Bag of Features Tracking

Fan Yang, Huchuan Lu
School of Information and Communication Engineering
Dalian University of Technology
Dalian, CHINA
fyang.dut@gmail.com, lhchuan@dlut.edu.cn

Yen-Wei Chen
College of Information Science and Engineering
Ritsumeikan University
Kusatsu, JAPAN
chen@is.ritsumei.ac.jp

***Abstract*—In this paper, we propose a visual tracking approach based on "bag of features" (BoF) algorithm. We randomly sample image patches within the object region in training frames for constructing two codebooks using RGB and LBP features, instead of only one codebook in traditional BoF. Tracking is accomplished by searching for the highest similarity between candidates and codebooks. Besides, updating mechanism and result refinement scheme are included in BoF tracking. We fuse patch-based approach and global template-based approach into a unified framework. Experiments demonstrate that our approach is robust in handling occlusion, scaling and rotation.**

***Keywords*-visual tracking; bag of features; patch-based;**

I. Introduction

For successful tracking, researchers have proposed several approaches and attained remarkable experimental results [1], [2], [3]. Nevertheless, these approaches are inclined to fail in presence of occlusion, for that they rely on the global template model focusing on the integrated appearance information of the object. To specifically deal with occlusion, several approaches have been presented [4], [5], [6].

Recently, "bag of features" (BoF) representation has been successfully applied to object and natural scene classification owing to its simplicity, robustness and good practical performance. Fei-Fei *et al.* [7] use BoF for learning natural scene categories by representing the image as a collection of "codewords" and constructing a Bayesian hierarchical model to describe the codewords distribution. Sivic *et al.* [8] and Fergus *et al.* [9] apply the similar approaches for multi-class image classification. Most previous works use key-points detection strategy to select image patches and SIFT descriptor to represent them; however, Nowak *et al.* [10] evaluate several common-used approaches and verify that random sampling strategy has more discriminative power than interest point based sampler for the large numbers of patches.

Inspired by the ability of BoF in handling intra-class pose variant and occlusion in the image classification, we propose to incorporate BoF approach for object tracking and name it "BoF tracking". It combines BoF with incremental PCA tracking [3]. Online updating and result refinement are also included to ensure accurate results.

II. Algorithms

A. Bag of Features

Given N training images belonging to M classes, an interest point detector is employed to detect representative interest points. As basic elements of object representation, image patches around detected points are extracted and normalized to uniform scale. Then SIFT descriptor is used to represent features of these local regions. However, sliding window, regular grid and random sampling can also be applied to obtain patches, as well as other feature descriptors.

Given a collection of image patches, codebook is formed by performing K-means clustering algorithm. K is the size of codebook. Codewords are then defined as the centers of clusters. Thus descriptors in each training image can be coded by hard assignment to the nearest codeword, yielding a histogram $n(w_i, d_j)$ counting the frequency of occurrence of each codeword, where w_i denotes i^{th} visual word in the K-size codebook and d_j is j^{th} class. The histogram is treated as a "bag". After representing a test image as a histogram, the most similar training histogram can be found by some similarity metric and corresponding object class label is also returned.

B. Incremental PCA

Lim *et al.* [3] employ a model that reflects changes in appearance for tracking. It builds a PCA subspace representation $M(I_{0\ldots t-1}) = (\mu, B)$ for the target, where the subspace mean μ and the eigenbasis B are updated by an incremental subspace learning algorithm on the previous tracks $I_0, \ldots, I_{t-1}$. Incremental subspace method learns a probability distribution model to depict the object appearance online. It only needs to store the old mean, the old eigenbasis and the old covariance matrix rather than to record all previous data, and then it updates the probability distribution model when new data arrive. The measurement of the similarity is defined as the reconstruction error of I_t with respect to $M(I_{0\ldots t-1})$

$$E(I_t; (\mu, B)) = \left\|(I_t - \mu) - BB^T(I_t - \mu)\right\|^2 \qquad (1)$$

where B contains subspace bases and μ is subspace mean. Details of the algorithm can be found in [3].

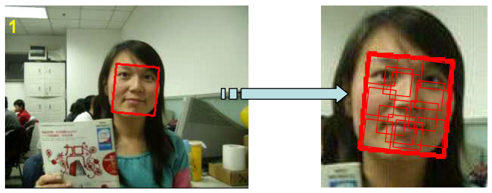

Figure 1. Random sampling

III. BoF Tracking

A. Initialization

Before tracking, initial parameters should be specified manually, including central point, width, height and rotation angle of the object, as well as sampling radius of affine parameters – scaling, translation and rotation. For each frame, amounts of candidate targets with different affine parameters are randomly generated by particle filter around the tracked result in the previous frame. Our task is to select the best particle that is most similar to the object.

B. Codebook Construction

In the tracking realm, there are only few training images at the beginning, which are insufficient to generate codebook and bags. In order to provide sufficient training samples, we apply incremental PCA tracking in the first M frames ($M = 5$ in our experiments). Then patches with a constant scale are randomly sampled within the region of the tracked object, as Figure 1 shows. The merit of random sampling is testified in [10].

In each frame, we collect N image patches and represent them by RGB descriptor and Local Binary Pattern (LBP) descriptor [11]. For color, we quantize each of the RGB channels into k bins and construct a k^3 bins histogram. For texture, we adopt the extended LBP operator $LBP_{P,R}^{riu2}$. As frames slip, patches accumulate. By performing K-means clustering algorithm, features are gathered into a number of clusters and cluster centers compose the codebook. Figure 2 depicts the process of codebook construction. Different from the large-size codebook required in object classification (because different objects belonging to the same class usually contain various appearance and poses), to ensure the computational efficiency we limit K to be smaller than 50, and run the tracking system of several Ks. We find that $K = 20$ is a good trade-off between accuracy and efficiency.

After codebook construction, training images are represented by codewords as bags. A bag is equivalent to the occurrence frequency of codewords in an image and can be represented as a histogram. M training images are converted to a set of bags $\{B_m\}_{m=1}^{M}$ by raw counts.

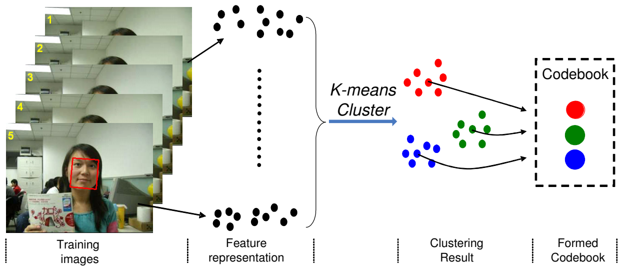

Figure 2. Codebook construction

C. Main Tracking Procedure

Given trained codebooks and bags, we perform our BoF tracking. When a new frame arrives, we randomly sample T candidate targets, extract N patches and calculate candidates' similarities with codewords. Algorithm 1 shows the process (w.r.t. one codebook and one candidate). To compute the distance between a patch and a codeword, we simply use Euclidean distance metric, and for distance between the test bag and a trained bag, we measure them through χ^2 test.

Algorithm 1 BoF Tracking

```
Given: set of patches $\{P_n\}_{n=1}^{N}$ in a candidate target $S_t \in \{S_t\}_{t=1}^{T}$; codebook $\{C_k\}_{k=1}^{K}$ containing $K$ codewords; set of trained bags $\{B_m\}_{m=1}^{M}$; bag of the candidate $B_t^* = 0$ with length $K$
1: for $n$=1 to $N$ do
2:   for $k$=1 to $K$ do
3:     compute distance between codewords and patches $d_{n,k} = dist(P_n, C_k)$
4:   end for
5:   select $D_n = \min_{\{C_k\}_{k=1}^{K}} d_{n,k}$ and corresponding index $I$ of the codeword
6:   $B_t^*(I) = B_t^*(I) + 1$
7: end for
8: compute bag similarity $V_{B,t} = \min_{\{B_m\}_{m=1}^{M}} dist(B_t^*, B_m)$
9: compute patch similarity $V_{P,t} = \sum_{n=1}^{N} e^{-D_n}$
Output: Similarities $V_{B,t}$ and $V_{P,t}$
```

Once achieving a candidate's similarities for the two features, we combine them using Eq.2.

$$\begin{aligned} S_o &= \arg\max_{\{S_t\}_{t=1}^{T}} \left(w_t^{LBP} V_{P,t}^{LBP} + w_t^{RGB} V_{P,t}^{RGB} \right) \\ w_t^{LBP} &= \frac{V_{B,t}^{LBP}}{V_{B,t}^{LBP} + V_{B,t}^{RGB}}, w_t^{RGB} = \frac{V_{B,t}^{RGB}}{V_{B,t}^{LBP} + V_{B,t}^{RGB}} \end{aligned} \tag{2}$$

Bag similarity is used to weight the corresponding patch similarity. The candidate with maximal combined similarity is selected as the tracked object S_o.

D. Updating

Since appearance and pose changes of an object occur all the time, updating is necessary or even crucial. In our approach, we devise a codebook updating scheme, as shown in Algorithm 2.

Algorithm 2 Codebook Updating

```
1: After obtaining the tracked object $S_o$) in frame $F_i$
    1.1: order the values $\{e^{-D_n}\}_{n=1}^{N}$ that indicate the similarity of a patch with the codebook
    1.2: collect $p$ patches with highest value $e^{-D_n}$ using Eq.3
2: Repeat step 1.1 and step 1.2 in the following $f$ frames
3: Perform K-means again on the new patch collection containing $fp$ patches and the old codebook
```

We use Eq.3 to adaptively adjust the number of selected ”good” patches.

$$p = \begin{cases} p_1 & V_{P,t} \le Th \\ p_2 & V_{P,t} > Th \end{cases}, \quad p_1 < p_2 \tag{3}$$

where $V_{P,t}$ is the patch similarity the tracked object. Th is a predefined and constant threshold. After f frames, we obtain a new collection of patches $\{P_i\}_{i=1}^{fp}$. We then perform K-means again on $\{P_i\}_{i=1}^{fp}$ and the old codebook $\{C_k\}_{k=1}^{K}$ using Eq.4. Note that the size of the codebook remains 20 after updating.

$$\{C_k^*\}_{k=1}^{K} = kmeans(\{P_i\}_{i=1}^{fp}, \lambda \{C_k\}_{k=1}^{K}) \tag{4}$$

where $\{C_k^*\}_{k=1}^{K}$ denotes the new codebook and $0 < \lambda < 1$. λ is a forget factor imposed on the old codebook to reduce its importance gradually so that the newly-constructed codebook pays more attention to the latest patches.

E. Results Refinement

To improve the tracking performance, we use incremental PCA tracking to refine the results: when the combined similarity of BoF tracking is below a threshold Th_r, we combine the respective “best” particle of incremental PCA tracking and BoF tracking

$$X_t = \alpha X_{t1} + (1 - \alpha) X_{t2} \tag{5}$$

where X_{t1} and X_{t2} represent the affine parameters of incremental PCA tracking and BoF tracking. X_t is affine parameters of the refined result and $\alpha = 0.7$. If the combined similarity of BoF tracking is greater than Th_r, we treat its result reliable. Then the tracked object is only determined by BoF tracking. In this case, we omit the result refinement stage. IVT is only updated using the method in [3], without contribution to the final tracking result.

IV. Experimental Results

Testing sequences are color images from CAVIAR Database and our own dataset. 64-dim RGB and 59-dim LBP histogram are computed to represent patches. The number of candidates $T = 300$. Patches collected from a candidate are 50. We test 4 kinds of patch's size and choose 12×12 pixels. Forget factor $\lambda = 0.9$. ***Note that these parameters are fixed for all sequences.*** For comparison we run FragTrack [6] and incremental PCA tracking (IVT) [3] on the same video sequences. Some screenshots of the tracking results are shown in Figure 3. Figure 4 is the quantitative comparison.

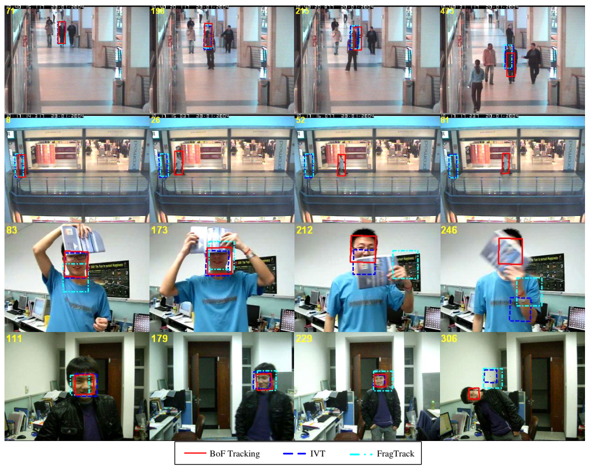

Figure 3. Screenshots of tracking results

A. Result Analysis

In Figure 3, the first row shows a sequence *ShopAssistant2cor* from CAVIAR Database [12]. In the sequence, a person walks along the passage and is occluded by another person. We can see that all the three approaches keep track of the person during the entire tracking procedure, but the result of our approach is more accurate, especially when the occlusion ends. The second row is the tracking results on sequence *OneStopEnter1front* also from CAVIAR Database. In this sequence, a person walks along the street from left to right. Although the object's motion is simple, successful tracking is still challenging due to severe background clutter and small size of the human body. Only BoF tracking can find the person's position; whereas both FragTrack and IVT fails at the beginning of the sequence.

In the third and the last row, we compare the results on our own sequences. The experiments intend to test the ability of tracking methods to deal with continuous occlusion and various pose changes. In sequence *occlusion1*, the person's face is occluded by a book constantly. IVT gradually fails when encountering complex and long-time occlusion. Although FragTrack is designed to deal with occlusion, it still fails when encountering complex occlusion. In contrast, BoF tracking keeps track of the face during the entire tracking procedure. In sequence *boy*, FragTrack cannot deal with rotation and scale change due to lacking updating mechanism. Contrarily, our approach does not rely on a constant template and include an update stage; thus it is more robust.

The experiments have demonstrated that our BoF tracking remarkably outperforms FragTrack and IVT in dealing with occlusion. We attribute it to the usage of codebook and patch-based approach. Since we extract some representative

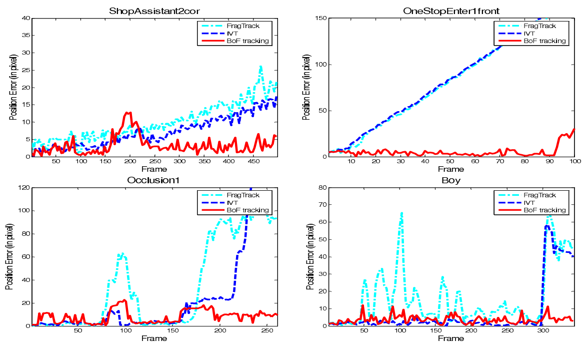

Figure 4. Error curves

features of the target and rely on the collection of patches, we can obtain sufficient information to locate the target, as long as part of the target is visible. With the update scheme, our approach can also capture the changes of the object. For more tracking results on some sequences from CAVIAR database, please refer to Figure 5.

B. *Influence of Parameters*

We also test the influence of patch size and adaptive weight in Eq.2. Table I shows a comparison among the results (numbers indicate the average position error per frame in pixel). In the table, "adaptive" means adaptive weight is used while "constant" means we use a constant weight (0.5 for both of the two features). Note that all other parameters are the same.

Table I
INFLUENCE OF PARAMETERS

Param / *Seq.*	8×8 adaptive	12×12 adaptive	16×16 adaptive	20×20 adaptive	12×12 constant
ShopAssistant2cor	65.03	**3.36**	4.02	14.84	5.28
OneStopEnter1front	5.65	**4.01**	5.37	9.24	4.28
occlusion1	11.23	**8.23**	9.31	12.10	31.78
boy	6.97	**2.97**	6.17	6.37	3.67

From Table I, we can see that the performance is the best when patch size is 12×12 pixels with adaptive weight. It is in accordance with the assumption that small patches excessively focus on local details and overlook general information while too large patches lose enough discriminative features. Adaptive weight also improves the tracking performance.

V. CONCLUSIONS

In this paper, we propose a framework incorporating BoF algorithm into visual tracking. We apply incremental PCA tracking to collect training samples to construct two small-size codebooks using RGB and LBP features. Meanwhile, we devise a scheme to update codebooks. By employing incremental PCA tracking to refine poor results of BoF tracking, we combine patch-based approach and global template-based approach into a unified framework. Experiments show that our approach outperforms incremental PCA tracking and FragTrack. We also study the influence of some parameters on the tracking performance.

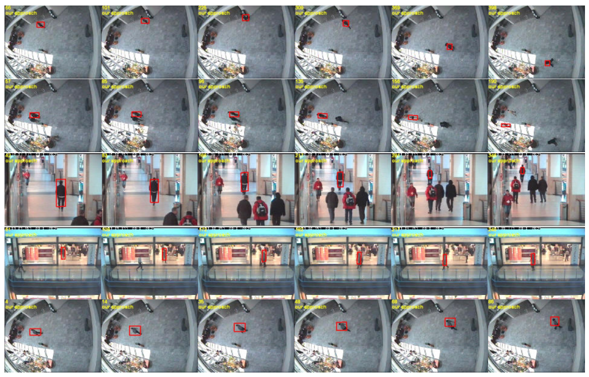

Figure 5. More tracking results of our approach

REFERENCES

[1] S. Avidan, "Ensemble tracking," in *Proc. CVPR'05*, vol. 1, 2005, pp. 494–501.

[2] H. Grabner and H. Bischof, "On-line boosting and vision," in *Proc. CVPR'06*, vol. 1, 2006, pp. 260–267.

[3] J. Lim, D. Ross, R. Lin, and M. Yang, "Incremental learning for visual tracking," in *Proc. NIPS'04*, 2005, pp. 793–800.

[4] X. Song, J. Cui, H. Zha, and H. Zhao, "Vision-based multiple interacting targets tracking via on-line supervised learning," in *Proc. ECCV'08*, 2008.

[5] B. Babenko, M.-H. Yang, and S. Belongie, "Visual tracking with online multiple instance learning," in *Proc. CVPR'09*, 2009.

[6] A. Adam, E. Rivlin, and I. Shimshoni, "Robust fragments-based tracking using the integral histogram," in *Proc. CVPR'06*, vol. 1, 2006, pp. 798–805.

[7] L. Fei-Fei and P. Perona, "A bayesian hierarchical model for learning natural scene categories," in *Proc. CVPR'05*, 2005.

[8] J. Sivic and B. C. Russell, "Discovering object categories in image collections," in *Proc. ICCV'05*, 2005.

[9] R. Fergus, L. Fei-Fei, P. Perona, and A. Zisserman, "Learning object categories from google's image search," in *Proc. ICCV'05*, 2005.

[10] E. Nowak, F. Jurie, and B. Triggs, "Sampling strategies for bag of features image classification," in *Proc. ECCV'06*, 2006.

[11] T. Ojala, M. Pietikainen, and T. Maenpaa, "Multiresolution gray-scale and rotation invariant textureclassification with local binary patterns," *PAMI*, vol. 24(7), pp. 971–987, 2002.

[12] CAVIAR database. [Online]. Available: http://groups.inf.ed.ac.uk/vision/caviar/