Real-time Compressive Tracking

Kaihua Zhang[1], Lei Zhang[1], and Ming-Hsuan Yang[2]

[1]Depart. of Computing, Hong Kong Polytechnic University
[2]Electrical Engineering and Computer Science, University of California at Merced
{cskhzhang,cslzhang}@comp.polyu.edu.hk, mhyang@ucmerced.edu

Abstract. It is a challenging task to develop effective and efficient appearance models for robust object tracking due to factors such as pose variation, illumination change, occlusion, and motion blur. Existing online tracking algorithms often update models with samples from observations in recent frames. While much success has been demonstrated, numerous issues remain to be addressed. First, while these adaptive appearance models are data-dependent, there does not exist sufficient amount of data for online algorithms to learn at the outset. Second, online tracking algorithms often encounter the drift problems. As a result of self-taught learning, these mis-aligned samples are likely to be added and degrade the appearance models. In this paper, we propose a simple yet effective and efficient tracking algorithm with an appearance model based on features extracted from the multi-scale image feature space with data-independent basis. Our appearance model employs non-adaptive random projections that preserve the structure of the image feature space of objects. A very sparse measurement matrix is adopted to efficiently extract the features for the appearance model. We compress samples of foreground targets and the background using the same sparse measurement matrix. The tracking task is formulated as a binary classification via a naive Bayes classifier with online update in the compressed domain. The proposed compressive tracking algorithm runs in real-time and performs favorably against state-of-the-art algorithms on challenging sequences in terms of efficiency, accuracy and robustness.

1 Introduction

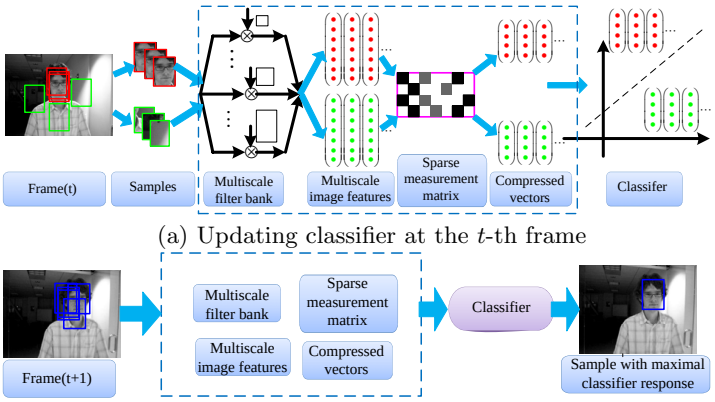
Despite that numerous algorithms have been proposed in the literature, object tracking remains a challenging problem due to appearance change caused by pose, illumination, occlusion, and motion, among others. An effective appearance model is of prime importance for the success of a tracking algorithm that has been attracting much attention in recent years [1–10]. Tracking algorithms can be generally categorized as either generative [1, 2, 6, 10, 9] or discriminative [3–5, 7, 8] based on their appearance models.

Generative tracking algorithms typically learn a model to represent the target object and then use it to search for the image region with minimal reconstruction error. Black et al. [1] learn an off-line subspace model to represent the object of interest for tracking. The IVT method [6] utilizes an incremental subspace model

to adapt appearance changes. Recently, sparse representation has been used in the ℓ_1-tracker where an object is modeled by a sparse linear combination of target and trivial templates [10]. However, the computational complexity of this tracker is rather high, thereby limiting its applications in real-time scenarios. Li et al. [9] further extend the ℓ_1-tracker by using the orthogonal matching pursuit algorithm for solving the optimization problems efficiently. Despite much demonstrated success of these online generative tracking algorithms, several problems remain to be solved. First, numerous training samples cropped from consecutive frames are required in order to learn an appearance model online. Since there are only a few samples at the outset, most tracking algorithms often assume that the target appearance does not change much during this period. However, if the appearance of the target changes significantly at the beginning, the drift problem is likely to occur. Second, when multiple samples are drawn at the current target location, it is likely to cause drift as the appearance model needs to adapt to these potentially mis-aligned examples [8]. Third, these generative algorithms do not use the background information which is likely to improve tracking stability and accuracy.

Discriminative algorithms pose the tracking problem as a binary classification task in order to find the decision boundary for separating the target object from the background. Avidan [3] extends the optical flow approach with a support vector machine classifier for object tracking. Collins et al. [4] demonstrate that the most discriminative features can be learned online to separate the target object from the background. Grabner et al. [5] propose an online boosting algorithm to select features for tracking. However, these trackers [3–5] only use one positive sample (i.e., the current tracker location) and a few negative samples when updating the classifier. As the appearance model is updated with noisy and potentially misaligned examples, this often leads to the tracking drift problem. Grabner et al. [7] propose an online semi-supervised boosting method to alleviate the drift problem in which only the samples in the first frame are labeled and all the other samples are unlabeled. Babenko et al. [8] introduce multiple instance learning into online tracking where samples are considered within positive and negative bags or sets. Recently, a semi-supervised learning approach [11] is developed in which positive and negative samples are selected via an online classifier with structural constraints.

In this paper, we propose an effective and efficient tracking algorithm with an appearance model based on features extracted in the compressed domain. The main components of our compressive tracking algorithm are shown by Figure 1. Our appearance model is generative as the object can be well represented based on the features extracted in the compressive domain. It is also discriminative because we use these features to separate the target from the surrounding background via a naive Bayes classifier. In our appearance model, features are selected by an information-preserving and non-adaptive dimensionality reduction from the multi-scale image feature space based on compressive sensing theories [12, 13]. It has been demonstrated that a small number of randomly generated linear measurements can preserve most of the salient information and allow almost



(a) Updating classifier at the t-th frame

(b) Tracking at the $(t+1)$-th frame

Fig. 1. Main components of our compressive tracking algorithm.

perfect reconstruction of the signal if the signal is compressible such as natural images or audio [12–14]. We use a very sparse measurement matrix that satisfies the restricted isometry property (RIP) [15], thereby facilitating efficient projection from the image feature space to a low-dimensional compressed subspace. For tracking, the positive and negative samples are projected (i.e., compressed) with the same sparse measurement matrix and discriminated by a simple naive Bayes classifier learned online. The proposed compressive tracking algorithm runs at real-time and performs favorably against state-of-the-art trackers on challenging sequences in terms of efficiency, accuracy and robustness.

2 Preliminaries

We present some preliminaries of compressive sensing which are used in the proposed tracking algorithm.

2.1 Random projection

A random matrix $R \in \mathbb{R}^{n\times m}$ whose rows have unit length projects data from the high-dimensional image space $\mathbf{x} \in \mathbb{R}^m$ to a lower-dimensional space $\mathbf{v} \in \mathbb{R}^n$

$$\mathbf{v} = R\mathbf{x}, \tag{1}$$

where $n \ll m$. Ideally, we expect R provides a stable embedding that approximately preserves the distance between all pairs of original signals. The Johnson-Lindenstrauss lemma [16] states that with high probability the distances between the points in a vector space are preserved if they are projected onto a randomly selected subspace with suitably high dimensions. Baraniuk et al. [17] proved that the random matrix satisfying the Johnson-Lindenstrauss lemma also holds

true for the restricted isometry property in compressive sensing. Therefore, if the random matrix R in (1) satisfies the Johnson-Lindenstrauss lemma, we can reconstruct $\mathbf{x}$ with minimum error from $\mathbf{v}$ with high probability if $\mathbf{x}$ is compressive such as audio or image. We can ensure that $\mathbf{v}$ preserves almost all the information in $\mathbf{x}$. This very strong theoretical support motivates us to analyze the high-dimensional signals via its low-dimensional random projections. In the proposed algorithm, we use a very sparse matrix that not only satisfies the Johnson-Lindenstrauss lemma, but also can be efficiently computed for real-time tracking.

2.2 Random measurement matrix

A typical measurement matrix satisfying the restricted isometry property is the random Gaussian matrix $R \in \mathbb{R}^{n \times m}$ where $r_{ij} \sim N(0, 1)$, as used in numerous works recently [14, 9, 18]. However, as the matrix is dense, the memory and computational loads are still large when m is large. In this paper, we adopt a very sparse random measurement matrix with entries defined as

$$r_{ij} = \sqrt{s} \times \begin{cases} 1 & \text{with probability } \frac{1}{2s} \\ 0 & \text{with probability } 1 - \frac{1}{s} \\ -1 & \text{with probability } \frac{1}{2s}. \end{cases} \tag{2}$$

Achlioptas [16] proved that this type of matrix with $s = 2$ or 3 satisfies the Johnson-Lindenstrauss lemma. This matrix is very easy to compute which requires only a uniform random generator. More importantly, when $s = 3$, it is very sparse where two thirds of the computation can be avoided. In addition, Li et al. [19] showed that for $s = O(m)$ ($\mathbf{x} \in \mathbb{R}^m$), this matrix is asymptotically normal. Even when $s = m/\log(m)$, the random projections are almost as accurate as the conventional random projections where $r_{ij} \sim N(0, 1)$. In this work, we set $s = m/4$ which makes a very sparse random matrix. For each row of R, only about c, $c \leq 4$, 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 which makes the memory requirement also very light.

3 Proposed Algorithm

In this section, we present our tracking algorithm in details. The tracking problem is formulated as a detection task and our algorithm is shown in Figure 1. We assume that the tracking window in the first frame has been determined. At each frame, we sample some positive samples near the current target location and negative samples far away from the object center to update the classifier. To predict the object location in the next frame, we draw some samples around the current target location and determine the one with the maximal classification score.

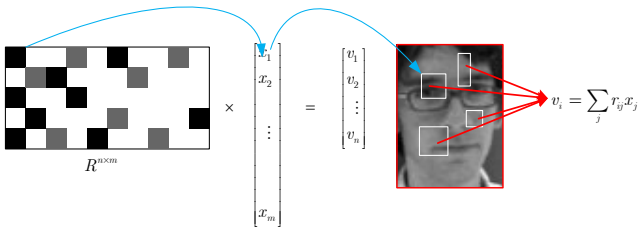


Fig. 2. Graphical representation of compressing a high-dimensional vector $\mathbf{x}$ to a low-dimensional vector $\mathbf{v}$. In the matrix R, dark, gray and white rectangles represent negative, positive, and zero entries, respectively. The blue arrows illustrate that one of nonzero entries of one row of R sensing an element in $\mathbf{x}$ is equivalent to a rectangle filter convolving the intensity at a fixed position of an input image.

3.1 Efficient dimensionality reduction

For each sample $\mathbf{z} \in \mathbb{R}^{\underline{w}\times\underline{h}}$, to deal with the scale problem, we represent it by convolving $\mathbf{z}$ with a set of rectangle filters at multiple scales $\{h_{1,1}, \ldots, h_{\underline{w},\underline{h}}\}$ defined as

$$h_{i,j}(x,y) = \begin{cases} 1, & 1 \leq x \leq i,\ 1 \leq y \leq j \\ 0, & \text{otherwise} \end{cases} \tag{3}$$

where i and j are the width and height of a rectangle filter, respectively. Then, we represent each filtered image as a column vector in $\mathbb{R}^{\underline{wh}}$ and then concatenate these vectors as a very high-dimensional multi-scale image feature vector $\mathbf{x} = (x_1, ..., x_m)^\top \in \mathbb{R}^m$ where $m = (\underline{wh})^2$. The dimensionality m is typically in the order of 10^6 to 10^{10}. We adopt a sparse random matrix R in (2) with $s = m/4$ to project $\mathbf{x}$ onto a vector $\mathbf{v} \in \mathbb{R}^n$ in a low-dimensional space. The random matrix R needs to be computed only once off-line and remains fixed throughout the tracking process. For the sparse matrix R in (2), the computational load is very light. As shown by Figure 2, we only need to store the nonzero entries in R and the positions of rectangle filters in an input image corresponding to the nonzero entries in each row of R. Then, $\mathbf{v}$ can be efficiently computed by using R to sparsely measure the rectangular features which can be efficiently computed using the integral image method [20].

3.2 Analysis of low-dimensional compressive features

As shown in Figure 2, each element v_i in the low-dimensional feature $\mathbf{v} \in \mathbb{R}^n$ is a linear combination of spatially distributed rectangle features at different scales. As the coefficients in the measurement matrix can be positive or negative (via (2)), the compressive features compute the relative intensity difference in a way similar to the generalized Haar-like features [8] (See also Figure 2). The Haar-like features have been widely used for object detection with demonstrated success [20, 21, 8]. The basic types of these Haar-like features are typically

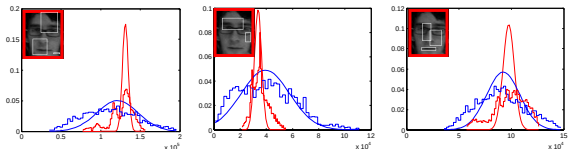


Fig. 3. Probability distributions of three different features in a low-dimensional space. The red stair represents the histogram of positive samples while the blue one represents the histogram of negative samples. The red and blue lines denote the corresponding estimated distributions by our incremental update method.

designed for different tasks [20, 21]. There often exist a very large number of Haar-like features which make the computational load very heavy. This problem is alleviated by boosting algorithms for selecting important features [20, 21]. Recently, Babenko et al. [8] adopted the generalized Haar-like features where each one is a linear combination of randomly generated rectangle features, and use online boosting to select a small set of them for object tracking. In our work, the large set of Haar-like features are compressively sensed with a very sparse measurement matrix. The compressive sensing theories ensure that the extracted features of our algorithm preserve almost all the information of the original image. Therefore, we can classify the projected features in the compressed domain efficiently without curse of dimensionality.

3.3 Classifier construction and update

For each sample $\mathbf{z} \in \mathbb{R}^m$, its low-dimensional representation is $\mathbf{v} = (v_1, \ldots, v_n)^\top \in \mathbb{R}^n$ with $m \gg n$. We assume all elements in $\mathbf{v}$ are independently distributed and model them with a naive Bayes classifier [22],

$$H(\mathbf{v}) = \log\left(\frac{\prod_{i=1}^{n} p(v_i|y=1)p(y=1)}{\prod_{i=1}^{n} p(v_i|y=0)p(y=0)}\right) = \sum_{i=1}^{n} \log\left(\frac{p(v_i|y=1)}{p(v_i|y=0)}\right), \quad (4)$$

where we assume uniform prior, $p(y=1) = p(y=0)$, and $y \in \{0, 1\}$ is a binary variable which represents the sample label.

Diaconis and Freedman [23] showed that the random projections of high dimensional random vectors are almost always Gaussian. Thus, the conditional distributions $p(v_i|y=1)$ and $p(v_i|y=0)$ in the classifier $H(\mathbf{v})$ are assumed to be Gaussian distributed with four parameters $(\mu_i^1, \sigma_i^1, \mu_i^0, \sigma_i^0)$ where

$$p(v_i|y=1) \sim N(\mu_i^1, \sigma_i^1), \qquad p(v_i|y=0) \sim N(\mu_i^0, \sigma_i^0). \quad (5)$$

The scalar parameters in (5) are incrementally updated

$$\begin{aligned} \mu_i^1 &\leftarrow \lambda\mu_i^1 + (1-\lambda)\mu^1 \\ \sigma_i^1 &\leftarrow \sqrt{\lambda(\sigma_i^1)^2 + (1-\lambda)(\sigma^1)^2 + \lambda(1-\lambda)(\mu_i^1 - \mu^1)^2}, \end{aligned} \quad (6)$$

where $\lambda > 0$ is a learning parameter, $\sigma^1 = \sqrt{\frac{1}{n}\sum_{k=0|y=1}^{n-1}(v_i(k)-\mu^1)^2}$ and $\mu^1 = \frac{1}{n}\sum_{k=0|y=1}^{n-1} v_i(k)$. The above equations can be easily derived by maximal likelihood estimation. Figure 3 shows the probability distributions for three different features of the positive and negative samples cropped from a few frames of a sequence for clarity of presentation. It shows that a Gaussian distribution with online update using (6) is a good approximation of the features in the projected space. The main steps of our algorithm are summarized in Algorithm 1.

Algorithm 1 Compressive Tracking

Input: t-th video frame

1. Sample a set of image patches, $D^\gamma = \{\mathbf{z} | ||\mathbf{l}(\mathbf{z}) - \mathbf{l}_{t-1}|| < \gamma\}$ where $\mathbf{l}_{t-1}$ is the tracking location at the (t-1)-th frame, and extract the features with low dimensionality.
2. Use classifier H in (4) to each feature vector $\mathbf{v}(\mathbf{z})$ and find the tracking location $\mathbf{l}_t$ with the maximal classifier response.
3. Sample two sets of image patches $D^\alpha = \{\mathbf{z} | ||\mathbf{l}(\mathbf{z}) - \mathbf{l}_t|| < \alpha\}$ and $D^{\zeta,\beta} = \{\mathbf{z} | \zeta < ||\mathbf{l}(\mathbf{z}) - \mathbf{l}_t|| < \beta\}$ with $\alpha < \zeta < \beta$.
4. Extract the features with these two sets of samples and update the classifier parameters according to (6).

Output: Tracking location $\mathbf{l}_t$ and classifier parameters

3.4 Discussion

We note that simplicity is the prime characteristic of our algorithm in which the proposed sparse measurement matrix R is independent of any training samples, thereby resulting in a very efficient method. In addition, our algorithm achieves robust performance as discussed below.

Difference with related work. It should be noted that our algorithm is different from the recently proposed ℓ_1-tracker [10] and compressive sensing tracker [9]. First, both algorithms are generative models that encode an object sample by sparse representation of templates using ℓ_1-minimization. Thus the training samples cropped from the previous frames are stored and updated, but this is not required in our algorithm due to the use of a data-independent measurement matrix. Second, our algorithm extracts a linear combination of generalized Haar-like features but these trackers [10][9] use the holistic templates for sparse representation which are less robust as demonstrated in our experiments. In [9], an orthogonal matching pursuit algorithm is applied to solve the ℓ_1-minimization problems. Third, both of these tracking algorithms [10][9] need to solve numerous time-consuming ℓ_1-minimization problems but our algorithm is efficient as only matrix multiplications are required.

Random projection vs. principal component analysis. For visual tracking, dimensionality reduction algorithms such as principal component analysis

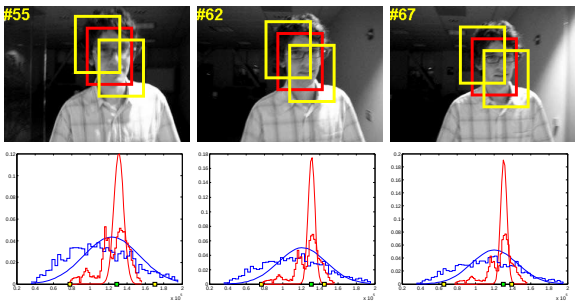


Fig. 4. Illustration of robustness of our algorithm to ambiguity in detection. Top row: three positive samples. The sample in red rectangle is the most "correct" positive sample while other two in yellow rectangles are less "correct" positive samples. Bottom row: the probability distributions for a feature extracted from positive and negative samples. The red markers denote the feature extracted from the most "correct" positive sample while the yellow markers denote the feature extracted from the two less "correct" positive samples. The red and blue stairs as well as lines denote the estimated distributions of positive and negative samples as shown in Figure 3.

and its variations have been widely used in generative tracking methods [1, 6]. These methods need to update the appearance models frequently for robust tracking. However, these methods are sensitive to occlusion due to the holistic representation schemes. Furthermore, it is not clear whether the appearance models can be updated correctly with new observations (e.g., without alignment errors to avoid tracking drift). In contrast, our algorithm does not suffer from the problems with online self-taught learning approaches [24] as the proposed model with the measurement matrix is data-independent. It has been shown that for image and text applications, favorable results can be achieved by methods with random projection than principal component analysis [25].

Robustness to ambiguity in detection. The tracking-by-detection methods often encounter the inherent ambiguity problems as shown in Figure 4. Recently Babenko et al. [8] introduced multiple instance learning schemes to alleviate the tracking ambiguity problem. Our algorithm is robust to the ambiguity problem as illustrated in Figure 4. While the target appearance changes over time, the most "correct" positive samples (e.g., the sample in the red rectangle in Figure 4) are similar in most frames. However, the less "correct" positive samples (e.g., samples in yellow rectangles of Figure 4) are much more different as they involve some background information which vary much more than those within the target object. Thus, the distributions for the features extracted from the most "correct" positive samples are more concentrated than those from the less "correct" positive samples. This in turn makes the features from the most "correct" positive samples much more stable than those from the less "correct" positive samples (e.g., bottom row in Figure 4, the features denoted by red markers are more stable than those denoted by yellow markers). Thus, our algorithm is able to select the most "correct" positive sample because its probability is larger than

those of the less "correct" positive samples (See the markers in Figure 4). In addition, our measurement matrix is data-independent and no noise is introduced by mis-aligned samples.

Robustness to occlusion. Each feature in our algorithm is spatially localized (Figure 2) which is less sensitive to occlusion than holistic representations. Similar representations, e.g., local binary patterns [26] and generalized Haar-like features [8], have been shown to be more effective in handling occlusion. Moreover, features are randomly sampled at multiple scales by our algorithm in a way similar to [27, 8] which have demonstrated robust results for dealing with occlusion.

Dimensionality of projected space. Assume there exist d input points in $\mathbb{R}^m$. Given $0 < \epsilon < 1$ as well as $\beta > 0$, and let $R \in \mathbb{R}^{n\times m}$ be a random matrix projecting data from $\mathbb{R}^m$ to $\mathbb{R}^n$, the theoretical bound for the dimension n that satisfies the Johnson-Lindenstrauss lemma is $n \geq \left(\frac{4+2\beta}{\epsilon^2/2-\epsilon^3/3}\right)\ln(d)$ [16]. In practice, Bingham and Mannila [25] pointed out that this bound is much higher than that suffices to achieve good results on image and text data. In their applications, the lower bound for n when $\epsilon = 0.2$ is 1600 but $n = 50$ is sufficient to generate good results. In our experiments, with 100 samples (i.e., $d = 100$), $\epsilon = 0.2$ and $\beta = 1$, the lower bound for n is approximately 1600. Another bound derived from the restricted isometry property in compressive sensing [15] is much tighter than that from Johnson-Lindenstrauss lemma, where $n \geq \kappa\beta\log(m/\beta)$ and κ and β are constants. For $m = 10^6, \kappa = 1$, and $\beta = 10$, it is expected that $n \geq 50$. We find that good results can be obtained when $n = 50$ in our experiments.

4 Experiments

We evaluate our tracking algorithm with 7 state-or-the-art methods on 20 challenging sequences among which 16 are publicly available and 4 are our own. The *Animal*, *Shaking* and *Soccer* sequences are provided in [28] and the *Box* and *Jumping* are from [29]. The 7 trackers we compare with are the fragment tracker (Frag) [30], the online AdaBoost method (OAB) [5], the Semi-supervised tracker (SemiB) [7], the MILTrack algorithm [8], the ℓ_1-tracker [10], the TLD tracker [11], and the Struck method [31]. We note that the source code of [9] is not available for evaluation and the implementation requires some technical details and parameters not discussed therein. It is worth noticing that we use the most challenging sequences from the existing works. For fair comparison, we use the source or binary codes provided by the authors with tuned parameters for best performance. For our compared trackers, we either use the tuned parameters from the source codes or empirically set them for best results. Since all of the trackers except for Frag involve randomness, we run them 10 times and report the average result for each video clip. Our tracker is implemented in MATLAB, which runs at 35 frames per second (FPS) on a Pentium Dual-Core 2.80 GHz CPU with 4 GB RAM. The source codes and datasets are available at `http://www4.comp.polyu.edu.hk/~cslzhang/CT/CT.htm`.

Table 1. Success rate (SR)(%). **Bold** fonts indicate the best performance while the *italic* fonts indicate the second best ones. The total number of evaluated frames is 8270.

Sequence	CT	MILTrack	OAB	SemiB	Frag	ℓ_1-track	TLD	Struck
Animal	*76*	73	15	47	3	5	75	**97**
Bolt	*79*	**83**	1	16	39	2	1	10
Biker	**75**	21	42	*62*	26	31	42	35
Box	*89*	65	13	38	16	5	**92**	**92**
Coupon book	**100**	*99*	98	37	27	**100**	16	*99*
Cliff bar	**89**	65	23	65	22	38	67	*70*
David indoor	*89*	68	31	46	8	41	**98**	**98**
Girl	78	50	71	50	68	*90*	57	**99**
Jumping	**100**	*99*	86	84	36	9	*99*	18
Kitesurf	*68*	**90**	31	67	10	31	65	40
Occluded face 2	**100**	*99*	47	40	52	84	46	78
Panda	**81**	*75*	69	67	7	56	29	13
Sylvester	75	80	70	68	34	46	**94**	*87*
Skiing	*70*	42	69	69	7	10	59	**80**
Shaking	**92**	*85*	40	31	28	10	16	1
Soccer	**78**	17	8	9	*27*	13	10	14
Twinings	*89*	72	**98**	23	69	83	46	**98**
Tiger 1	**78**	39	24	28	19	13	65	*73*
Tiger 2	**60**	*45*	37	17	13	12	41	22
Walking person	89	32	86	81	32	*98*	55	**100**
Average SR	**84**	*69*	52	48	31	46	49	64

4.1 Experimental setup

Given a target location at the current frame, the search radius for drawing positive samples is set to $\alpha = 4$ which generates 45 positive samples. The inner and outer radii for the set $X^{\zeta,\beta}$ that generates negative samples are set to $\zeta = 8$ and $\beta = 30$, respectively. We randomly select 50 negative samples from set $X^{\zeta,\beta}$. The search radius for set D^{γ} to detect the object location is set to $\gamma = 20$ and about 1100 samples are generated. The dimensionality of projected space is set to $n = 50$, and the learning parameter λ is set to 0.85.

4.2 Experimental results

All of the video frames are in gray scale and we use two metrics to evaluate the proposed algorithm with 7 state-of-the-art trackers. The first metric is the success rate, $score = \frac{area(ROI_T \bigcap ROI_G)}{area(ROI_T \bigcup ROI_G)}$, where ROI_T is the tracking bounding box and ROI_G is the ground truth bounding box. If the *score* is larger than 0.5 in one frame, the tracking result is considered as a success. The other is the center location error measured with manually labeled ground truth data. Table 1 and Table 2 show the quantitative results averaged over 10 times as described above. We note that although TLD tracker is able to relocate on the target during tracking, it is easy to lose the target completely for some frames in most of the test sequences. Thus, we only show the center location errors for the sequences that TLD can keep track all the time. The proposed compressive tracking algorithm achieves the best or second best results in most sequences in terms of both success rate and center location error. Furthermore, our tracker runs faster than all the other algorithms although they (except for the TLD method and ℓ_1-tracker) are implemented in C or C++ which is intrinsically more efficient than MATLAB. Figure 5 shows screenshots of some tracking results.

Scale, pose and illumination change. For the *David indoor* sequence shown in Figure 5(a), the illumination and pose of the object both change gradually.

Table 2. Center location error (CLE)(in pixels) and average frame per second (FPS). **Bold** fonts indicate the best performance while the *italic* fonts indicate the second best ones. The total number of evaluated frames is 8270.

Sequence	CT	MILTrack	OAB	SemiB	Frag	ℓ_1-track	TLD	Struck
Animal	**17**	32	62	*25*	99	155	−−	**17**
Bolt	**8**	**8**	227	101	*43*	58	−−	157
Biker	**11**	38	*12*	13	72	74	−−	14
Box	*14*	57	74	119	160	196	−−	**11**
Coupon book	**4**	*6*	9	74	63	*6*	−−	10
Cliff bar	**7**	*14*	33	56	34	35	−−	20
David indoor	16	19	57	37	73	42	*12*	**9**
Girl	21	25	23	50	26	*13*	−−	**10**
Jumping	**6**	10	11	11	29	99	*8*	42
Kitesurf	*9*	**5**	33	10	55	51	−−	30
Occluded face 2	**10**	*16*	36	39	58	19	−−	25
Panda	**6**	*9*	10	10	69	10	20	67
Sylvester	*9*	10	12	14	47	42	**7**	*9*
Skiing	**10**	15	12	*11*	134	189	−−	166
Shaking	**9**	*12*	22	133	41	192	−−	166
Soccer	**16**	64	96	135	*54*	189	−−	95
Twinings	*9*	14	**7**	70	15	10	15	**7**
Tiger 1	**10**	27	42	39	39	48	−−	*12*
Tiger 2	**13**	*18*	22	29	37	57	−−	22
Walking person	5	8	5	7	59	*4*	6	**2**
Average CLE	**10**	*19*	36	49	56	60	−−	36
Average FPS	**35.2**	*10.5*	8.6	6.7	3.8	0.1	9.6	0.01

The MILTrack, TLD and Struck methods perform well on this sequence. For the *Shaking* sequence shown in Figure 5(b), when the stage light changes drastically and the pose of the subject changes rapidly as he performs, all the other trackers fail to track the object reliably. The proposed tracker is robust to pose and illumination changes as object appearance can be modeled well by random projections (based on the Johnson-Lindenstrauss lemma) and the classifier with online update is used to separate foreground and background samples. Moreover, the proposed tracker is a discriminative model with local features that has been demonstrated to handle pose variation well (e.g., MILTrack [8]). The generative subspace tracker (e.g., IVT [6]) has been shown to be effective in dealing with large illumination changes while the discriminative tracking method with local features (i.e., MILTrack [8]) has been demonstrated to handle pose variation adequately. Furthermore, the features we use are similar to generalized Haar-like features which have been shown to be robust to scale and orientation change [8] as illustrated in the *David indoor* sequence. In addition, our tracker performs well on the *Sylvester* and *Panda* sequences in which the target objects undergo significant pose changes (See the supplementary material for details).

Occlusion and pose variation. The target object in *Occluded face 2* sequence in Figure 5(c) undergoes large pose variation and heavy occlusion. Only the MILTrack and Struck methods as well as the proposed algorithm perform well on this sequence. In addition, our tracker achieves the best performance in terms of success rate, center location error, and frame rate. The target player in *Soccer* sequence is heavily occluded by others many times when he is holding up the trophy as shown in Figure 5(d). In some frames, the object is almost fully occluded (e.g., #120). Moreover, the object undergoes drastic motion blur and illumination change (#70, and #300). All the other trackers lose track of the targets in numerous frames. Due to drastic scene change, it is unlikely that online appearance models are able to adapt fast and correctly. Our tracker can handle

Fig. 5. Screenshots of some sampled tracking results.

occlusion and pose variations well as its appearance model is discriminatively learned from target and background with a data-independent measurement, thereby alleviating the influence from background (See also Figure 4). Furthermore, our tracker performs well for objects with non-rigid pose variation and camera view change in the *Bolt* sequence (Figure 5(j)) because the appearance model of our tracker is based on local features which are insensitive to non-rigid shape deformation.

Out of plane rotation and abrupt motion. The object in the *Kitesurf* sequence (Figure 5(e)) undergoes acrobat movements with 360 degrees out of plane rotation. Only the MILTrack, SemiB and the proposed trackers perform well on this sequence. Both our tracker and the MILTrack method are designed to handle object location ambiguity in tracking with classifiers and discriminative features. The object in the *Animal* sequence (Figure 5(f)) exhibits abrupt motion. Both the Struck and the proposed methods perform well on this sequence. However, when the out of plane rotation and abrupt motion both occur in the *Biker* and *Tiger 2* sequences (Figure 5(g),(h)), all the other algorithms fail to track the target objects well. Our tracker outperforms the other methods in all the

metrics (accuracy, success rate and speed). The *Kitesurf*, *Skiing*, *Twinings*, *Girl* and *Tiger 1* sequences all contain out of plane rotation while *Jumping* and *Box* sequences include abrupt motion. Similarly, our tracker performs well in terms of all metrics.

Background clutters. The object in the *Cliff bar* sequence changes in scale, orientation and the surrounding background has similar texture. As the ℓ_1-tracker uses a generative appearance model that does not take background information into account, it is difficult to keep track of the objects correctly. The object in the *Coupon book* sequence undergoes significant appearance change at the 60-th frame, and then the other coupon book appears. Both the Frag and SemiB methods are distracted to track the other coupon book (#230 in Figure 5(i)) while our tracker successfully tracks the correct one. Because the TLD tracker relies heavily on the visual information in the first frame to re-detect the object, it also suffers from the same problem. Our algorithm is able to track the right objects accurately in these two sequences because it extracts discriminative features for the most "correct" positive sample (i.e., the target object) online (See Figure 4) with classifier update for foreground/background separation.

5 Concluding Remarks

In this paper, we proposed a simple yet robust tracking algorithm with an appearance model based on non-adaptive random projections that preserve the structure of original image space. A very sparse measurement matrix was adopted to efficiently compress features from the foreground targets and background ones. The tracking task was formulated as a binary classification problem with online update in the compressed domain. Our algorithm combines the merits of generative and discriminative appearance models to account for scene changes. Numerous experiments with state-of-the-art algorithms on challenging sequences demonstrated that the proposed algorithm performs well in terms of accuracy, robustness, and speed.

References

1. Black, M., Jepson, A.: Eigentracking: Robust matching and tracking of articulated objects using a view-based representation. IJCV **38** (1998) 63–84
2. Jepson, A., Fleet, D., Maraghi, T.: Robust online appearance models for visual tracking. PAMI **25** (2003) 1296–1311
3. Avidan, S.: Support vector tracking. PAMI **26** (2004) 1064–1072
4. Collins, R., Liu, Y., Leordeanu, M.: Online selection of discriminative tracking features. PAMI **27** (2005) 1631–1643
5. Grabner, H., Grabner, M., Bischof, H.: Real-time tracking via online boosting. In BMVC (2006) 47–56
6. Ross, D., Lim, J., Lin, R., Yang, M.H.: Incremental learning for robust visual tracking. IJCV **77** (2008) 125–141
7. Grabner, H., Leistner, C., Bischof, H.: Semi-supervised on-line boosting for robust tracking. In ECCV (2008) 234–247

8. Babenko, B., Yang, M.H., Belongie, S.: Robust object tracking with online multiple instance learning. PAMI **33** (2011) 1619–1632
9. Li, H., Shen, C., Shi, Q.: Real-time visual tracking using compressive sensing. In CVPR (2011) 1305–1312
10. Mei, X., Ling, H.: Robust visual tracking and vehicle classification via sparse representation. PAMI **33** (2011) 2259–2272
11. Kalal, Z., Matas, J., Mikolajczyk, K.: P-n learning: bootstrapping binary classifier by structural constraints. In CVPR (2010) 49–56
12. Donoho, D.: Compressed sensing. IEEE Trans. Inform. Theory **52** (2006) 1289–1306
13. Candes, E., Tao, T.: Near optimal signal recovery from random projections and universal encoding strategies. IEEE Trans. Inform. Theory **52** (2006) 5406–5425
14. Wright, J., Yang, A., Ganesh, A., Sastry, S., Ma, Y.: Robust face recognition via sparse representation. PAMI **31** (2009) 210–227
15. Candes, E., Tao, T.: Decoding by linear programing. IEEE Trans. Inform. Theory **51** (2005) 4203–4215
16. Achlioptas, D.: Database-friendly random projections: Johnson-Lindenstrauss with binary coins. J. Comput. Syst. Sci **66** (2003) 671–687
17. Baraniuk, R., Davenport, M., DeVore, R., Wakin, M.: A simple proof of the restricted isometry property for random matrices. Constr. Approx **28** (2008) 253–263
18. Liu, L., Fieguth, P.: Texture classification from random features. PAMI **34** (2012) 574–586
19. Li, P., Hastie, T., Church, K.: Very sparse random projections. In KDD (2006) 287–296
20. Viola, P., Jones, M.: Rapid object detection using a boosted cascade of simple features. In CVPR (2001) 511–518
21. Li, S., Zhang, Z.: Floatboost learning and statistical face detection. PAMI **26** (2004) 1–12
22. Ng, A., Jordan, M.: On discriminative vs. generative classifier: a comparison of logistic regression and naive bayes. In NIPS (2002) 841–848
23. Diaconis, P., Freedman, D.: Asymptotics of graphical projection pursuit. Ann. Stat **12** (1984) 228–235
24. Raina, R., Battle, A., Lee, H., Packer, B., Ng, A.Y.: Self-taught learning: Transfer learning from unlabeled data. In ICML (2007)
25. Bingham, E., Mannila, H.: Random projection in dimensionality reduction: Applications to image and text data. In KDD (2001) 245–250
26. Ahonen, T., Hadid, A., Pietikainen, M.: Face description with local binary patterns: application to face recogntion. PAMI **28** (2006) 2037–2041
27. Leonardis, A., Bischof, H.: Robust recogtion using eigenimages. CVIU **78** (2000) 99–118
28. Kwon, J., Lee, K.: Visual tracking decomposition. In CVPR (2010) 1269–1276
29. Santner, J., Leistner, C., Saffari, A., Pock, T., Bischof, H.: PROST Parallel Robust Online Simple Tracking. In CVPR (2010)
30. Adam, A., Rivlin, E., Shimshoni, I.: Robust fragements-based tracking using the integral histogram. In CVPR (2006) 798–805
31. Hare, S., Saffari, A., Torr, P.: Struck: structured output tracking with kernels. In ICCV (2011) 263–270