2016 23rd International Conference on Pattern Recognition (ICPR)
Cancún Center, Cancún, México, December 4-8, 2016

Adaptive and Compressive Target Tracking Based on Feature Point Matching

Fengjiao Li and Yuanyuan Zhang
Information Research Institute of Shandong Academy of Sciences
Jinan, China
Email: {lifj, zhangyy}@sdas.org

Wei Qi Yan and Reinhard Klette
Auckland University of Technology
Auckland, New Zealand
Email: {wyan, rklette}@aut.ac.nz

***Abstract*—In compressive tracking algorithms, a feature reduction projection matrix is constructed by using compressed sensing theory. Target and non-target objects are discriminated by using a naive Bayesian classifier. Such an algorithm may ensure accuracy of target tracking in real-time. But it is not adaptive for tracking with respect to scales and rotations. In this paper, we propose a novel adaptive algorithm based on feature point matching for tracking objects which appear with various changes. We combine weight-average and improved compressive tracking algorithms together for tracking objects, then calculate the corresponding feature points between two subsequent frames of the same object for obtaining the target changes related to various scales and rotations. Our experimental results show that the improved algorithm effectively improves the accuracy of target tracking and ensures adaptability of the tracking algorithm.**

I. Introduction

Target tracking technologies have manifold applications, for example in intelligent video surveillance, traffic systems, or mobile robotics. They are mainly employed to locate and understand targets of interest in video data for quantitative analysis. Object tracking is a fundamental step of target analysis. In the past decades, object tracking has been an extremely important research topic; for example, see [1]–[3] in computer vision.

Over years, significant progress has been achieved in target tracking. For instance, meanshift [4] has been typically applied to object tracking: the kernel density of a reference model and a candidate model (in a specific feature space) are employed, and matching scores of reference model and candidate model are compared. That candidate model with the maximum matching score was identified with the final position of the target.

Because the meanshift algorithm is not adaptive to scale changes of the target, Bradski [5] proposed the camshift algorithm, Zhang [7] presented a fast object tracking algorithm with *spatio-temporal context* [6] (STC) which extracted a model of the target position and its regional context based on Bayesian theorem according to the statistical correlation between the target and its context. Additionally it applied a confidence map to determine the final position of the target.

A tracking algorithm based on consistency of feature point matching is proposed in [8] which can process hundreds of frames per second. It combines pyramidal optical flow tracking and feature point matching to determine the position of the target preliminarily, and then assigns to each feature point a vote. Those feature points which satisfy the consistency using a clustering algorithm are selected as the target position.

A kernel-related filter [10] leads to a fast object tracking algorithm; it is based on target detection [9] which extracts features of positive and negative samples by the circular shift method, and combines those by using the fast Fourier transform for training the classifier (for locating a target).

In recent deep learning, *Convolutional Neural Networks* (CNNs) achieve remarkable results for target tracking [11]–[13]. Due to the high computational complexity of neural networks, there are still real-time issues for these algorithms to be resolved.

Kalal [14] proposed a *long-term stable tracking* (TLD) algorithm which combined tracker and detector effectively. A compressive tracking algorithm [15] selects a proper feature space to establish a target model, and then constructs a classifier to discriminate the target and the background. This method combines generative models and discriminative models together effectively [16], [17].

Original compressive tracking cannot deal with changes in scale and rotation. For solving this problem, there are various improvements [18]–[20]. For example, Zhang [18] uses structure constraints for the target to ensure multiscale samples. In [19], Affine transforms of samples were adopted to obtain different scales and rotations.

Different to existing work, in order to achieve the real-time effect (i.e. to reduce the computational complexity), in this paper we use feature points to obtain scaling and rotation factors of tracking targets. To the best of our knowledge, this is the first time to improve the adaptability of a compressive target tracking algorithm. We implemented the improved algorithm and report our promising experimental results.

The paper is structured as follows. Section II outlines compressive tracking. Section III provides our novel additions to the approach. Our work in this paper is discussed in Section IV. Section V concludes.

II. Compressive Tracking

A. Feature Extraction

Feature extraction requires convolutions using multiscale rectangular filters designed by using positive and negative

978-1-5090-4846-5/16/$31.00 ©2016 IEEE

samples. Based on the convolutions, we obtain feature vectors of different lengths. These vectors are concatenated to form a high-dimensional vector, named as the *vector of a multiscale rectangular feature*. For a sample: $\mathbf{z} \in \mathbb{R}^{w*h}$ the multiscale rectangular filter is a set of functions $\{h_{1,1}, h_{1,2}, ..., h_{w,h}\}$, defined as

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

where $0 < i \leq w$ and $0 < j \leq h$ represent the width and height of the rectangular filter, respectively, and x and y are the pixel coordinates in a multiscale rectangular filter. Then, we represent each filtered image as a column vector $\mathbf{X}$ in a very high-dimensional image space $\mathbb{R}^m$. The dimensionality m is typically in the order of 10^6 to 10^{10}.

In order to ensure real-time tracking, compressive sensing theory [21] was used to reduce the dimension of the image vectors $\mathbf{X}$. The compressive sensing theory [21] analyses mappings of signals $\mathbf{X}$ by a special projection matrix $\mathbf{P}$ for redundancy reduction. The result $\mathbf{V} = \mathbf{PX}$ maps a high-dimensional vector from image space into a low-dimensional (say, n-dimensional) space of feature vectors, with assuming that $n << m$. The original signal $\mathbf{X}$ can be reconstructed from $\mathbf{V}$ with very high probability [22].

The random projection matrix $\mathbf{P} = [p_{ij}]$ is as follows:

$$p_{i,j} = \sqrt{s} \times \begin{cases} 1 & \text{with probability } \frac{1}{2s} \\ 0 & \text{with probability } 1 - \frac{1}{s} \\ -1 & \text{with probability } \frac{1}{2s} \end{cases} \quad (2)$$

where s is often recommended to be equal to 2 or 3. The matrix $\mathbf{P}$ is very sparse and satisfies the *restricted isometry property* (RIP). The literature also discusses $s = m/4$ which means that we only have four non-zero numbers at most in the projection matrix $\mathbf{P}$; thus this greatly reduces the computational complexity.

B. Design of the Classifier

Compressive target tracking adopts a naive Bayesian classifier which is based on the assumption that elements of the feature vector are independent on the sense of a normal distribution. The specific classifier is modeled in [14] as follows:

$$\begin{aligned} H(V) &= \log(\frac{\prod_{i=1}^{n} p(v_i|y=1)p(y=1)}{\prod_{i=1}^{n} p(v_i|y=0)p(y=0)}) \\ &= \sum_{i=1}^{n} \log(\frac{p(v_i|y=1)}{p(v_i|y=0}) \end{aligned} \quad (3)$$

where y represents a sample label, and $p(y=1) = p(y=0) = 0.5$ are the prior probabilities of positive and negative sample, respectively. v_i represents the i-th detected sample feature, then the current target position was the sample with the biggest $H(V)$.

The random projection of a high-dimensional random vector almost satisfies a Gaussian distribution $p(v_i|y=1) \sim N(\mu_i^1, \delta_i^1)$, $p(v_i|y=0) \sim N(\mu_i^0, \delta_i^0)$. Therefore, the conditional probability of the positive and negative samples is also consistent with the Gaussian distribution. The parameters of these Gaussian distributions are constantly updated as follows:

$$\begin{aligned} \mu_i^1 &= \lambda\mu_i^1 + (1-\lambda)\mu^1 \\ \delta_i^1 &= \sqrt{\lambda(\delta_i^1)^2 + (1-\lambda)(\delta^1)^2 + \lambda(1-\lambda)(\mu_i^1 - \mu^1)^2} \end{aligned} \quad (4)$$

where μ^1 and δ^1 represent mean and variance of the dimension-reduced feature vectors of the i-th sample, respectively, and λ is a learning factor. μ_i^1 and δ_i^1 are the updated mean and variance.

Finally, that object region (of rectangular shape) having the biggest classification score is the detected position of the target.

III. Scale and Rotation-based Adaptive Compressive Tracking

The common sampling method of compressive tracking is presented in the following equations:

$$R_p = \{\gamma_{1p} < \| R_p - R_{t-1} \| < \gamma_{2p}\} \quad (5)$$

$$R_n = \{\| R_n - R_{t-1} \| > \gamma_n\} \quad (6)$$

$$R_d = \{\| R_d - R_{t-1} \| < \gamma_d\} \quad (7)$$

where R_p, R_n, and R_d represent the sets of positive samples, negative samples, and detected samples, respectively. γ_{1p}, γ_{2p}, γ_n, and γ_d are thresholds for sample collection given based on observations. R_{t-1} is the target position of last frame, we can see that all samples can be obtained based on it. The collected samples have only slight changes related to locations; the sample sizes are always identical. Therefore, compressive tracking cannot track targets with scale or rotation changes.

In compressive target tracking, drift errors are usually occurring in cases of intense movement, occlusion, or similarity in color or silhouettes (i.e. appearance). In compressive tracking, selecting the sample having the highest score as the target location is improper sometimes, although the sample might be more similar to the target than others, least resembling to the shown background, the corresponding region of this sample cannot reflect the exact location of the target.

Therefore, in this paper we propose a target localization method based on averaging weighted multi-rectangles [23]. We compute the average of weights as the scores. We regard the weighted average of the top $N > 0$ rectangular regions G_i as the candidate for the target location:

$$G_o = \frac{1}{N} \sum_{i=1}^{N} \omega_i \cdot G_i \quad (8)$$

where G_o is the final target position, and G_i is the location of the i-th sample amongst all the selected samples; ω_i represents the weight of the i-th sample. ω_i is defined as follows:

$$\omega_i = \begin{cases} \frac{H(V_{G_i})}{\overline{H(V)}} & H(V_{G_i}) > \overline{H(V)} \\ \frac{\overline{H(V)}}{H(V_{G_i})} & H(V_{G_i}) < \overline{H(V)} \end{cases} \quad (9)$$

where $\overline{H(V)}$ is the average score of the top N rectangular regions, $H(V_{G_i})$ represents the ith detected sample score.

Applying this method we are not only able to deal with a range of target scale changes in target tracking, we are also able to better exclude the interference by similar objects. We show that this approach offers the possibility of improving the accuracy of object tracking.

We also propose an adaptive compressive tracking algorithm based on feature-point matching. The original algorithm flowchart is shown in Fig. 1.

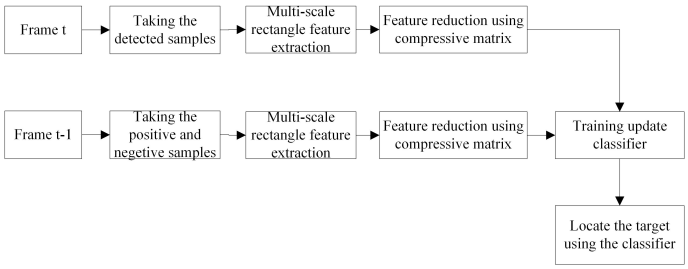


Fig. 1. Flowchart of our compressive tracking

Figure 1 is the flowchart for the initial phase of compressive tracking. According to the position in frame $t-1$, we first collect the positive and negative samples using Eqs. (5) and (6). Then we compute the multiscale rectangular features of the collected samples and reduce the feature dimensions so as to train the classifier. We take the detected samples and the reduced feature vector for frame t in the same way as for frame $t-1$. Finally, we get scores in frame t by using the classifier trained by frame $t-1$ to classify the detected samples. The final location of the target is the weighted average of the top N regions G_i where the corresponding classification scores are ranked.

Algorithm 1 Compute scale and rotation factor

Input: $frame_t$,$frame_{t-1}$, $Rect_t$, $Rect_{t-1}$
Output: α, θ

function GETFACTOR($frame_t, frame_{t-1}, Rect_t, Rect_{t-1}$)
  $ROI_t \leftarrow SetROI(frame_t, Rect_t)$,
  $ROI_{t-1} \leftarrow SetROI(frame_{t-1}, Rect_{t-1})$
  $KP_t \leftarrow detect(ROI_t)$
  $KP_{t-1} \leftarrow detect(ROI_{t-1})$
  $M \leftarrow match(KP_t, KP_{t-1})$
  $num\_point \leftarrow estimate\_num()$
  if $num\_point > T$ **then**
    $\alpha \leftarrow estimate\_scale(M_t, M_{t-1})$
    $\theta \leftarrow estimate\_rotation(M_t, M_{t-1})$
  else
    Return to detect the target
  end if
end function

A. Scale-based Adaptive Target Tracking

We illustrates how to calculate the object scaling and rotation factor in Algorithm 1. We first detect feature points [24], [25] of the target position in frames t and $t-1$, respectively, and match those based on the Euclidean distance [26]. Finally we obtain the scaling factors based on the successfully matched feature points in the considered adjacent frames.

At the same time, we use a trained threshold to ensure a high tracking accuracy rate. If the number of successfully matched points is bigger than the threshold, the tracking result is positive, otherwise we need to relocate the object.

During the process, the feature points of the ROI region are modeled as follows:

$$KP = \{(p_i, d_i)|0 < i \leq num_{point}\} \tag{10}$$

where p_i represents the point plane coordinate, d_i is the descriptor of the feature. Then Euclidean distance is used for descriptors matching:

$$d(d_i, d_j) = \|d_i - d_j\| \tag{11}$$

The computation of a scaling factor is given by the following equation:

$$\alpha_{i,j} = \frac{d_{ij}^t}{d_{i,j}^{t-1}} = \frac{\| p_i^t - p_j^t \|}{\| p_i^{t-1} - p_j^{t-1} \|} \tag{12}$$

where $d_{i,j}^t$ represents the Euclidean distance between pixels p_i^t and p_j^t in frame t.

In the same way, $d_{i,j}^{t-1}$ is the Euclidean distance between pixels p_i^{t-1} and p_j^{t-1} in frame $t-1$. p_i^{t-1} is the matching point to p_i^t in frame $t-1$, so is p_j^t to p_j^{t-1}. We choose the mean or the median of those $\alpha_{i,j}$ values as the final scaling factor α.

B. Rotation-based Adaptive Target Tracking

The angle of rotation is computed as follows:

$$\theta_{i,j} = atan2(y_{i,j}^t, x_{i,j}^t) - atan2(y_{i,j}^{t-1}, x_{i,j}^{t-1}) \tag{13}$$

where $y_{i,j}^t$ and $y_{i,j}^{t-1}$ represent the distance between pixel i and pixel j in y-direction in frame t and $t-1$, respectively. $x_{i,j}^t$ and $x_{i,j}^{t-1}$ are the distances between pixels i and j in x-direction in frames t and $t-1$, respectively. Again we choose either the median or the mean of the $\theta_{i,j}$ values as the final scale factor θ.

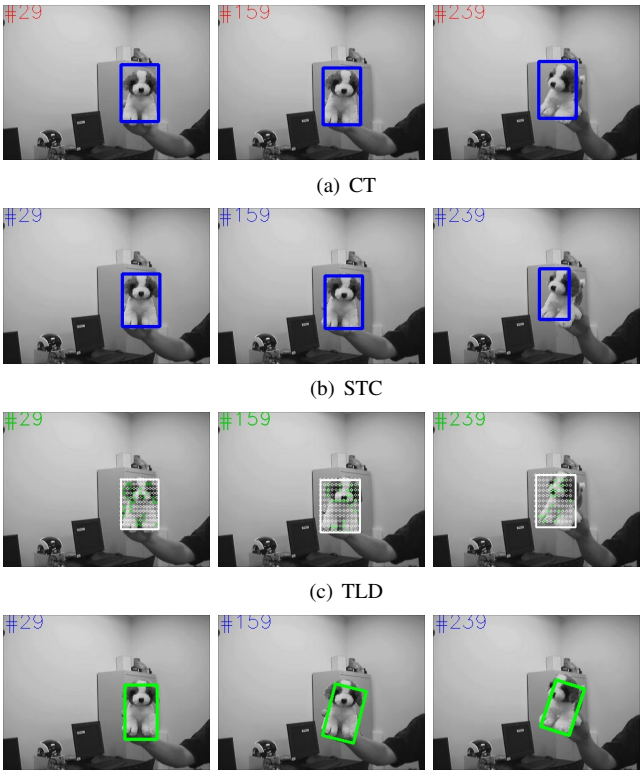
We obtain the 2-dimensional rotation matrix for angle θ:

$$\mathbf{R} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \tag{14}$$

Therefore, the final target position is expressed as follows:

$$G_{tf} = \alpha \cdot \mathbf{R} \cdot G_t \tag{15}$$

where G_{tf} is the final object position; G_t represents the position prior to scaling and rotation.



(a) CT

(b) STC

(c) TLD

(d) Our Algorithm

Fig. 2. dog results

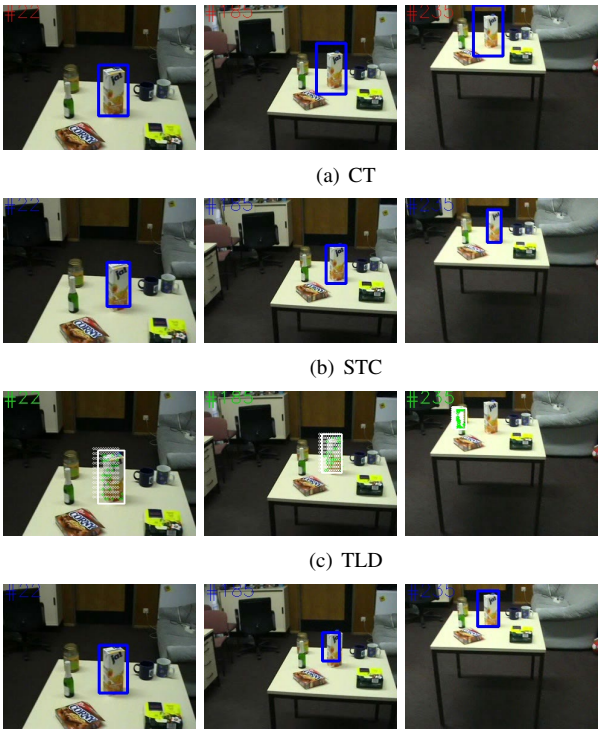
IV. Experimental Results and Analysis

In this paper, we choose seven video sequences and four target tracking algorithms to verify the validity of the proposed algorithm. The video sequences are dog [6], juice [6], David [10], bike [10], dollar [10], toy [6], and mountain-bike [6]. The tracking algorithms are *compressive tracking* (CT), *spatio-temporal context tracking* (STC), *tracking learning detection* (TLD), and our tracking algorithm. The tracking results are shown from Fig. 2 to Fig. 8.

For the experiment with object dog, these four tracking algorithms all track the target successfully, but only the proposed algorithm correctly reflects rotations of the target, as shown in Fig. 2(d). CT did not provide any indication of target scale and rotation, and STC and TLD were able to track the changes with respect to target scale, but not with respect to target rotation. STC depends on the context of objects severely, tracking errors happen at the edge of the object. For example, we can find tracking drifts in frame 239 in Fig. 2(b).

For juice, the STC algorithm is the best both in tracking accuracy and for scaling. It has tracking errors due to the interference of similarity in TLD. Compared with original compressive tracking, our algorithm can track size changes, but there is some slight tracking drift.

Scaling changes for David are not significant, so all the methods can track the position of the target. Although there exists a drift in both CT and our method, but the drift is greatly reduced in our algorithm. This indicates that compressive tracking is dependent on the choice of the initial position. The TLD tracking method, combined with target detection



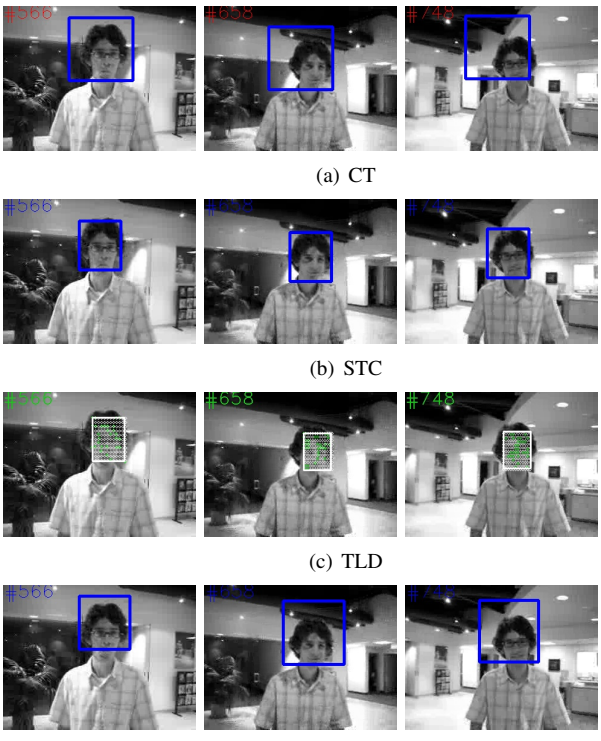
(a) CT

(b) STC

(c) TLD

(d) Our Algorithm

Fig. 3. juice results



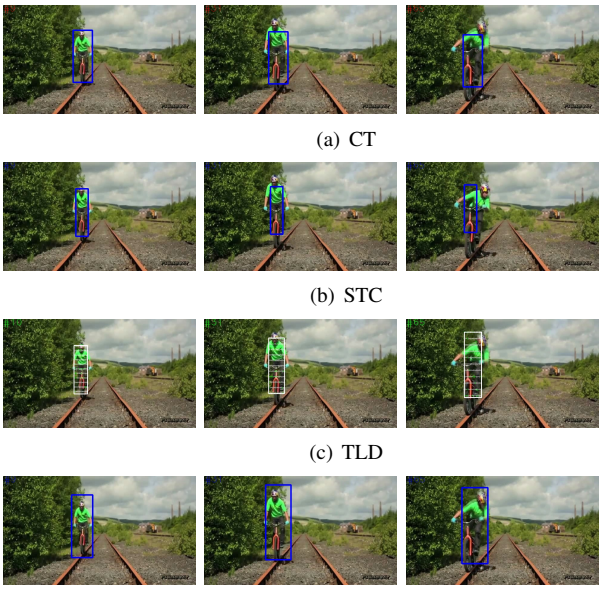
(a) CT

(b) STC

(c) TLD

(d) Our Algorithm

Fig. 4. David results



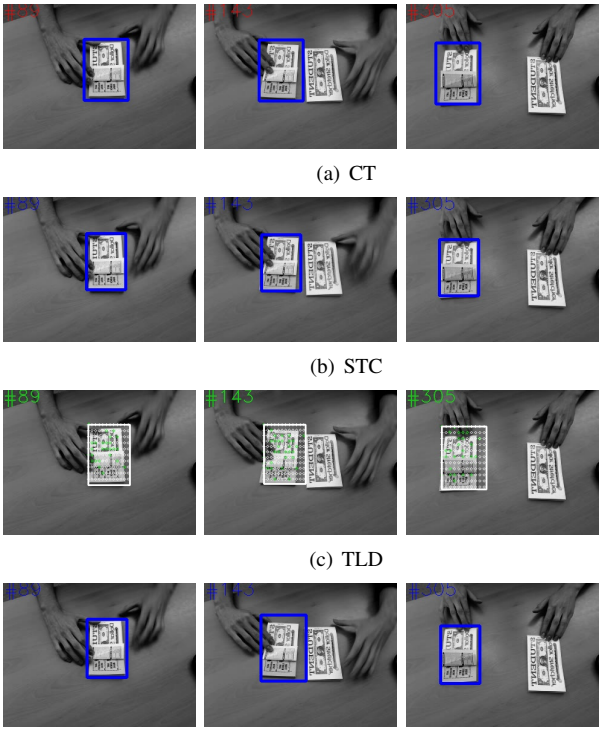
(a) CT

(b) STC

(c) TLD

(d) Our Algorithm

Fig. 5. `bike` results



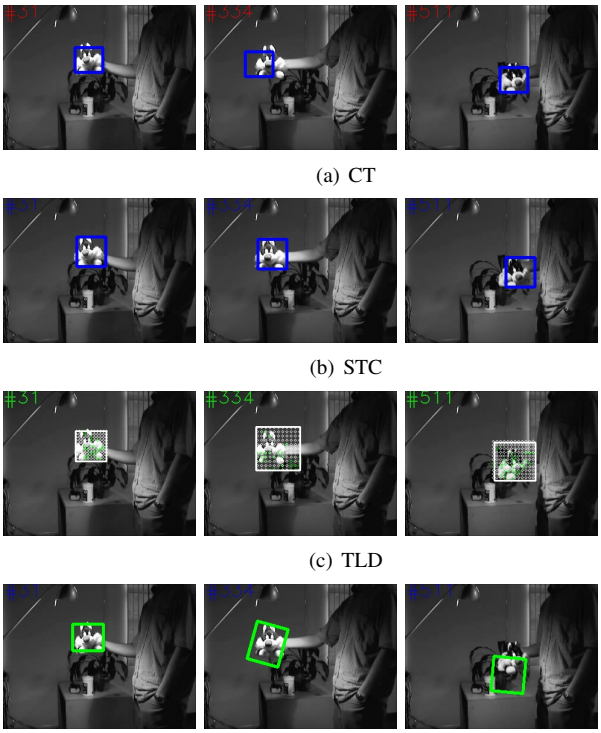
(a) CT

(b) STC

(c) TLD

(d) Our Algorithm

Fig. 6. `dollar` results



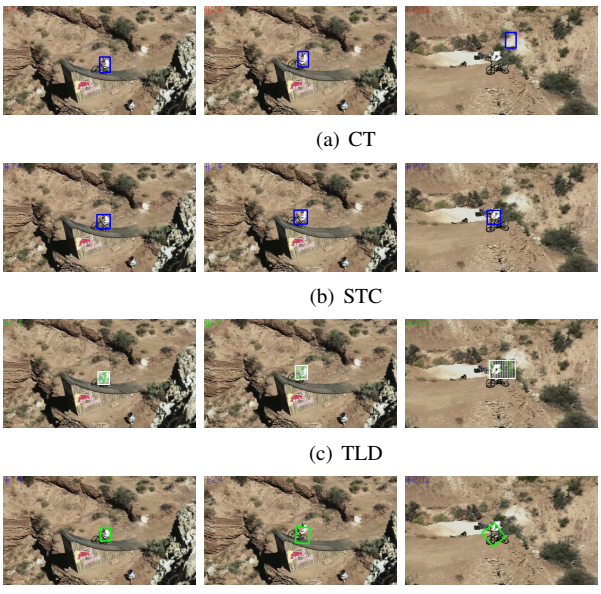
(a) CT

(b) STC

(c) TLD

(d) Our Algorithm

Fig. 7. `toy` results



(a) CT

(b) STC

(c) TLD

(d) Our Algorithm

Fig. 8. `mountain-bike` results

and model learning, has a reduced dependency on the initial position, and STC is much more stable and accurate in tracking compared to TLD.

For `bike`, compared with CT, our algorithm is much more robust. TLD and STC also process scaling changes, but it can be seen from frames 37 and 65 that the results of scale-based adaptive tracking are not ideal, so these two algorithms still have room to be improved.

The four algorithms track the object `dollar` accurately. But, according to our observations, compressive tracking cannot locate the target correctly when the given initial position varies. This indicates that the initial target position plays a

pivotal role in compressive tracking. In our algorithm, the initial target position is given by a number of feature points, which greatly reduces the dependency on the initial position and increases the tracking success rate tremendously.

Figures 7 (`toy`) and 8 (`mountain-bike`) illustrate again rotation-adaptive test results for these two videos. The first column shows initial object positions. The other two columns show tracking results. It can be seen that our algorithm can deal with object rotation changes well for these two videos, in difference to the other three algorithms.

Tracking success rates of the compared four methods, for all the used video test data, are summarised in Table I.

TABLE I
COMPARISON OF TRACKING SUCCESS RATES

Video	CT	STC	TLD	Ours
`dog`	93%	94%	96%	98%
`juice`	84%	91%	95%	88%
`David`	90%	93%	92%	93%
`bike`	73%	75%	85%	90%
`dollar`	69%	93%	90%	93%
`toy`	92%	94%	95%	96%
`mountain-bike`	86%	92%	96%	93%

The algorithm evaluation criteria is defined as:

$$VE = \sqrt{\sum_{i=1}^{4}(p_i - p_g)^2} \tag{16}$$

if $VE < \sqrt{w^2 + h^2}$, it means tracking success. p_i is the plane coordinate of the rectangular's vertex, p_g indicates the ground truth of target position.

V. CONCLUSIONS

The originally proposed compressive tracking algorithms have not the ability to track objects when changing scales or rotating. In order to overcome these drawbacks, we proposed adaptive compressive tracking based on feature point matching. We calculate the changes of scale and rotation by computing ratios and angles of matching feature points between subsequent frames. At the same time, we decide whether the number of successfully matching points satisfy the given thresholds or not, so as to relocate the target. Our experimental results show that the algorithm, proposed in this paper, effectively improves compressive target tracking, and also tracks targets showing changes with respect to scaling or rotation with improved accuracy.

VI. ACKNOWLEDGMENTS

This project was supported by the Natural Science Foundation of Shandong Province China (No. ZR2014YL010), and the Shandong Science and Technology Development Program China (No. 2014GSF120018).

REFERENCES

[1] R. Klette, *Concise Computer Vision*. Springer, London, 2014.
[2] K. Huang, X. Chen, Y. Kang, T. Tan: The techniques of intelligent video surveillance. *J. Computers*, 49(37): 1–27, 2014.
[3] H. Yang, L. Shao, F. Zheng, L. Wang, Z. Song: Recent advances and trends in visual tracking: A review. *Neurocomputing*, 74(18): 3823–3831, 2011.
[4] K. Fukunaga, L.D. Hostetler: The estimation of the gradient of a density function with applications in pattern recognition. *IEEE Trans. Information Theory*, 21(1): 32–40, 1975.
[5] G.B. Bradski: Real time face and object tracking as a component of a perceptual user interface. *IEEE Workshop Applications of Computer Vision*, pp. 214–219, 1998.
[6] G. Nebehay, R. Pflugfelder: Consensus-based matching and tracking of keypoint for object tracking. *IEEE Winter Conference on Applications of Computer Vision*, pp. 862–869, 2014.
[7] K. Zhang, L. Zhang, M.H. Yang, D. Zhang: Fast tracking via spatial temporal context learning. *CVPR*, pp. 1–16, 2013.
[8] Longyin Wen, Zhaowei Cai, Zhen Lei, Dong Yi, Stan Z. Li Online spatio-temporal structural context learning for visual tracking. *ECCV*, pp. 716–729, 2012.
[9] Joo F. Henriques, Rui Caseiro, Pedro Martins, and Jorge Batista: High-speed tracking with kernelized correlation filters. *IEEE Trans. on Pattern Analysis and Machine Intelligence*. 37(3): 583–596, 2015.
[10] A.S. Montero, J. Lang, R. Laganière: Scalable kernel correlation filters with sparse feature integration. *ICCV*, pp. 24-31, 2015.
[11] Hyeonseob Nam and Bohyung Han: Learning multi-domain convolutional neural networks for visual tracking. *CVPR*, pp. 1–10, 2016.
[12] Lijun Wang, Wanli Ouyang Xiaogang Wang, Huchuan Lu: Visual tracking with fully convolutional networks. *ICCV*, pp. 3119–3127, 2015.
[13] S. Cheng, J.-X. Sun, Y.-G. Cao, L.-R. Zhao: Target tracking based on incremental deep learning. *Optics and Precision Engineering*, 23(4): 1161–1170, 2015.
[14] Kaihua Zhang, Lei Zhang, and Ming-Hsuan Yang: Real-time compressive tracking. *ECCV*, pp. 866–879, 2012.
[15] Z. Kalal, K. Mikolajczyk, J. Matas: Tracking-learning-detection. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 6(1): 1–14, 2010.
[16] Qian Liu, Jianhua Hou, Haijun Mou, Wei Zhao, Bangyou Da. Object detection and tracking combining generative and discriminative model. *Journal of Image and Graphics*, 18(10): 1293–1301, 2013.
[17] David A. Ross, Jongwoo Lim, Ruei-Sung Lin, Ming-Hsuan Yang: Incremental learning for robust visual tracking. *International Journal of Computer Vision*, 77(1): 125–141, 2008.
[18] L. ZhangJ. HanB. Li, L. Wang: Scale adaptive compressive tracking. *Journal of National University of Defense Technology*, 35(5): 146–151, 2013.
[19] H. Huang, Y. Ge, J. Zhang, Y. Zha , D. Bi, Z. Hou: Application of affine transformation in compressed sensing tracking. *journal of Xidian University. (Natural Science Edition)*, 42(2): 162–166, 2015.
[20] S. Jiang, F. Li, Y. Zhang: Compression tracking algorithm based on multi-feature fusion. *Computer Engineering and Applications*, pp. 1-6, 2015.
[21] D.L. Donoho: Compressed sensing. *IEEE Trans. on Information Theory*, 52(4): 1289-1306, 2006.
[22] R. G. Baraniuk, M. A. Davenport, R. A. DeVore, and M. B. Wakin: A simple proof of the restricted isometry property for random matrices. *Constructive Approximation*, 28(3): 253–263, 2008.
[23] J. JingG. XUB. LeiY. HE: An improved real-time tracking algorithm based on compressed domain. *Computer Engineering*, 40(4): 170–175, 2014.
[24] Stefan Leutenegger, Margarita Chli and Roland Y. Siegwart: BRISK: Binary robust invariant scalable keypoint. *ICCV*, pp. 2548-2555, 2011.
[25] D.G. Lowe: Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2): 91-110, 2004.
[26] dward Rosten, Reid Porter, Tom Drummond: Faster and better: A matching learning approach to corner detection. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 32(1): 105–119, 2010.