International Journal of Computer and Information Technology (ISSN: 2279 – 0764)
Volume 02– Issue 03, May 2013

A Novel Approach of On-line Discriminative Tracking Feature Selection

Yehong Chen
School of Information
Shandong Polytechnic University
Jinan, China

Pil Seong Park[*]
Department of Computer Science
University of Suwon
Suwon, Korea
pspark {at} suwon.ac.kr

Aimin Li
School of Information
Shandong Polytechnic University
Jinan, China

[*]*Corresponding author*

Abstract — **We propose a framework of a multi-object tracking method based on image classification. Aiming to build an appearance model of a target, a sparse measurement matrix is used as a projection for dimensional reduction and online feature selection to improve discriminative power. Our main idea is putting appropriate weights on the features selected to form a feature pattern. Using our method, it is easy to build a multi-object tracking framework, the tracking performance of which is dependent on the power of classifiers. Since our feature selection method is motivated by experience and from the observation of experimental results, we discuss why our method works well. Experiments show good performance of our method.**

Keywords-Sparse Random measurement, Naive Bayes classifier, Feature Selection, Multi-object Tracking

I. INTRODUCTION

Robust visual tracking, particularly of multiple targets, under real-world conditions is challenging, because it involves multi-camera, data association, etc. Our goal is to realize reliable multi-object tracking. Previous works on multi-object tracking focus on three different aspects [7]: motion models (e.g. Kalman filter or particle filter), searching methods (e.g. mean-shift), and object representation (e.g. classifiers).

Recently, tracking methods which are formulated as binary classification problems received a lot of attention due to their promising results [10]. From the point of object representation, building a powerful appearance model is very important. An active appearance model was introduced by Cootes et al. [5] and their approach is widely used for matching and tracking. An appearance model includes both object representation and classification.

Collins et al. [4] gives a summary of a variety of tracking methods, saying “Tracking success or failure depends primarily on how distinguishable an object is from its surroundings.” If an object is very distinctive, we can use a simple tracker to follow it and obtain a robust tracking result in general. Our hypothesis is that the features which are more distinguished from the background, are easier to track.

This feature ranking mechanism is usually embedded in a tracking system that adaptively selects the top-ranked discriminative features for tracking. The popular AdaBoost algorithm can be used as an embedded method for feature selection [6]. Okuma et al. [9] proposed an approach that combines the strength of two successful algorithms: mixture particle filters and AdaBoost. However it requires a lot of offline training.

To deal with a multi-object tracking reliability problem from the angle of pattern recognition, there are a lot of theoretical methods that can be applied [11], but these methods cannot be applied directly to tracking as they are. Han et al. [7] did a wonderful work on applying pattern recognition techniques in multi-target tracking. They first proposed a new approach for visual object tracking based on sample-based Adaptive Sparse Representation (AdaSR). They then extended the AdaSR for multi-object tracking by integrating the sample set of each tracked object.

In order to realize reliable multi-object tracking, our strategy is to build a powerful appearance model of a target. This model will focus on its discriminating power which is discussed a lot in [6]. Our goal can be achieved by combining the efforts of the two aspects synchronously, on the one hand, through improvement of pattern recognition technology, and on the other hand via more complex tracking framework.

Our paper is organized as follows: In section II, we introduce an efficient real time compressive tracking (CT) algorithm. In section III, we propose a novel feature selection algorithm, and explain its meaning. In section Ⅳ, multi-target tracking framework is introduced. Experimental results are presented and discussed in Sec. V, followed by conclusions and future work.

II. RELATED WORKS

In a video tracking task, object representation consists of extracted features (expressed as vectors) of an image patch which includes the target of interest. Feature extraction is essentially a dimension reduction problem with the goal of finding meaningful projections of the original data vectors. There are many such popular methods, e.g. PCA(Principal Component Analysis), NMF(non-negative matrix factorization) and approaches based on sparsity [6]. For example, a sparse expression once obtained by use of the L1 norm minimization [3]. Recently in face recognition, a compression sensing method was used for compressed representation in feature spaces, and achieved good results [12].

After dimensional reduction, the number of features is usually very huge. Feature selection is a related research area, whose goal is to pick a 'good' subset of features from some large set of candidate features [6]. Depending on the task, we can formulate the basis for feature selection.

Zhang et al. [12] used an off-line calculated random measurement matrix to make dimensional reduction and built an appearance model including a general template with 50 features and classification based on a naive Bayes classifier. We obtained the source codes from [13], which we will call CT (compress tracking). CT is a state-of-the-art algorithm, in terms of efficiency, accuracy and robustness. It has a concise code and efficient operating efficiency.

We would like to improve the CT by embedding an online feature selection function, and then extend it to a multi-object tracking framework. The following is a brief summary of CT. For more details, see [1,2,8,12].

A. Two-stage framework

The whole tracking process includes two stages: learning and detection, as shown in Figure 1. A) and B) belongs to the training stage, and C) and D) belong to the detection stage. A) shows the frame where we manually select a target. B) Labeled samples(Target samples are labeled positive, and background samples are labeled negative), which are then used for learning/updating a classifier. C) samples detectors as input of a classifier to search for objects of interest. D) Then we obtain the maximum posteriori as tracking result. The entire tracking process is a cycle alternating between these two stages.

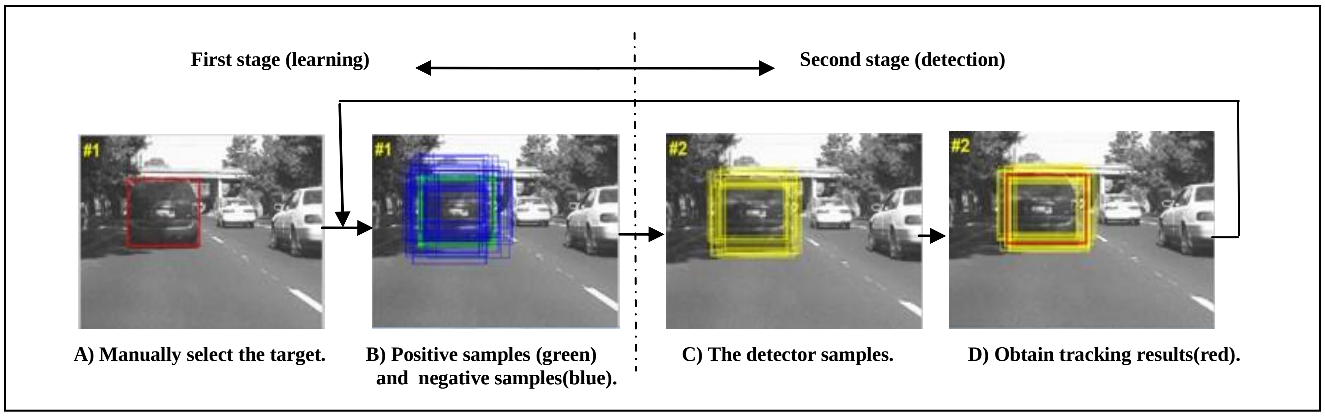
First stage (learning)
Second stage (detection)
#1
#1
#2
#2
A) Manually select the target.
B) Positive samples (green) and negative samples(blue).
C) The detector samples.
D) Obtain tracking results(red).

Figure 1. A two-stage tracking loop. The movie was obtained from the VIVID benchmark dataset PkTest02 [15].

B. Dimensional reduction

In CT, for each sample $Z \in \mathbb{R}^{w \times h}$, where w is the width and h is the height of an image patch, we represent it by convolving an image with a set of rectangular filters at multiple scales and multiple positions. A high dimensional representation of a sample is a vector $x = (x_1, ..., x_m) \in \mathbb{R}^m$, where $m = (wh)^2$. CT uses a very sparse random measurement matrix $R \in \mathbb{R}^{n \times m}$, $m>>n$, to project a high dimensional space $X \in \mathbb{R}^m$ onto a low dimensional space $V = (v_1, \dots, v_n) \in \mathbb{R}^n$, by the equation $V=RX$, as long as it satisfies the restricted isometry property [1,12], in other words, all pairwise distances are maintained within an arbitrarily small factor.

CT uses R, *where* $R(i,j)=r_{ij}$

$$r_{ij} = \sqrt{s} \times \begin{cases} 1 & \text{with probability } 1/2s \\ 0 & \text{with probability } 1\text{-}1/s \\ -1 & \text{with probability } 1/2s \end{cases} \quad (1)$$

$$i=1,...,n,\ j=1,...,m.$$

In CT, $s = m/4$, m is typically in the order of 10^6 to 10^{10}, which makes a very sparse random matrix. For each row of R, only about c, $c \le 4$, entries need to be computed. Therefore, the computational complexity is only $O(cn)$, which is very low. Furthermore, we only need to store the nonzero entries of R so that we can save memory. For more details, see [1,12].

Since the entries of R can be either positive or negative, the compressive features $v_i \in \sum_{j=1}^{n} r_{ij} x_j$, computes relative intensity difference in a way similar to the generalized Haar-like features, which is a linear combination of randomly generated Haar-like features [2,6,12].

C. Classifier construction and online update

In [2,8,12], for each sample $X \in \mathbb{R}^m$, its low-dimensional representation is $V=(v_1,...,v_n)^T \in \mathbb{R}^n$ with $m>>n$. We assume that v_i (feature ingredients) are independently distributed, and model them with a naive Bayes classifier. A naive Bayes classifier responses to two-class variance ratio measure, and uses log likelihood values computed from empirical distributions of objects and background.

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

In [2,8,12], they assumed uniform prior, $p(y=1)= p(y=0)$, and $y \in \{0, 1\}$ is a binary variable which represents the sample label (0/1 for negative/positive). The random projections of high dimensional random vectors are almost always Gaussian distributed. Thus, the conditional distributions $p(v_i|y=1)$ and $p(v_i|y=0)$ in the classifier $H(v)$ are assumed to be Gaussian distributed through labeled samples, with four parameters $(\mu_i^1, \sigma_i^1, \mu_i^0, \sigma_i^0)$ where

$$p(v_i \mid y=1) \sim N(\mu_i^1, \sigma_i^1) \quad (3)$$
$$p(v_i \mid y=0) \sim N(\mu_i^0, \sigma_i^0) \quad (4)$$

μ_i^1 and σ_i^1 are the mean and variance of the positive sample set X^1 at the i-th feature, and μ_i^0 and σ_i^0 are the mean and variance of negative sample set X^0 at the i-th feature. $H(v)$ is learned and updated online, according to the current training sample sets X^1 and X^0 (Figure 1.B).

D. Search a matching object

In Figures 1.C) and 1.D), $Max(H(v))$ in detector set D with maximum a posteriori will be taken as the target object. We can get the position of the target by $T(x,y;w,h)$, where (x,y) is the coordinate of the left top of the target, and w and h are the width and the height of the target box, respectively.

III. ON-LINE SELECTION OF DISCRIMINATIVE TRACKING FEATURES

A. Problem analysis and improvement

In CT, R is calculated offline. Once R is generated, it will be kept constant throughout. R can be thought of as a feature extraction template. Since $V=RX$, the compressive feature V corresponds to a linear combination of template generated Haar-like features [2,6,12]. For example, Figure 2 demonstrates the template.

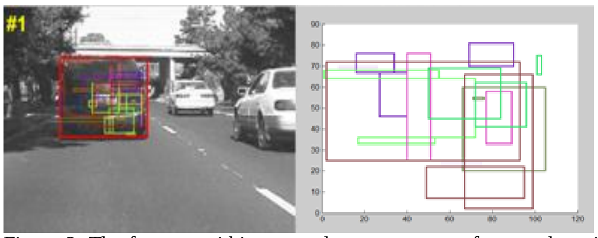

Figure 2. The features within a sample, every group of rectangles with the same color corresponds to a random Haar-like feature. The figure on the right is a general template, and every feature in it is equally important. The movie is from the VIVID benchmark dataset PkTest02 [15].

This template is shared by all samples. In other words, different objects are represented by the same feature set. We want to make some improvement here, so as to select a different feature set with different objects for classification, thereby improving the discriminative power of objects.

B. Embedded feature selection

1) Feature selection

Our goal is to choose different feature sets for different objects, and assign weights to these features. Some machine learning methods such as AdaBoost are widely used to select a weighted feature set, and it is adaptive to tracking procedure [6]. But in an online learning case, we have neither enough samples nor enough time to train our classifiers.

From (2), (3), and (4), H is obtained by training, and can be thought of as a linear combination of h_i which is a weaker classifier corresponding to a feature.

$$h_i = \log\left(\frac{p(v_i \mid y=1)}{p(v_i \mid y=0)}\right) \quad (5)$$

We can construct a matrix h $(n \times l)$, where l is the number of samples and

$$h_{ij} = \log\left(\frac{p(v_{ij} \mid y=1)}{p(v_{ij} \mid y=0)}\right), \quad (6)$$

Here i is the index of each feature $(i=1, \ldots,n)$, and j is the index of samples in the test set. v_{ij} is the i-th feature component of the j-th sample; $h_{ij}>0$ represents that the i-th feature of the j-th sample is classified as positive, and negative otherwise. If we average the data by row of h, we get a column vector $\bar{h}_i$. In the learning stage (Figure 1.B), after learning H, the system uses formula (2), followed by feature selection.

Since we use a positive sample set X^1 to select features, we expect that every feature will show a positive response to the classifier. If some features show negative responses, we may think that some detection error might have occurred on those features. Based on the above analysis, every feature with classifier response less than 0 gets weight 0, and those with positive responses get a weight value normalized to [0,1].

Algorithm 1. Feature selection

Input: Learned classifier H from X^1 and X^0 .

```
1. Use $X^1$ as a test set to calculate the matrix $H$ by (6).
2. $\bar{h}_i = \sum_{j=1}^{l} h_{ij} / l$                (7)
3. $W_i = \begin{cases} 0, \bar{h}_i \le 0 \\ \bar{h}_i / \sum_{j=1}^{n} \bar{h}_j, \bar{h}_i > 0 \end{cases}$    (8)
4. $H'(v) = \sum_{i=1}^{n} W_i \bar{h}_i$                (9)
Output: $H'$
```

From now, we call the CT with embedded feature selection DCT (discriminative CT).

2) *Brief discussion:*

Our feature selection is from intuition, and needs some more reasonable explanation.

a) *Feature pattern*

We run CT to track different objects, hence $\bar{h}_i$ from X^1 has a different pattern(Figure 3). And these feature patterns will change continually with the process of tracking. The existence of such a pattern explains that selecting different feature sets for different objects and assigning weights to these features are reasonable.

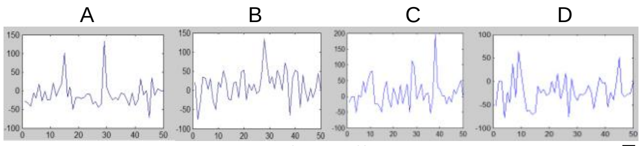

Figure 3. Feature structures from different targets, *y*-axis value is $\bar{h}_i$, *x*-axis related to *n* features. Different objects get different feature pattern. (A for a car, B for a human face, C for hockey player, and D for a walking human.)

From observation, positive and negative samples are Gaussian distributed at every feature value. Since positive samples show narrow peak distribution, using average approximation $\bar{h}_i$ as feature pattern seems to be reasonable.

In addition, we hope that a positive sample gains positive features combination, so that if $\bar{h}_i$<0, the *i*-th feature will be removed from feature structure. Intuitively, different features have different discriminative power, therefore we give weights to a weaker classifiers $\bar{h}_i$ to build a strong classification H'.

b) *What do weights of features mean?*

$H'(v)$ is based on selected features (9). In the second stage (Figure 1.C), if $H'(V_i) > 0$, then the *i*-th feature in detector will be enhanced by weight. If $H'(V_i) > 0$, there may exists occlusion or appearance change in the *i*-th feature region, and then the feature was eliminated by weight. A sample enhanced by weight will have high possibility to be selected as a target by all detectors. The sample supports the selected weighted features which are important in feature structure, and has higher possibility be detected as a target object.

So these weights of features have amplification effect, which zoom those features contrasting backgrounds, thereby improving the discriminative power of the appearance model.

In addition, through on-line selection of discriminating tracking features, if some occlusion occurs in such a removed feature region, occlusion effect may be eliminated.

IV. MULTI-TARGET TRACKING FRAMEWORK

If an object is very distinctive, we can use a simple tracker to follow it and obtain robust tracking results in general. Compared with CT, DCT enhances the discriminating power of the classifier. We modify and improve the DCT for multi-target tracking, which we will call M-DCT later.

A. *Iteration shift*

In the second stage, given the target's position in the last frame *T* $(x_0, y_0; w_0, h_0)$, we can sample detector set *D* and use the strong classifier H' to get a confidence MAP (maximum a posteriori) of *T(x, y; w, h).* We added iteration shift to realize convergence of tracking position.

Algorithm 2. Iteration-shift

```
Input: $T(x_0,y_0;w_0,h_0)$, Є (the deviation in predicted location.)
  Repeat until $\Delta d \le$ Є.
    1. Get $T(x,y;w,h)$ from MAP of $H'_k(V_{T(x_0,y_0;w_0,h_0)})$
    2. $\Delta d = \sqrt{(x-x_0)^2+(y-y_0)^2}$ .
    3. $x_0=x$ , $y_0 = y$
Output: Tracking position $T(x,y;w,h)$.
```

B. *Extension to multi-target tracking*

We thought tracking problem as a classification problem which is relatively easy to extend to Multi-target situation. We learn a couple of classifiers *H* with a couple of training sets corresponding to different objects. The framework is ease to build, and the performance is due to power of classification H. Figure 5 gives the two stage structure of the final framework of multi-target tracking.

Algorithm 3. M-DCT

```
Input: $T_k (x_0, y_0; w_0, h_0)$ as posteriori ($1 \le k \le c$, where $k$ is the
       object index) and Є.,
For $k$=1 to $c$ , until the end of the video,
  1. Sample $X^1_k$ , $X^0_k$ , and learn/update $H_k$ by (2), (3) and (4).
  2. Using $H_k$, test $X^1_k$.
  3. Build $H'_k$ by (7), (8), and (9).
  4. Go to the next frame, $T_k (x_0, y_0; w_0, h_0)$ as estimation,
     $1 \le k \le c$.
  5. Sample detectors $D_k$ .
  6. Apply Algorithm 2(Iteration-shift).
  7. Get a posteriori $T_k (x,y;w,h)$, and draw a tracking box,
     $T_k(x_0,y_0;w_0,h_0) \longleftarrow T_k(x,y;w,h)$.
```

V. EXPERIMENTS

We name our final multi-object tracking program M-DCT(multi-discriminative CT), and it can also track a single target. We compare the result by M-DCT with that by CT under same basic parameters.

Given a target location in the current frame, the search radius for drawing positive samples is set to α = 4 which generates 45 positive samples. For negative samples, we set to ζ =8 and β = 30, respectively.

This program randomly selects 50 negative samples from the set $X^{\varsigma,\beta}$. The search radius for the set D^{γ} to detect the object location is set to γ =20 and about 1,100 samples are generated. The dimension of the projected space is set to *n* = 50. Beyond that, we carefully tune some parameters for different videos.

Under the same parameters set, M-DCT is proven to be more robust in handling occlusion problems than CT.

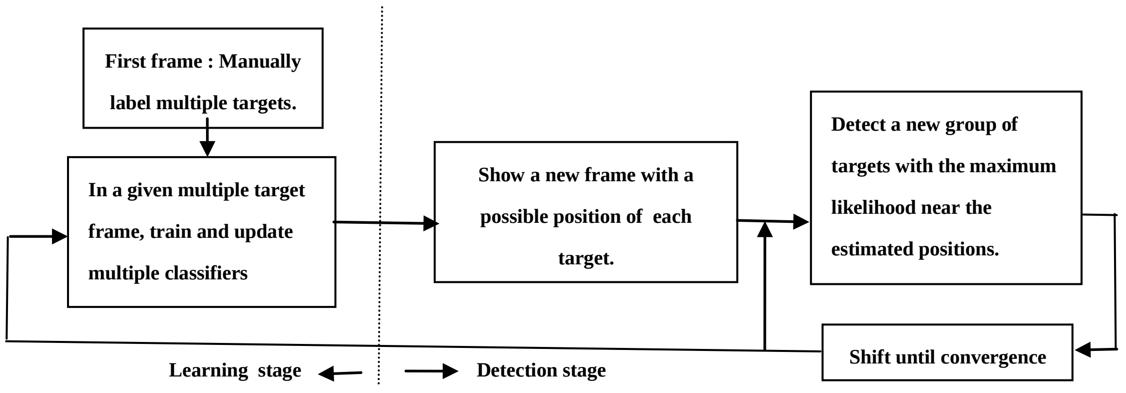
First frame : Manually label multiple targets.
In a given multiple target frame, train and update multiple classifiers
Show a new frame with a possible position of each target.
Detect a new group of targets with the maximum likelihood near the estimated positions.
Shift until convergence
Learning stage
Detection stage

Figure 5. Our framework of multi-target tracking

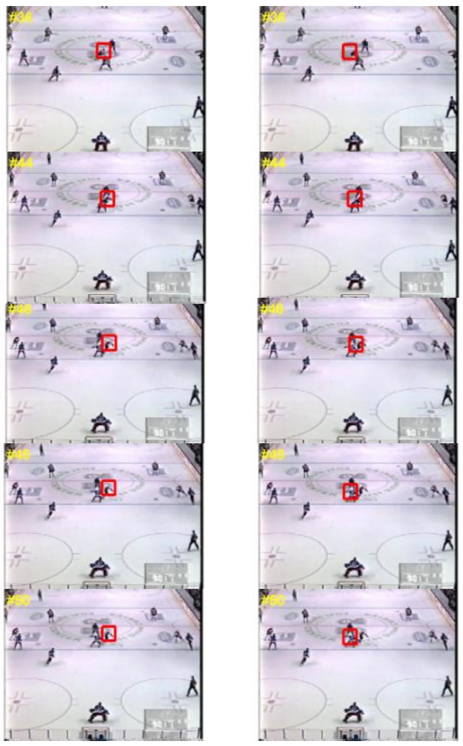

Figure 6. Occlusion of three hockey players. The right column is the results by CT, and the left column is by our improved program M-DCT. The video is obtained from [14].

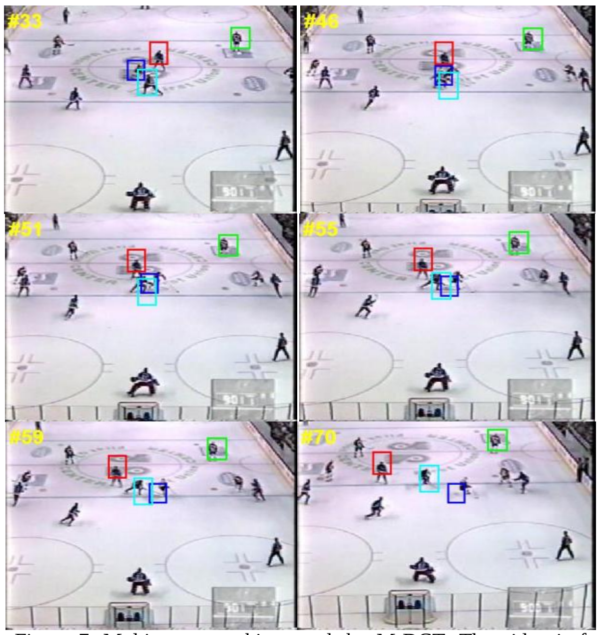

Figure 7. Multi-target tracking result by M-DCT. The video is from [14].

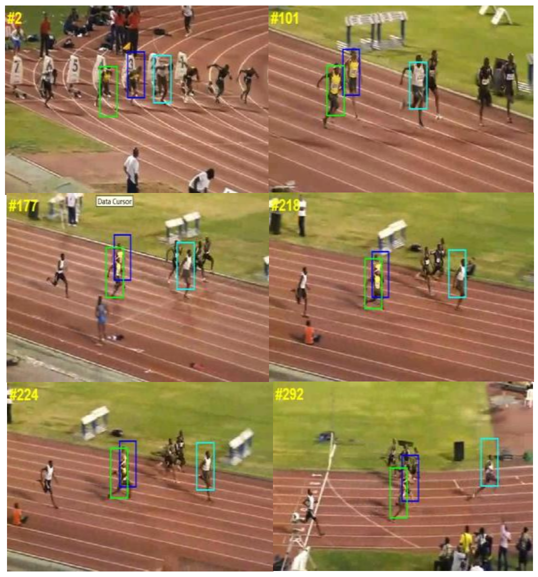

Figure 8. Another multi-target tracking result by M-DCT. The video was obtained from [13]

A. *Tracking results*

Figure 6 shows a single-object tracking result when occlusion occurs. When the CT was used, the red bounding box drifted to some other wrong player at the end. However our M-DCT gives the correct tracking result throughout.

We also tested our M-DCT on the same video to track multiple objects simultaneously. Figure 7 shows a correct tracking result when two players(one in the deep blue box and the other in the cyan box) move across each other, even though the player in the red box were present very close to them.

In Figure 8, runners in blue or green boxes were very close to each other in the process of the contest. However the M-DCT was successful to distinguish the two runners.

B. Experiment discussion and limitation

We obtained satisfactory results from the above experiments, but in real world tracking tasks, this method still has some limitations.

The target appearance may change in the sequences of video, but a computer cannot tell if such changes are from occlusion or the variance of appearance model. So online learning may be a two-sided sword. On the one hand , if changes were from the variance of appearance model, on-line appearance model update will adapt to the tracking task; on the other hand, if there was occlusion, on-line update may cause a drift to some other object.

VI. CONCLUSION AND FUTURE WORK

In this paper, we proposed a concise multi-target tracking algorithm based on image classification involving online feature selection. Our algorithm combines the compressed representation by a random measurement matrix and a novel feature selection method. This combination has been seldom employed in other literature. The purpose of this framework is to provide an experimental platform to promote both pattern recognition and multi-target tracking in the future work, and eventually to get reliable multi-object tracking results.

Our multi-target tracking framework is succinct, and its performance is good due to an appearance model, but not considering a motion model. In future works, we will involve a motion model of the target, combining detection and probability reference method to improve the performance of tracking tasks.

ACKNOWLEDGMENT

The authors would like to thank Dr. K. Okuma and Dr. K. Zhang for making their precious videos publicly available with no restriction. This work was supported by the GRRC program of Gyeonggi province [GRRC SUWON2012-B1, Cooperative CCTV Image Based Context-Aware Process Technology].

REFERENCES

[1] D. Achlioptas. “Database-friendly random projections: Johnson-Lindenstrauss with binary coins”, J. Comput. Syst. Sci, vol. 66, no. 4,pp.671–687, 2003,doi: 10.1016/S0022-0000(03)00025-4.

[2] B. Babenko, M.-H. Yang, and S. Belongie. “Robust object tracking with online multiple instance learning”, IEEE Tran. J. Pattern Analysis and Machine Intelligence, Vol. 33, pp.1619-1632, Aug. 2011.

[3] C. Bao, Y. Wu , H. Ling , and H. Ji. “ Real Time Robust L1 Tracker Using Accelerated Proximal Gradient Approach”, IEEE CVPR, pp. 1830 – 1837, 2012 .

[4] R. Collins, Y. Liu, and M. Leordeanu. “On-Line Selection of Discriminative Tracking Features”, IEEE Transaction J. Pattern Analysis and Machine Intelligence, Vol. 27, No. 10, pp. 1631 – 1643,October, 2005.

[5] T. F. Cootes, G. J. Edwards, and C. J. Taylor. "Active appearance models", ECCV'98, Lecture Notes in Computer Science, pp.681-685, 1998, doi:10.1007/BFb0054760.

[6] P. Dollar, Z. Tu, H. Tao, and S. Belongie. “Feature mining for image classification,” IEEE Conference on CVPR’07, pp.1-8, June 2007. .

[7] Z. Han, J. Jiao, B. Zhang, Q. Ye , and J. Liu. "Visual object tracking via sample-based Adaptive Sparse Representation (AdaSR)” ,J. Pattern Recognition,Vol. 44, Issue 9, Pages 2170–2183, September ,2011.

[8] Z. Kalal, J. Matas, and K. Mikolajczyk. “P-N Learning: Bootstrapping Binary Classifiers by Structural Constraints”, CVPR, pp.49-56, 2010. .

[9] K. Okuma, A. Taleghani, N. D. Freitas, J. J. Little, and D. G. Lowe. “Boosted Particle Filter: Multitarget Detection and Tracking”, ECCV04, Part I, pp. 28-39, May 2004, doi :10.1007/978-3-540-24670-1_3. .

[10] S. Stalder, H. Grabner, and L. van Gool. “Beyond Semi-Supervised Tracking:Tracking Should Be as Simple as Detection, but not Simpler than Recognition”, IEEE ICCV Workshops, pp. 1409 – 1416, Oct. 2009. .

[11] J. Vermaak, A. Doucet, and P. Perez . ”Maintaining Multi-Modality through Mixture Tracking”, ICCV03, Vo.2, pp. 1110-1116, 2003.

[12] K. Zhang, L. Zhang, and M. Yang. “Real-Time Compressive Tracking”, ECCV, 2012. .

[13] http://www4.comp.polyu.edu.hk/~cslzhang/CT/CT.htm. .

[14] http://www.cs.ubc.ca/~okumak/research.html.

[15] http://www.dabi.temple.edu/~hbling/code_data.htm.