Superpixel Tensor Pooling for Visual Tracking using Multiple Midlevel Visual Cues Fusion

Chong Wu, Student Member, IEEE, Le Zhang, Jiawang Cao, and Hong Yan, Fellow, IEEE

Abstract—In this paper, we propose a method called superpixel tensor pooling tracker which can fuse multiple midlevel cues captured by superpixels into sparse pooled tensor features. Our method first uses superpixel segmentation to produce different patches from the target template or candidates. Then it encodes different midlevel cues like HSI color, RGB color, and spatial coordinates of each superpixel into a histogram matrix to construct a new feature space. Next, these matrices are formed to a 3-order tensor. After that, the tensor is pooled into the sparse representation. Then the incremental positive and negative subspaces learning is performed. Our method has both good characteristics of midlevel cues and sparse representation hence is more robust to large appearance variations and can capture compact and informative appearance of the target object. To validate the proposed method, we compare it with eight state-of-the-art methods on 24 sequences with multiple visual challenges. Experiment results demonstrate that our method outperforms them significantly.

Index Terms—Incremental positive and negative subspace learning, midlevel visual cues, superpixels, tensor pooling, visual tracking.

I. INTRODUCTION

The study of visual tracking has been achieved great successes in recent years. However, because of the heavy occlusion, drifts, fast motion, severe scale variation, large shape deformation, etc., visual tracking is still a challenge in computer vision [1], [2], [3].

To overcome these challenges, several effective visual tracking methods have been proposed. Different levels of appearance and spatial cues are successfully applied in visual tracking methods [4], [5]. Compared to high-level structure cues and low-level visual cues, midlevel visual cues are shown more effective in representing the structure of the image [4], [5], [6], [7]. Some researchers applied superpixel methods to obtain the midlevel cues in visual tracking and their methods show robust against heavy occlusion and drifts [4], [5], [6], [7], [8]. Because of the utilization of superpixels, the dimension of original data will reduce (for a matrix, it will reduce to a vector). It will cause the losing of spatial information of the target object. Hence, how to fuse spatial information into the appearance model constructed by midlevel cues is still a problem [8]. Some researchers used the Euclidean distances from the target to the candidates as the weight to integrate spatial information [4], [5]. It can preserve some spatial information to some extent. However, the spatial information like the shape of target is still lost. Hence, some of these methods are more sensitive to color variations than spatial variations and it might result in a bad performance in a situation like background clutters. Some researchers have shown that integrating more information in sparse representation can improve the tracking performance [9]. And some researchers tried to fuse the depth cue with superpixel-based target estimation using graph-regularized sparse coding and improved the discriminative ability of the trackers [8]. What’s more, different color channels are suitable to different tracking scenarios. Integrating different color channels into the unified framework will also help to improve the robustness of tracker. Hence, it is of great interest to develop an elegant method for fusing multiple midlevel cues in sparse representation.

In this paper, we propose a visual tracking method called superpixel tensor pooling tracker (SPTPT) which can integrate multiple midlevel cues (like the information of different color channels, spatial coordinates, shape, etc.) obtained by superpixels in a unified sparse coding tensor form. With the utilization of sparse representation and midlevel cues, our method has both good characteristics of the midlevel cues and sparse representation. What’s more, through fusing multiple midlevel cues, our method shows more robust than some state-of-the-art methods under large appearance variations. The contribution of this paper can be concluded as follows,

1) This is the first attempt of using superpixels to obatin tensor-pooled sparse features. With the utilization of superpixels, the patches obtained are more meaningful than the patches obtained by sliding window.
2) Our method provides an effective fusion method for fusing multiple midlevel cues in a unified sparse representation. Hence the constructed discriminative appearance model can take advantage of different midlevel cues.

To validate our method, we select 8 state-of-the-art tracking methods (TPT [5], SPT [5], STRUCK [10], TLD [11], VTD [12], CSK [13], SMS [14], and CT [15]) and 24 sequences with multiple tracking challenges from the benchmark of the paper [16], and we compare their reasonable lower-bound performances (we used one default parameter for all sequences without any tuning). Experiment results show that the lower-bound performance of our method is significantly better than existing methods.

The rest of this paper is organized as follows. In Section 2, we first introduce the superpixel segmentation method used in our method. Then we describe the fusion model and the
Fig. 1. The workflow of SPTPT. **Step 1**: Use particle filter (PF) and affine transformation to produce some candidates. **Step 2**: Use SNIC to generate superpixels. **Step 3**: Compute the histograms of color and spatial information in each superpixel. **Step 4**: Construct the histograms to a 3-order tensor. **Step 5**: Continue Steps 3-4 till the features of all superpixels of all candidates are obtained. **Step 6**: Determine whether the current frame is the 1st frame. If yes, produce the dictionary matrix, store the tensor of it into an updating sequence and then go to **Step 1**, otherwise go to **Step 7**. **Step 7**: Use the dictionary matrix to do the pooling of tensors obtained in **Step 3**. **Step 8**: Evaluate the likelihood of tensors after pooling in positive and negative subspaces. **Step 9**: According to the likelihood, update the discriminative appearance model (if the max likelihood > 0, store the tensor of max likelihood into the updating sequence and use PF draw some negative samples to update negative subspace and if algorithm reaches update rate u, use IRTSA to update positive subspace). Continue **Steps 1-9** till the last frame.

incremental positive and negative subspaces learning method. In Section 3, we first illustrate the experiment settings and evaluation metrics. We then analyze the experiment results. Finally, we present the conclusion in Section 4.

**II. SUPERPIXEL TENSOR POOLING TRACKER**

In this section, we will introduce four main parts of the proposed SPTPT.

A. **Patches Extraction using Superpixel Segmentation**

Producing meaningful patches is very important to construct tensor-pooled sparse features. Compared to the patches obtained by sliding window [2], [3], superpixels are more meaningful, because superpixels can preserve the image structure and reduce the redundancy. Hence, we introduce superpixel segmentation to obtain patches (superpixels) in SPTPT. To construct tensor-pooled sparse features, we need to keep the number of patches obtained precisely. Hence, a superpixel method which can control the superpixel number precisely is needed. We select simple non-iterative clustering (SNIC) [17] which can generate precise number of superpixels as the superpixel method in SPTPT (compactness coefficient: 20, number of superpixels: 30, in this paper).

B. **Fusion Model for Multiple Midlevel Cues**

After producing the superpixels, we apply local sparse codes to encode features. Multiple midlevel cues (HSI color channels, RGB color channels, and spatial information are used in this paper) of different superpixels are first constructed to several same size histogram vectors, each vector for each cue in a superpixel is as follow,

\[ f = [f_1, f_2, \ldots, f_n]^T, \]

where, \( n \) is the number of bins of each histogram. Each element in \( f \) represents the frequency of each bin in a superpixel region. It can be calculated as follow,

\[ f_i = \frac{c}{\sum_{k=1}^{r} r_k}, \]

where, \( c \) is the number of pixels corresponding to bin \( i \) in a superpixel region and \( r \) is the total number of pixels in this superpixel region.

These vectors are combined to construct the feature matrix of each superpixel as follow,

\[ F = [f_1, f_2, \ldots, f_m], \]

where, \( m \) is the number of features, in this paper, \( m = 8 \) (H, S, I, R, G, B, x, y).

After that, we reshape the feature matrix to a feature vector \( a \in \mathbb{R}^B \) of each superpixel and compute the sparse coefficient vector \( h \in \mathbb{R}^z \) of it using the formula as follow,

\[ \min_{h_i} ||a_i - Dh_i||^2_2 + \lambda ||h_i||_1, \]

where, \( B = n \times m, D \in \mathbb{R}^{z \times s} \) is the dictionary matrix learned by the clustering result of \( a \) of the superpixels obtained in the 1st frame, \( z \) is the number of cluster centroids and \( s \) is the number of superpixels.

To preserve the spatial order of the superpixels, we arrange the sparse coefficient vector \( h \) of each superpixel of each candidate in a unified 3-order tensor \( T \in \mathbb{R}^{3 \times s \times d'} \) according
to the spatial order of their corresponding superpixel in the candidate templates. \( v \) is the number of candidate templates.

**C. Incremental Positive and Negative Subspaces Learning**

The update and learning scheme used in SPTPT is based on the incremental subspace learning [13]. In order to make the tracker more robust against drifts, we refer to papers [2], [3] to introduce discriminative framework called negative subspace learning into the learning scheme. Hence, the learning scheme used in SPTPT is called incremental positive and negative subspaces learning.

If algorithm reaches the update rate \( u \) (in this paper, \( u \) is set to 5), then a 3-order tensor \( \mathcal{T} \in \mathbb{R}^{2 \times 8 \times u} \) corresponding to positive subspace is constructed. We use IRTSA algorithm [18] to find the dominant projection subspaces of it. The details of IRTSA can be referred in [18]. To learn the positive subspace, it is necessary to evaluate the likelihood between the candidate sample and its approximation in the learned positive subspace. Given a 3-order tensor \( \mathcal{J} \in \mathbb{R}^{2 \times 8 \times 1} \) of a candidate in the new frame, the evaluation of its likelihood in the learned positive subspace can be determined by the reconstruction error as follows,

\[
RE_1 = \sum_{i=1}^{2} ||(\mathbf{J}(i) - \mathbf{M}(i)) - (\mathbf{J}(i) - \mathbf{M}(i)) \prod_{j=1}^{2} \times_j (U^{(j)}, U^{(j)T})||^2,
\]

\[
RE_2 = ||(\mathbf{J}(3) - \mathbf{M}(3)) - (\mathbf{J}(3) - \mathbf{M}(3) \cdot (V^{(3)}, V^{(3)T}))||^2,
\]

\[
RE = \gamma RE_1 + (1 - \gamma) RE_2,
\]

where \( \mathbf{M} \) is the mean tensor of \( \mathcal{T} \), \( \mathbf{M}(i) \) (\( i = 1, 2 \)) is the column mean and \( \mathbf{M}(i) \) (\( i = 3 \)) is the row mean of the mode-\( i \) unfolding matrix of \( \mathcal{T} \). \( \mathbf{J}(i) \) is the mode-\( i \) unfolding matrix of \( \mathcal{J} \), and \( \gamma \) is the control weight, in this paper, \( \gamma = 0.5 \).

As to the negative subspace learning, in contrast to the positive subspace learning which collects the positive samples one per frame using the sparse pooled tensor features obtained from the tracked frames and is incremental learned by IRTSA, the negative samples are only collected in the last tracked frame through extracting superpixels a certain distance threshold (several pixels) around the estimated location of the target [2], [3]. Since these negative samples are collected in only one frame rather than a sequence of frames, the negative subspace is learned directly by doing tensor decomposition (TD) of the sparse pooling tensors of these samples. The likelihood can also be calculated by Equations (1)-(3).

**D. Motion Model based on Bayesian Inference**

The motion model of SPTPT is based on Bayesian inference. Let \( X_t = \{x_t, y_t, \theta_t, s_t, \beta_t, \phi_t\} \) represent the state (affine transformation parameters) at \( t \)th frame, where \( x_t \) is \( x \) translation, \( y_t \) is \( y \) translation, \( \theta_t \) is rotation angle, \( s_t \) is the scale, \( \beta_t \) is the aspect ratio, and \( \phi_t \) is skew direction, let \( S_t \) represent a set of the observations \( \{S_1, S_2, ..., S_i\} \) at time \( t \). The posterior probability is calculated as follow,

\[
p(X_t|S_t) \propto p(S_t|X_t) \int p(X_t|X_{t-1})p(X_{t-1}|S_{t-1})dX_{t-1},
\]

where, \( p(S_t|X_t) \) represents the observation model here is the likelihood function, and \( p(X_t|X_{t-1}) \) denotes the dynamic model between states \( X_t \) and \( X_{t-1} \). We apply a particle filter [19] to generate samples (number of positive samples: 600 and number of negative samples: 200) in this paper) through estimating the distribution. The optimal state can be obtained by using the maximum a posteriori (MAP) estimation,

\[
\hat{X}_t = \arg \max_{X_t} p(S_t^i|X_t^i)p(X_t^i|X_{t-1}^i), i = 1, 2, ..., b,
\]

where, \( b \) is the number of samples and \( X_t^i \) represents the sample \( i \) of state \( X_t \). The dynamic model \( p(X_t|X_{t-1}) \) is formulated using random walk as follow [3],

\[
p(X_t|X_{t-1}) = N(X_t; X_{t-1}, \Omega),
\]

where, \( \Omega \) is a diagonal covariance matrix. And its diagonal elements are \( \sigma^2_{\theta}, \sigma^2_{\beta}, \sigma^2_{s}, \sigma^2_{\phi}, \sigma^2_{\phi} \), respectively.

Finally, the likelihood of a candidate in both positive and negative subspaces is formulated as follow:

\[
p(Y_t|X_t) \propto \exp(RE^{(-)} - RE^{(+)})
\]

To make the tracker more robust and avoid overfitting, SPTPT uses the likelihood function above to control the learning: only when the best candidate’s likelihood \( \exp(RE^{(-)} - RE^{(+)}) > \Phi \), it can be accepted into the updating scheme, where \( \Phi \) is a threshold, in this paper, it is set to 0. The details of SPTPT are shown in Fig. 1.

![Fig. 2. Visual comparison of three top ranked trackers in the experiments. Cyan box: SPTPT; Yellow box: SPT; Red box: TPT.](image-url)

**III. EXPERIMENTS AND RESULTS**

To validate the proposed method, we follow the protocol of benchmark [16]. We compared our method with nine state-of-the-art methods: (TPT [5], SPT [3], STRUCK [10], TLD [11], VTD [12], CSK [13], SMS [14], CT [15]) and tested them on 24 sequences (Basketball, Biker, Bird1, Bird2, Bolt, CarScale, DragonBaby, Football1, Lemming, Liquor, MountainBike, Panda, RedTeam, Rubik, Singer1, Skating1, Skating2, Soccer, Subway, Surfer, Tiger2, Trans, Trellis, Woman) with multiple visual tracking challenges (illumination variation, scale variation, occlusion, deformation, motion blur, fast motion, in-plane rotation, out-of-plane rotation, out-of-view, background clutter, and low resolution). SPTPT is implemented using MATLAB and conducted on a Mac with OSX 10.14, Intel
TABLE I
NUMBER OF SUCCESSFUL TRACKED FRAMES (LEFT COLUMN PER METHOD) AND CENTER LOCATION ERROR (RIGHT COLUMN PER METHOD)

| Sequence     | CSK     | CT     | SMS     | SPT     | Ours    | STRUCK | TLD     | TPT     | VTD     |
|--------------|---------|--------|---------|---------|---------|--------|---------|---------|---------|
| Soccer       | 53      | 34     | 68      | 76      | 69      | 58     | 20      | 22      | 25      |
| Football     | 62      | 43     | 73      | 89      | 74      | 66     | 24      | 27      | 29      |
| Basketball   | 53      | 34     | 68      | 76      | 69      | 58     | 20      | 22      | 25      |
| Singer       | 62      | 43     | 73      | 89      | 74      | 66     | 24      | 27      | 29      |
| Skating1     | 53      | 34     | 68      | 76      | 69      | 58     | 20      | 22      | 25      |
| Skating2     | 86      | 63     | 91      | 98      | 87      | 79     | 30      | 33      | 36      |
| Soccer       | 53      | 34     | 68      | 76      | 69      | 58     | 20      | 22      | 25      |
| Singer       | 62      | 43     | 73      | 89      | 74      | 66     | 24      | 27      | 29      |
| Skating1     | 53      | 34     | 68      | 76      | 69      | 58     | 20      | 22      | 25      |
| Skating2     | 86      | 63     | 91      | 98      | 87      | 79     | 30      | 33      | 36      |
| Soccer       | 53      | 34     | 68      | 76      | 69      | 58     | 20      | 22      | 25      |
| Singer       | 62      | 43     | 73      | 89      | 74      | 66     | 24      | 27      | 29      |
| Skating1     | 53      | 34     | 68      | 76      | 69      | 58     | 20      | 22      | 25      |
| Skating2     | 86      | 63     | 91      | 98      | 87      | 79     | 30      | 33      | 36      |

Fig. 3. Overall success plots of 9 trackers. Rank: Ours, TPT, SPT, STRUCK, TLD, VTD, CSK, SMS, CT.

Core i5 2.3 GHz 4 cores CPU, and 16 GB RAM. To keep the fairness, all methods are used one default parameter for all sequences without any tuning, that is to say, the results shown in this section are the lower-bound performance of each method.

A. Evaluation Metrics

Precision plots and success plots [16] are used to evaluate the overall performance and robustness of trackers. A precision plot illustrates the ratio of frames whose center location error is within a threshold distance to the ground truth. A success plot illustrates the percentage of frames of which overlapping rate between tracked results and ground truth is larger than a certain threshold. The final rank is according to the area under curve (AUC) of each tracker. Also to compare the performance of each tracker in per sequence, the number of successful tracked frames and center location error [20] are used.

B. Performance Analysis

Fig. 3 shows the visual comparison results of SPTPT, SPT (midlevel cues based method without using midlevel cues fusion), and TPT (tensor pooling tracker which uses sliding window to get patches and without using midlevel cues fusion). It shows that SPTPT has both good characteristics of SPT and TPT, at the same time, outperforms them. Table I shows the number of successful tracked frames and center location error obtained by each tracker in per sequence. The value in bold face means the best value and the value in underline italic type means the second best value. We can see that our tracker achieves the best or the second best score in most of sequences. Figs. 3 and 4 show the overall performance of nine trackers. The AUC of success plot and precision plot of our tracker are significantly higher than other trackers. It shows the robustness of our tracker. As to the running time, compared to TPT: 2.8s per frame (SPTPT), 3.0s per frame (TPT), SPTPT is faster than TPT.

IV. CONCLUSION

In this paper, we have proposed a visual tracking method which can fuse multiple midlevel cues obtained by superpixels to construct tensor-pooled sparse features. Our method has both good characteristics of the midlevel cues and sparse representation. In the validation, our method shows more robust against different visual tracking challenges than state-of-the-art methods.

ACKNOWLEDGMENT

This work was supported by the Hong Kong Research Grants Council (Project C1007-15G) and City University of Hong Kong (Project 9610034).

Fig. 4. Overall precision plots of 9 trackers. Rank: Ours, TPT, SPT, STRUCK, VTD, SMS, TLD, CSK, CT.
REFERENCES

[1] X. Zhan and B. Ma, “Gaussian mixture model on tensor field for visual tracking,” IEEE Signal Processing Letters, vol. 19, no. 11, pp. 733–736, Nov 2012.

[2] L. Huang and B. Ma, “Tensor pooling for online visual tracking,” in 2015 IEEE International Conference on Multimedia and Expo (ICME), June 2015, pp. 1–6.

[3] B. Ma, L. Huang, J. Shen, and L. Shao, “Discriminative tracking using tensor pooling,” IEEE Transactions on Cybernetics, vol. 46, no. 11, pp. 2411–2422, Nov 2016.

[4] S. Wang, H. Lu, F. Yang, and M. Yang, “Superpixel tracking,” in 2011 IEEE International Conference on Computer Vision (ICCV), Nov 2011, pp. 1323–1330.

[5] F. Yang, H. Lu, and M. Yang, “Robust superpixel tracking,” IEEE Transactions on Image Processing, vol. 23, no. 4, pp. 1639–1651, April 2014.

[6] S. D. Lin and D. Yang, “A visual tracking method using superpixel,” in 2017 International Conference on Machine Learning and Cybernetics (ICMLC), vol. 2, July 2017, pp. 552–557.

[7] L. Wang, H. Lu, and M. Yang, “Constrained superpixel tracking,” IEEE Transactions on Cybernetics, vol. 48, no. 3, pp. 1030–1041, March 2018.

[8] H. Li, J. Fang, and Q. Wang, “Robust superpixel tracking via depth fusion,” IEEE Transactions on Circuits and Systems for Video Technology, vol. 24, no. 1, pp. 15–26, Jan 2014.

[9] C. Huang and B. Jiang, “Occlusion handling of visual tracking by fusing multiple visual clues,” in 2017 IEEE International Conference on Systems, Man, and Cybernetics (SMC), Oct 2017, pp. 836–839.

[10] S. Hare, A. Saffari, and P. H. S. Torr, “Struck: Structured output tracking with kernels,” in 2011 IEEE International Conference on Computer Vision (ICCV), Nov 2011, pp. 263–270.

[11] Z. Kalal, J. Matas, and K. Mikolajczyk, “P-n learning: Bootstrapping binary classifiers by structural constraints,” in 2010 IEEE Computer Vision and Pattern Recognition (CVPR), June 2010, pp. 49–56.

[12] J. Kwon and K. M. Lee, “Visual tracking decomposition,” in 2010 IEEE Computer Vision and Pattern Recognition (CVPR), June 2010, pp. 1269–1276.

[13] F. Henriques, R. Caseiro, P. Martins, and J. Batista, “Exploiting the circulant structure of tracking-by-detection with kernels,” in European Conference on Computer Vision (ECCV). Springer, 2012, pp. 702–715.

[14] R. T. Collins, “Mean-shift blob tracking through scale space,” in 2003 IEEE Computer Vision and Pattern Recognition (CVPR), vol. 2, June 2003, pp. II–234.

[15] K. Zhang, L. Zhang, and M.-H. Yang, “Real-time compressive tracking,” in European Conference on Computer Vision (ECCV). Springer, 2012, pp. 864–877.

[16] Y. Wu, J. Lim, and M. Yang, “Object tracking benchmark,” IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 37, no. 9, pp. 1834–1848, Sep. 2015.

[17] R. Achanta and S. Sisirunk, “Superpixels and polygons using simple non-iterative clustering,” in 2017 IEEE Computer Vision and Pattern Recognition (CVPR), July 2017, pp. 4895–4904.

[18] X. Li, W. Hu, Z. Zhang, X. Zhang, and G. Luo, “Robust visual tracking based on incremental tensor subspace learning,” in 2007 IEEE International Conference on Computer Vision (ICCV), Oct 2007, pp. 1–8.

[19] M. Isard and A. Blake, “Contour tracking by stochastic propagation of conditional density,” in European Conference on Computer Vision. Springer, 1996, pp. 343–356.

[20] M. Everingham, L. Van Gool, C. Williams, J. Winn, and A. Zisserman, “The pascal visual object classes (VOC) challenge,” International Journal of Computer Vision, vol. 88, no. 2, pp. 303–338, 2010.