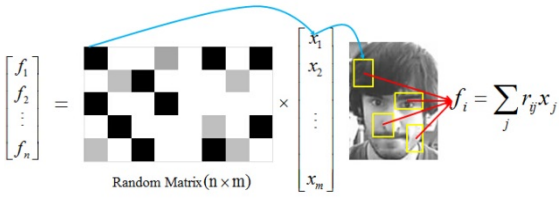


Fig. 1. Feature compression procedure, where **x** represents the original high-dimensional vector, matrix is a sparse random measurement matrix and **f** low-dimensional compressed feature

Then the tracking task was formulated as a binary classification problem with a simple naive Bayes classifier. The algorithm performs well in terms of accuracy, robustness, and speed.

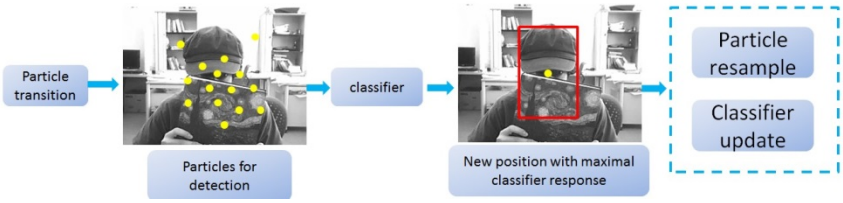


Fig. 2. Main components of our proposed algorithm at the t-th frame

3 Proposed Algorithm

3.1 Compressive Tracking Based on the Particle Filter Framework

In [12], the tracking problem is formulated as a detection task. To predict the object location in the next frame, samples are drawn from the neighborhood of the current target location. This may cause a problem that as long as drifting problems happens, the detection area will never cover the true neighborhood of the target which causes tracking failure. In our paper, to predict the object location, we draw samples from the particles. Thanks to the particle filter framework, after resampling step, there will always exists particles around the true location of target with relatively high weights, although maybe not the maximum weight. Once drifting happens, the target will be found with high probability in the subsequent frames. Meanwhile, it is worth considering that the examples for updating the classifier could be sampled from the particles. Furthermore, we use classifier response as the particle weight. Particle with the maximum classification score is determined as the object location in the next frame and particles with relative high weights are maintained while those with negligible weights discarded during resampling step. The main components of our algorithm are shown in Figure 2. Details are given in the following section.

3.1.1 Particle State Transition Model

We adopt a second-order autoregressive model

$$X_k - X_{k-1} = X_{k-1} - X_{k-2} + U_k \tag{1}$$

Where X_k represents the current states of particles and U_k the noise. It assumes that the motion between time k and time k-1 is the same as time k-1 and time k-2. In equation (1), we assume that the noise U_k are Gaussian.

$$U_k \sim N(0, \sigma_k) \tag{2}$$

where σ_k is the standard derivation. What should be mentioned is that when the target moves fast, σ_k should be increased. When the target moves slow, σ_k should be decreased. Furthermore, if σ_k is too large, more samples to predict will be included which will influence the determination of the classifier. If σ_k is too small, particles may not cover the whole area of the target.

3.1.2 Particle Weight

We use the classifier response as the weights of particles. However the classifier response often generates a negative value, which leads to a negative weight of a particle. Therefore, the next step is followed.

$$\mathrm{w}^i = \mathrm{w}^i - w_{\min} \tag{3}$$

From equation (3), all the weights will be positive. Then the normalization is performed and particles are resampled according to the weight so that more attention will be paid to the most likely area.

3.2 Improved Naive Bayes Classifier

The naive Bayes classifier in the compressive tracking is quite simple and its classification power is limited. We decide to take the correct rate of classifier on each feature into account.

We extract the expression that involve the specific feature from the overall Naive Bayesian classifier and the expression is defined as

$$p_i(\mathrm{i}) = \log\left(\frac{p(\mathrm{v}_i \mid \mathrm{y}=1)}{p(\mathrm{v}_i \mid \mathrm{y}=0)}\right) \tag{4}$$

If $p_i(\mathrm{i}) > 0$, the sample is considered as positive. If $p_i(\mathrm{i}) \le 0$, the sample is considered as negative. Here we define four parameters, n_c^+ (positive examples which are correct), n_f^+ (positive examples which are false), n_c^- (negative examples which are correct), n_f^- (negative examples which are false). Each feature's weight can be defined as

$$w_i = \frac{n_c^+ + n_c^-}{n_c^+ + n_c^- + n_f^+ + n_f^-} \tag{5}$$

The improved classifier

$$H(v) = \sum_{i=1}^{n} w_i \log\left(\frac{p(v_i \mid y=1)}{p(v_i \mid y=0)}\right) \tag{6}$$

3.3 The Algorithm

The pseudocode of our proposed algorithm is given below.

Algorithm. Compressive tracking based on the particle filter framework

Input: t-th video frame

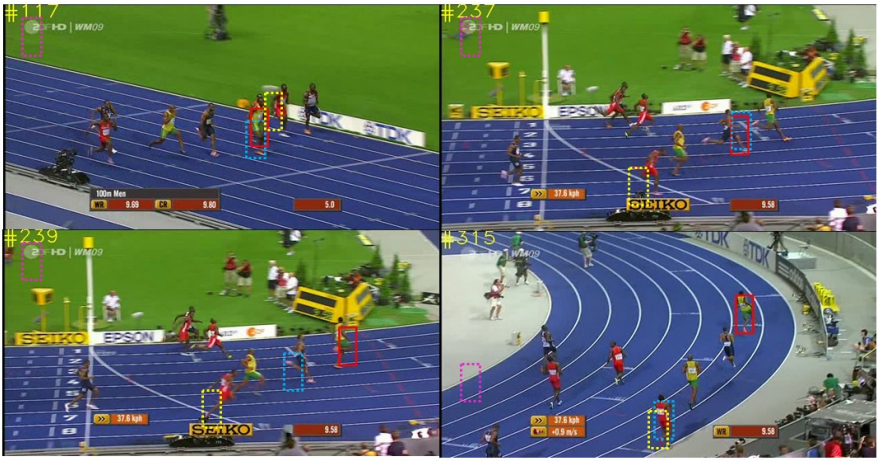
1. Particles resampled in the (t-1)-th frame transit according to the second-order autoregressive model, and features with low-dimensionality are extracted from the particles.
2. Use classifier in (6) to each particle, the classifier response is assigned to the particle weight and the particle in the location l_t with the maximal classifier response is found.
3. Normalize particle weights and resample particles according to weights
4. Sample two sets of image patches $D^{\alpha} = \{z \mid\mid l(z) - l_t \mid\mid < \alpha\}$ and $D^{\zeta,\beta} = \{z \mid \zeta < \mid\mid l(z) - l_t \mid\mid < \beta\}$ with $\alpha < \zeta < \beta$, where α, ζ, β are three parameters that we choose according to the experimental results, $l(z)$ is the center location of image patch used to update the classifier and D^{α}, $D^{\zeta,\beta}$ represent positive and negative samples respectively.
5. Extract the features with these two sets of samples and update the classifier.
6. Calculate the correct rate of each features' classifier on the above samples.

Output: Tracking location l_t and classifier parameters

4 Experiments

We evaluate our tracking algorithm with 3 state-of-the-art methods on some challenging video sequences. Four video sequences are presented in the Figure 3 to show advantages of our proposed algorithm over other methods. There are usually two evaluation methodologies which are the center location error (CLE) and bounding box overlap (BBO). We adopt the CLE for quantitative analysis which is showed in Table 1, for our algorithm is based on a fixed tracking scale and the ground truth usually a varied tracking scale so that the BBO methodology may not reflect the experimental results properly. Finally, our tracker is implemented in C++, which runs at 60 frames per second (FPS) on an Intel Core 3.20 GHz CPU with 4 GB RAM.

(a) Bolt



(b) Tiger 2

(c) Lemming

(d) David

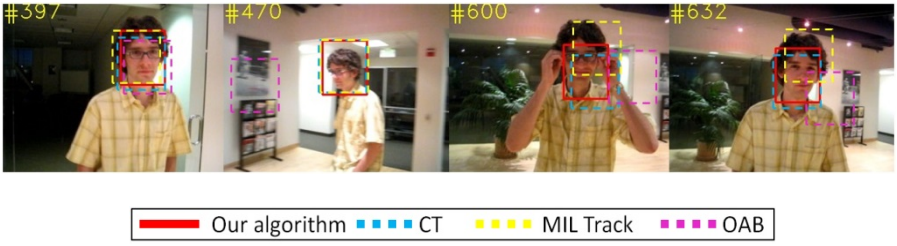


Fig. 3. Screenshots of some sampled tracking results

Table 1. Center location error. Red fonts indicate the best performance while the blue fonts indicate the second best ones

Sequence	Our algorithm	CT	OAB	MIL Track
Bolt	12	82	370	107
Tiger 2	10	18	26	8
Lemming	24	139	115	77
David	10	9	46	27
Average CLE	14	62	139	55

The ability to handle the drifting problem. As is known to us, the "tracking by detection" methods suffer from drifting problems where incorrectly labeled examples may degrade the discriminative power of the classifier and cause drift. In our proposed algorithm, when the tracking box drifts, there will still be lots of particles with rather high weights gathering around the neighborhood of the target. In subsequent frames, those particles may get the maximum classifier response with a relative high probability over the other non-target regions. The target player, Bolt, as shown in Figure 3(a) is almost lost in the frame 237 in both our algorithm and the compressive tracking because of the drastic appearance change after the finishing line. But our algorithm actually maintains some particles around the true target. As a result, in the frame 239, the mistake is corrected by our tracker while the traditional compressive tracker loses the target and never finds it back. The same situation is shown in the Figure 3(b) between the frame 149 and 151.

The improved discriminative power of the classifier. We calculate the correct rate of the classifier on each feature after update and the overall naive Bayes classifier is formulated as the weighted combination of each feature's classifier, which means that more samples are classified correctly by a certain feature's classifier, more we can trust on it. As shown in Figure 3(c), in the frame 229, the target is not detected precisely by compressive tracker and later in the frame 1049, the situation happens again when the appearance of the target changes dramatically which causes tracking failures. However, in our algorithm adopting the improved classifier, the tracking result has improved significantly.

From Table 1, our proposed tracker has the least average center location error among some state-of-art algorithms including compressive tracking. In the video sequence, Bolt and Lemming, drift problem happens and causes a big center location error in the compressive tracking while our algorithms achieve a rather good result.

5 Concluding Remarks

In this paper, we incorporate the particle filter framework into the compressive tracking. When detecting the target in the next frame, instead of searching in a neighborhood region of the previous loacation, we search from the particles resampled in the previous frame and use the classifier response as the particle weight. Meanwhile, the simple naive Bayes classifier is also modified to improve the discriminative power.

Experiments show that our proposed algorithm has the ability to handle the drifting problem and tracks object more robustly .

References

1. Babenko, B., Yang, M.H., Belongie, S.: Robust object tracking with online multiple instance learning. IEEE Transactions on Pattern Analysis and Machine Intelligence 33(8), 1619–1632 (2011)
2. Yilmaz, A., Javed, O., Shah, M.: Object tracking: A survey. Acm computing surveys (CSUR) 38(4), 13 (2006)
3. Gordon, N.J., Salmond, D.J., Smith, A.F.M.: Novel approach to nonlinear/non-Gaussian Bayesian state estimation. IEE Proceedings F (Radar and Signal Processing) 140(2), 107–113 (1993)
4. Isard, M., Blake, A.: Contour tracking by stochastic propagation of conditional density. In: Buxton, B.F., Cipolla, R. (eds.) ECCV 1996. LNCS, vol. 1064, pp. 343–356. Springer, Heidelberg (1996)
5. Liu, J.S., Chen, R.: Sequential Monte Carlo methods for dynamic systems. Journal of the American statistical association 93(443), 1032–1044 (1998)
6. Viola, P., Jones, M.: Rapid object detection using a boosted cascade of simple features. In: Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, CVPR 2001, vol. 1, pp. I-511–I-518. IEEE (2001)
7. Grabner, H., Grabner, M., Bischof, H.: Real-Time Tracking via On-line Boosting. BMVC 1(5), 6 (2006)
8. Grabner, H., Leistner, C., Bischof, H.: Semi-supervised on-line boosting for robust tracking. In: Forsyth, D., Torr, P., Zisserman, A. (eds.) ECCV 2008, Part I. LNCS, vol. 5302, pp. 234–247. Springer, Heidelberg (2008)
9. Stalder, S., Grabner, H., Van Gool, L.: Beyond semi-supervised tracking: Tracking should be as simple as detection, but not simpler than recognition. In: 2009 IEEE 12th International Conference on Computer Vision Workshops (ICCV Workshops), pp. 1409–1416. IEEE (2009)
10. Hare, S., Saffari, A., Torr, P.H.S.: Struck: Structured output tracking with kernels. In: 2011 IEEE International Conference on Computer Vision (ICCV), pp. 263–270. IEEE (2011)
11. Kalal, Z., Mikolajczyk, K., Matas, J.: Tracking-learning-detection. IEEE Transactions on Pattern Analysis and Machine Intelligence 34(7), 1409–1422 (2012)
12. Zhang, K., Zhang, L., Yang, M.-H.: Real-time compressive tracking. In: Fitzgibbon, A., Lazebnik, S., Perona, P., Sato, Y., Schmid, C. (eds.) ECCV 2012, Part III. LNCS, vol. 7574, pp. 864–877. Springer, Heidelberg (2012)