IEICE TRANS. INF. & SYST., VOL.E99–D, NO.6 JUNE 2016

LETTER

Robust Object Tracking with Compressive Sensing and Patches Matching

Jiatian PI[†a)], **Keli HU**[††b)], ***Nonmembers***, **Xiaolin ZHANG**[†], ***Member***, **Yuzhang GU**[†], ***Nonmember***, ***and*** **Yunlong ZHAN**[†], ***Student Member***

SUMMARY Object tracking is one of the fundamental problems in computer vision. However, there is still a need to improve the overall capability in various tracking circumstances. In this letter, a patches-collaborative compressive tracking (PCCT) algorithm is presented. Experiments on various challenging benchmark sequences demonstrate that the proposed algorithm performs favorably against several state-of-the-art algorithms.

key words: *object tracking, compressive sensing, patches matching, feature extraction*

1. Introduction

Visual object tracking has drawn significant attention because of its critical role in many applications. Although great progress has been made in the past decade, it remains a challenging problem due to baffling conditions, such as illumination variations, background clutter and shape deformation.

The process of object tracking could be described as a dynamic state estimation problem, and the state information is usually the appearance representation. As the most fundamental appearance representation, raw pixel values [1] are widely employed in visual object tracking because of their simplicity and efficiency. Despite this, without feature extraction, the raw pixel representation is susceptible to complicated appearance changes. A local template-based method represents an object using a set of part templates, such as local histograms [2]. In contrast to the global visual representation, local features are able to cope with partial occlusions and model shape articulations flexibly. Cabido et al. [14] propose two different local search methods to optimize the final solution obtained by the particle filtering (PF). The two search areas in this method are systematically generated and randomly generated respectively. To improve the computing speed in those big search areas, they take advantage of the Graphics Processing Unit (GPU) to implement it. Unlike the proposed method, our algorithm segments the target into multiple patches and searches the candidate target in a small area to save the computational load. Fuhr et al. [15] use multiple patches to represent the tracking target as our method. However, the proposed algorithm is limited to track the pedestrian and uses camera parameters to track each patch. To extract the features efficiently, Zhang et al. [3] propose a compressive tracker (CT) based on the compressive sensing (CS) theory [4]. It adopts a very sparse measurement matrix in the compressed domain for constructing the appearance model. In addition to the real-time performance, such a scheme also helps the tracker achieve relatively good results on some challenging video sequences. However, compressive tracking often suffers from the drifting problem. Thus, some inaccurate samples will be supplied for the naive Bayes classifier. As the error accumulation goes on, the tracker may drift or even fail in the end. In this letter, to overcome the defect of the compressive tracking, the method of local patches searching is incorporated into it. Compared to the traditional compressive tracker, the improved algorithm is more robust while only adding a very low computational cost. The key contribution of this work can be summarized as follows. First, we improve the performance of the traditional compressive tracker in video sequence, and promote the application of the compressive sensing (CS) theory. Second, we incorporate the local patches searching into the traditional compressive tracking to achieve a more accurate tracking position, which can handle the drifting problem to some extent.

The remainder of this letter is organized as follows. The patches-collaborative compressive tracking (PCCT) algorithm is presented in Sect. 2. Then experimental results and discussions are shown in Sect. 3. The conclusion and our future work are summarized in Sect. 4.

2. Patch-Collaborative Compressive Tracking

In this section, we give the details of our patches-collaborative compressive tracking (PCCT) algorithm. The tracking issue is formulated as a binary classification problem via a naive Bayes classifier which is online updated with the samples in the compressed domain. We assume that the bounding box in the first frame is an initial template. And then, low-dimensional features are extracted from the initial template based on the compressive sensing. In addition, the local patches are applied for representing the initial template which remains the same in the following frames. The method of local patches searching is applied for achieving a more accurate tracking position, followed by the classifica-

Manuscript received November 16, 2015.
Manuscript revised January 24, 2016.
Manuscript publicized February 26, 2016.
[†]The authors are with SIMIT, Shanghai, 200050 China.
[††]The author is with Shaoxing University, Shaoxing, 312000 China.
a) E-mail: pijiatian@126.com
b) E-mail: ancimoon@gmail.com
DOI: 10.1587/transinf.2015EDL8235

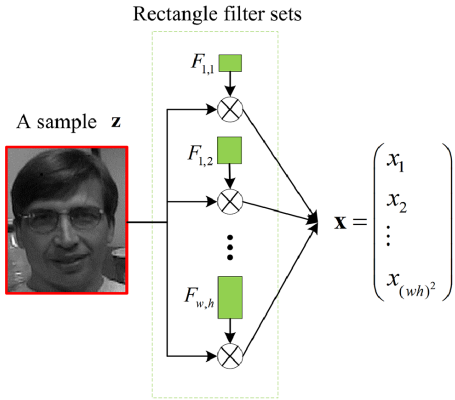

Fig. 1 The illustration of the convolution between a sample **z** and rectangle filters.

tion for the first time. At last, some positive samples near the more accurate tracking position, and negative samples away from it are used to update the classifier. Thus, the object location in the next frame will be predicted.

2.1 Low-Dimensional Feature

Assume that the tracking window is given by a single regular bounding box. The image representation of the template is formed by convolving the template with a Gaussian filter of different spatial variances. Zhang et al. [3] show that a truncated Gaussian filter can be replaced by several rectangle filters in practice. At each frame, positive samples near the current object bounding box and negative samples away from the object center are used to update the classifier. In the following frame, some samples around the previous target location to find the current target position. As shown in Fig. 1, for a sample $\mathbf{z} \in \mathbb{R}^{w \times h}$, it is processed by convolving $\mathbf{z}$ with a set of rectangle filters [3] at multiple scales $\{F_{1,1}, \ldots, F_{w,h}\}$ which are defined as

$$F_{w,h}(x,y) = \begin{cases} \frac{1}{wh} & 1 \le x \le w, 1 \le y \le h, \\ 0 & \text{otherwise.} \end{cases} \quad (1)$$

where w and h are the width and height of a rectangle filter respectively.

After the convolution, the filtered image is reshaped as a column vector $\mathbf{x} = (x_1, \ldots, x_n)^T \in \mathbb{R}^n$, $n = (wh)^2$. The dimension of this vector is typically very high. Then the random Gaussian matrix $\mathbf{R} \in \mathbb{R}^{m \times n}$ is adopted to project $\mathbf{x}$ into a low-dimensional space, denoted by $\mathbf{v} \in \mathbb{R}^m$, $m \ll n$. As shown in Fig. 2, the nonzero entries in $\mathbf{R}$ and the positions of rectangle filters in an input image corresponding to the nonzero entries in each row of $\mathbf{R}$ are needed to store. Then, the integral image [5] is used to efficiently compute the rectangular features corresponding to the nonzero entries in each row of $\mathbf{R}$.

2.2 Local Patches Searching

We use the vertical and horizontal patches to represent the

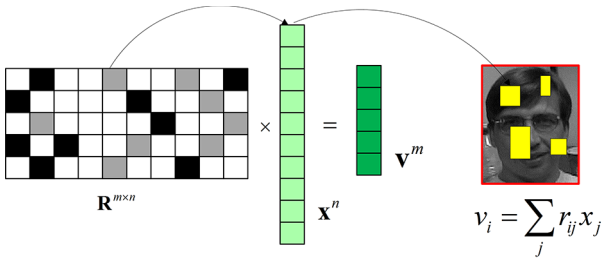

Fig. 2 Compressing a high-dimensional vector to a low-dimensional vector. Dark, gray and white rectangles represent positive, negative and zero entries respectively.

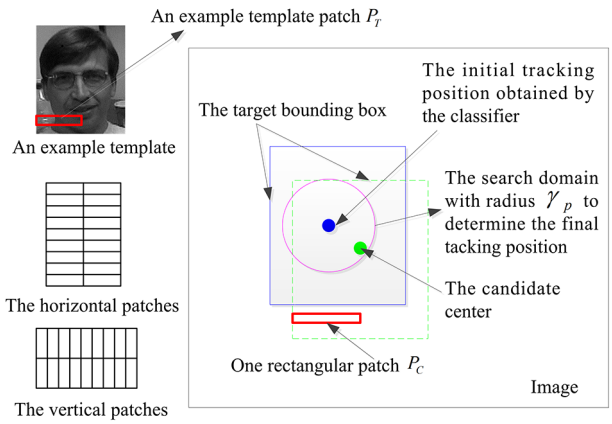

Fig. 3 The illustration of the vertical and horizontal patches. The bounding box shown in solid line is the initial tracking target obtained by the classifier. The bounding box shown in dashed line is the candidate target within radius γ_p.

template as shown in Fig. 3. Each small rectangular patch is regarded as the template patch. As noted in [2], this choice of patches is good enough for the patches searching. We extract histograms for each template patch with the integral histogram described in [6], which is an extension of the integral image [5]. In order to revise and determine the final tracking position, we will search in the neighborhood of the initial tracking position, which is estimated by using compressed information. In the search domain with a radius γ_p, we compare the similarity between patches. Assume $P_T = (d_x, d_y, d_w, d_h)$ is a rectangular patch in the template, whose center is (d_x, d_y), and whose width and height are d_w and d_h respectively. One of the candidate centers in the search domain is (x, y), the corresponding rectangular patch with P_T in this candidate can be defined as $P_C = (x + d_x, y + d_y, d_w, d_h)$. We measure the similarity between patches by computing the Earth Mover Distance (EMD) [6] between their histograms. Then, the corresponding score $S_{P_T}(x, y)$ for the template patch P_T is defined as

$$S_{P_T}(x,y) = EMD_{distance}(P_C, P_T). \quad (2)$$

Then, we sum the total corresponding score with all template patches for this candidate center (x, y). Every candidate center in the search domain will have a total score. At last, the candidate center with the minimal total score is

Table 1 Main steps of our algorithm

Algorithm 1:PCCT

Input:a template represented as a bounding box.
Use random measurement matrix $\mathbf{R}$ to extract the low-dimensional feature.
Use vertical and horizontal patches to represent the template.
for $t = 2$ to number of frames do
Sample a set of samples in $D^{\gamma_c} = \{\mathbf{z} \mid \|\mathbf{I}(\mathbf{z}) - \mathbf{I}_{t-1}\| < \gamma_c\}$, where $\mathbf{I}_{t-1}$ is the tracking position at the $(t-1)$-th frame, and extract the features with low dimensionality. γ_c is the searching radius for finding the initial tracking position. $\mathbf{z}$ is the sample.
Use the classifier to find the initial tracking position $\mathbf{I}_t$ with the maximal classifier response.
Use local patches searching and the initial template in the first frame to revise and determine the final tracking position $\mathbf{I}_t$ with the minimal total score.
Sample positive samples in $D^{\alpha} = \{\mathbf{z} \mid \|\mathbf{I}(\mathbf{z}) - \mathbf{I}_t\| < \alpha\}$ and negative samples in $D^{\beta,\lambda} = \{\mathbf{z} \mid \beta < \|\mathbf{I}(\mathbf{z}) - \mathbf{I}_t\| < \lambda\}$ with $\alpha < \beta < \lambda$, where α is the search radius for sampling positive samples, and β and λ are the radiuses for sampling negative samples. Extract the low-dimensional features with these two sets of samples.
Update the classifier.
end for
Output: Tracking position $\mathbf{I}_t$.

chosen as the final tracking position.

2.3 Tracking Algorithm

Assume all elements in the vector $\mathbf{v} \in \mathbb{R}^n$ are independently distributed. A naive Bayes classifier is applied for modeling these elements. We use Gaussian distribution which is a good approximation of features in the projected space to separate samples. Main steps of our algorithm are presented in Algorithm 1 (see Table 1). The main reasons that our method outperforms the traditional compressive tracker (CT) can be attributed to two factors. First, when there is a tracking drift, the patches of the current tracking box could be different with the initial template. After searching in the neighborhood, a more accurate tracking box which has a small difference with the initial template will be found. In addition, the initial template is unchangeable and will generate more "correct" positive samples to maintain good performance of the classifier. Second, the low-dimensional features based on compressive sensing still have better discriminative capability [3], which ensure the current tracking box found by the classifier is not far away from the correct location. Thus, only a small searching radius is needed for the local patches, which contributes a lot for decreasing the computation of patches matching.

3. Experiments

In this section, our patches-collaborative compressive tracking (PCCT) algorithm is evaluated with other 9 state-of-the-art methods on 7 challenging sequences. These methods are CT [3], BSBT [7], Frag [2], LOT [8], MSV, PDV in VIVID tested [9], SeminT [10], VTS [11] and VTD [12]. All of these trackers run in the one-pass evaluation (OPE) [13]. These 7 sequences and their corresponding ground truth

Table 2 Success rate (%). Bold fonts indicate the best performance.

Seqs	VTS	VTD	MSV	LOT	BSBT	SemiT	CT	Frag	PDV	PCCT
boy	62.9	61.9	56.8	52.4	54.1	40.3	58.6	38.2	19.3	**73.4**
dudek	**79.7**	78.5	48.3	53.1	69.0	51.0	64.0	52.9	20.4	70.5
tiger2	31.2	31.2	17.8	13.4	16.3	23.6	44.8	11.8	11.8	**47.4**
jump	15.5	12.8	25.4	57.4	15.1	6.2	4.6	66.0	63.2	**69.3**
david2	67.2	67.8	6.4	59.6	51.8	53.5	29.4	23.8	40.1	**73.3**
david3	53.4	39.8	10.4	**65.8**	35.5	14.7	30.3	66.0	44.2	43.0
lem	45.4	43.0	**63.5**	59.0	30.9	14.0	54.5	30.2	57.4	53.3
overall	50.8	47.9	32.7	51.5	38.9	29.0	40.9	41.3	36.6	**61.5**
fps	5.7	5.7	32.4	0.7	7.0	11.2	**64.4**	6.3	32.6	13.3

files, the compared code library, and all of the tracking results for compared trackers are available on the benchmark [13]. The total frame number for all sequences is 4550. For each tracker, the default parameters with the source code are used in all evaluations. The proposed PCCT algorithm runs at 13.3 frame per second (FPS) with a C++ implementation on an Xeon(R) X5675 machine with 3.07 GHz CPU and 6 GB RAM without any optimizing.

In our experiments, the search radius α for sampling positive samples is set to 4, and the searching radius β and λ for sampling negative samples are set to 8 and 22.5 respectively. In addition, 50 negative samples are randomly selected from the negative samples set $D^{\beta,\lambda}$. The searching radius γ_c for finding the initial tracking position is set to 25. Parameters in the classifier is set as the same as CT. The histograms of gray scale image used in patches contain 16 bins, and the searching radius to determine the final tracking position γ_p is set to 3. The template for local patches searching is fixed at the first frame and kept the same all through the sequence. The success rate based on the overlap metric is applied for analyzing the performance of each algorithm [13].

For each tracker, the average success rate of each sequence is given in Table 2. According to the experimental results, the proposed algorithm achieves outstanding performances in most sequences. The overall overlap scores achieved by the proposed algorithm are 61.5%, while the CT algorithm is 40.9%. Our method achieves much better results than the CT algorithm, which shows the effectiveness of using local patches searching.

The tracking results for the sequences *david2*, *dudek*, *tiger2* are shown in Fig. 4 (a)-(c). When the out-of-plane rotation occur, such as frame #123 in *david2* and frame #66 in *tiger2*, most algorithms fail to track the target except ours. The target object in the *boy*, *jump*, *dudek*, *lem* and *tiger2* sequences undergo abrupt movements and motion blur, especially in *boy* and *jump*. However, the proposed algorithm performs well as illustrated in frames #264, #386 and #403. The object in *jump* sequence jumps fast for a long time, our algorithm also tracks the target as well. The tracking results for the *boy*, *jump* sequences are shown in Fig. 5 (a), (b). When the part occlusion occurs in *lem* sequence at frame #303, #373, #453 and frame #81, #242 in *david3* sequence, our proposed tracker can find the target. The tracking results for the *david3* and *lem* sequences are shown in Fig. 5 (c), (d). Although the object in frame #107 from *tiger2* sequence un-

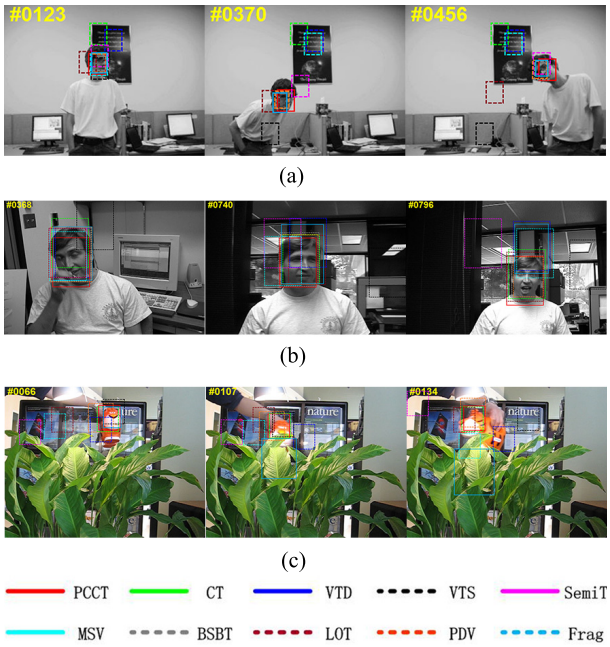

Fig. 4 The tracking results of the sequences (a) *david2*, (b) *dudek*, (c) *tiger2*.

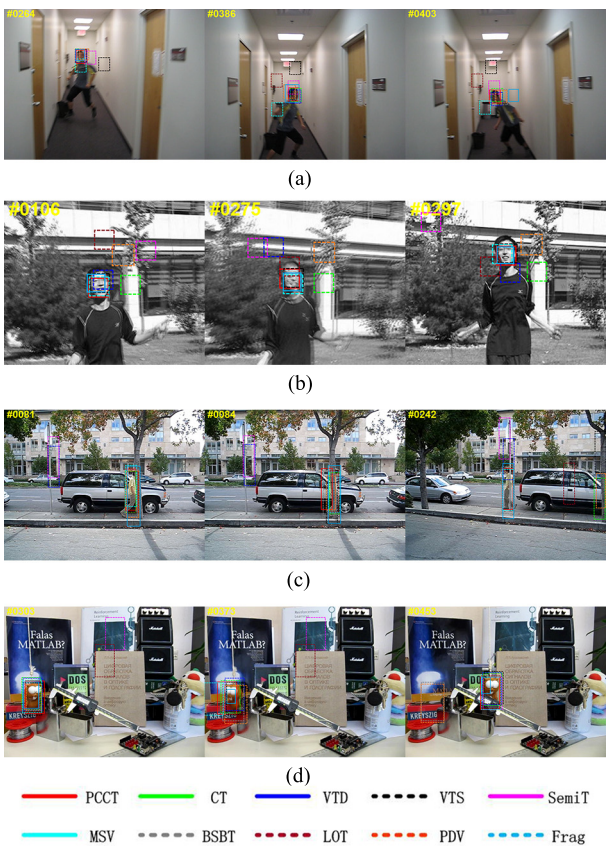

Fig. 5 The tracking results of the sequences (a) *boy*, (b) *jump*, (c) *david3*, (d) *lem*.

dergoes both the problem of illumination variation and part occlusion, the proposed algorithm can handle this well.

4. Conclusion

In this paper, we propose a robust tracking algorithm which combines the method of local patches searching with the compressive tracker. A very sparse measurement matrix is adopted to efficiently compress features from the target to find the initial object position in real time. Some local patches computed by the integral histogram are used to revise and determine the final object position. Experimental results on several challenging sequences demonstrate that the proposed algorithm performs well in terms of accuracy and robustness. However, the tracker loses the target when the object is totally occluded, as illustrated in frame #84 in *david3* sequence. Consequently, we will focus on improving the algorithm under heavy occlusion in the future.

References

[1] B.D. Lucas and T. Kanade, "An Iterative Image Registration Technique with An Application to Stereo Vision," Proc. 7th IJCAI, Vancouver, British Columbia, vol.81, pp.674–679, Aug. 1981.
[2] A. Adam, E. Rivlin, and I. Shimshoni, "Robust Fragments-based Tracking using the Integral Histogram," Proc. CVPR, New York, USA. vol.1, pp.798–805, June 2006.
[3] K. Zhang, L. Zhang, and M.-H. Yang, "Real-time Compressive Tracking," Proc. ECCV, Firenze, Italy, vol.7574, pp.864–877, Oct. 2012.
[4] E.J. Candes and T. Tao, "Decoding by linear programming," IEEE Trans. Inf. Theory, vol.51, no.12 pp.4203–4215, Dec. 2005.
[5] P. Viola and M. Jones, "Rapid object detection using a boosted cascade of simple features," Proc. CVPR, Kauai, USA, vol.1, pp.I-511–I-518, Dec. 2001.
[6] Y. Rubner, C. Tomasi, and L. Guibas, "The earth mover's distance as a metric for image retrieval," Int. J. Comput. Vis., vol.40, no.2, pp.91–121, Nov. 2000.
[7] S. Stalder, H. Grabner, and L. Gool, "Beyond Semi-Supervised Tracking: Tracking Should Be as Simple as Detection, but not Simpler than Recognition," Proc. IEEE ICCV, Kyoto, Japan, pp.1409–1416, Sept. 2009.
[8] S. Oron, A. Bar-Hillel, D. Levi, and S. Avidan, "Locally Orderless Tracking," Proc. IEEE CVPR, Providence, Rhode, pp.1940–1947, June 2012.
[9] R. Collins, X. Zhou, and S.K. Teh, "An Open Source Tracking Testbed and Evaluation Web Site," Proc. 12th IEEE Int. Workshop on PETS, Miami, Florida, USA. pp.17–24, Jan. 2009.
[10] H. Grabner, C. Leistner, and H. Bischof, "Semi-supervised On-Line Boosting for Robust Tracking," Proc. ECCV, Marseille, France, pp.234–247, Oct. 2008.
[11] J. Kwon and K.M. Lee, "Tracking by Sampling Trackers," Proc. IEEE ICCV, Barcelona, Spain, pp.1195–1202, Nov. 2011.
[12] J. Kwon and K.M. Lee, "Visual Tracking Decomposition," Proc. IEEE CVPR, San Francisco, CA, USA, pp.1269–1276, June 2010.
[13] Y. Wu, J. Lim, and M.-H. Yang, "Online Object Tracking: A Benchmark," Proc. IEEE CVPR, Portland, Oregon, USA, pp.2411–2418, June 2013.
[14] R. Cabido, A.S. Montemayor, J.J. Pantrigo, and B.R. Payne, "Multiscale and local search methods for real time region tracking with particle filters: local search driven by adaptive scale estimation on GPUs," J. Machine Vis. and Applications, vol.21, no.1, pp.43–58, Nov. 2009.
[15] G. Fuhr and C.R. Jung, "Robust Patch-Based Pedestrian Tracking using Monocular Calibrated Cameras," Proc. 25th SIBGRAPI Conf. Graphics, Patterns and Images, Ouro Preto, pp.166–173, Aug. 2012.