# A template consensus method for visual tracking*

**ZHOU Tong-xue** (周同雪)[1,2,3]**, **ZENG Dong-dong** (曾冬冬)[1,2,3], **ZHU Ming** (朱明)[1,2], and **Arjan Kuijper**[3]

1. *Changchun Institute of Optics, Fine Mechanics and Physics, Chinese Academy of Sciences, Changchun 130033, China*
2. *The University of the Chinese Academy of Sciences, Beijing 100049, China*
3. *Department of Graphic Interactive System and Mathematical and Applied Visual Computing, Fraunhofer Institute for Computer Graphics Research IGD, Darmstadt 64283, Germany*

(Received 7 July 2018; Revised 19 September 2018)



Visual tracking is a challenging problem in computer vision. Recently, correlation filter-based trackers have shown to provide excellent tracking performance. Inspired by a sample consensus approach proposed for foreground detection, which classifies a given pixel as foreground or background based on its similarity to recently observed samples, we present a template consensus tracker based on the kernelized correlation filter (KCF). Instead of keeping only one target appearance model in the KCF, we make a feature pool to keep several target appearance models in our method and predict the new target position by searching for the location of the maximal value of the response maps. Both quantitative and qualitative evaluations are performed on the CVPR2013 tracking benchmark dataset. The results show that our proposed method improves the original KCF tracker by 8.17% in the success plot and 8.11% in the precision plot.

**Document code:** A **Article ID:** 1673-1905(2019)01-0070-5

**DOI** https://doi.org/10.1007/s11801-019-8109-2

Visual tracking is a very classical and popular problem in computer vision with a wide range of applications, such as vehicle navigation, augmented reality, human computer interface and surveillance. A typical scenario of visual tracking is to track an unknown object initialized by a bounding box in subsequent image frames. Despite numerous tacking methods have been proposed in recent years[1-3], creating a generic tracker is still rather challenging due to factors, such as illumination changes, partial or full occlusion, background clutter, complex motion and scale variations.

Recently, correlation filter trackers[4-15] have achieved promising results on the visual tracking benchmark dataset both on accuracy and robustness, while operating at real-time. Henriques et al[6] proposed the circulant structure with kernels (CSK) tracker to explore the structure of the circulant patch to enhance the classifier by the augmentation of negative samples, which adopt the gray feature into the visual tracking. To further boost the performance of CSK tracker, Danelljan et al[7] adopt the color-naming feature into the visual tracking task, which is a powerful feature for the color objects. Based on CSK, Henriques et al[8] introduced the kernelized correlation filter (KCF) into the tracking application and adopted the histogram of oriented gradients (HOG) feature instead of raw pixel to improve both the accuracy and robustness of the tracker. Asha et al[11] proposed to use integral channel features in correlation filter framework with adaptive learning rate to efficiently track the object. Ji et al[12] proposed the correlation filter tracker based on sparse regularization to learn the correlation filter model on a significantly larger set of negative training samples, without worsening the positive samples. F Xu et al[13] proposed a multi-scale kernel correlation filter tracker with feature integration and robust model updater, by integrating HOG and color-naming feature to maintain a more powerful object representation, and principal component analysis (PCA) was applied to boost the computation speed. A kernelized correlation filters tracker with PSR redetection was employed[14] to correct the tracker and restore target tracking once tracking failure is detected. Bibi et al[15] proposed an approach to update the scale of the tracker by maximizing over the posterior distribution of a grid of scales. Altogether, the key idea of correlation filter is that the correlation can be calculated by fast Fourier transform (FFT) to avoid the time-consuming convolution operation. Based on a periodic assumption of the training samples in the target neighborhood and approximating the dense sampling strategy by generating a circulant matrix, a linear classifier or a linear regressor can be learned efficiently in the Fourier domain. From these trackers[11-15], there are some improvements for the correlation filter-based method in some respects, such as feature integration, target multi-scale, adaptive learning rate and redetection. However, this kind of target appearance model is simple,

* This work has been supported by the National Natural Science Foundation of China (No.61401425).

** E-mail: tongxuezhou@gmail.com

rigorous, and prone to drift. Due to factors like fast motion and occlusion, in the detection step, the localization of the target in the next frame might be inaccurate or erroneous, these errors will be propagated to the target appearance model, and the tracker will face the risk of unrecoverable drift. Therefore, using more effective target appearance model is crucial.

Wang et al[16] presented an effective and adaptive background modelling method for foreground detection in both static and dynamic scenes. They keep a cache of a fixed number of lasted observed background values for each pixel and classify an input pixel as background if it matches most of the values stored in its model. Inspired by this work, we present a template consensus method for visual tracking in this paper. In the KCF[6,8], a target appearance model is updated by a fixed learning rate. In our tracker, instead of keeping only one target appearance model, we make a feature map pool to take more target appearance models into consideration and predict the new target position by searching for the location of the maximal response score in the response maps.

To validate our method, we perform extensive experiments on the CVPR2013 tracking benchmark dataset[2] with 50 videos. The results show that the proposed method improves the success plot by 8.17% and the precision plot by 8.11% compared with the original KCF tracker.

Our method is built on the KCF tracker[8], which achieves promising results on visual tracking benchmark dataset[2]. In the following, we briefly introduce the main idea of KCF tracker. Readers can refer to Ref.[8] for more details. In Ref.[8], Henriques et al assumed that the cyclic shift version of base template is able to approximate the dense sampling procedure. For simplicity, suppose that all samples are one dimensional, and $\boldsymbol{x}=[x_1, x_2,\ldots, x_n]$ represents the base template at the first frame, then a cyclic shift of $\boldsymbol{x}$ is represented as $\boldsymbol{Px}=[x_n, x_1, x_2,\ldots, x_{n-1}]$, where $\boldsymbol{P}$ is a permutation matrix. Then, all the circular shifts of template $x$ are given by $\{\boldsymbol{P}^i\boldsymbol{x}|i=1, 2,\ldots, n-1\}$. In the case of the 2-dimensional image patch $x$ of size $M\times N$ pixels, there will be two possible shifting directions, KCF considers all cyclic shifts $x_{m,n}(m, n)\in \{0,1, \ldots, M-1\}\times\{0,1, \ldots, N-1\}$ as the training samples for the classifier. The regression labels $y_{m,n}$ follow a Gaussian function, which take the value 1 for the center of the target and smoothly decay to 0 for any other shifts. The goal of training is to find the optimal weight $w$ that minimizes the squared error over samples $x_{m,n}$ and their regression labels $y_{m,n}$.

$$w=\mathrm{argmin}_w\textstyle\sum_{m,n}|<\varphi(x_{m,n}),\ w>-y_{m,n}|^2+\lambda\|w\|^2, \quad (1)$$

where $\varphi$ is the mapping to the Hilbert space induced by kernel $k$, defining $<\varphi(\boldsymbol{x}),\varphi(\tilde{\boldsymbol{x}})>=k(\boldsymbol{x},\tilde{\boldsymbol{x}})$, and $\lambda$ is a parameter for the regularization term. Using the FFT, the solution $w$ can be expressed as $w=\sum_{m,n}a(m, n)\varphi(x_{m,n})$, and the coefficient $\boldsymbol{a}$ is calculated as:

$$\hat{\boldsymbol{a}}=\frac{\hat{y}}{\hat{k}^{xx}+\lambda}. \quad (2)$$

Note that $\hat{\boldsymbol{a}}$ is a vector and we denote the discrete Fourier transform (DFT) of a vector with a hat. $\hat{\boldsymbol{k}}^{xx}$ is the Gaussian kernel correlation of $\boldsymbol{x}$ with itself in the Fourier domain. For an image patch with $C$ feature channels, we have

$$\boldsymbol{k}^{xx'}=\exp\left(-\frac{1}{\sigma^2}\left(\|\boldsymbol{x}\|^2+\|\boldsymbol{x}'\|^2-2F^{-1}\left(\textstyle\sum_{c=1}^{c}\hat{\boldsymbol{x}}'_c\odot\hat{\boldsymbol{x}}_c^*\right)\right)\right), \quad (3)$$

where $\odot$ is the element-wise product, and $c$ is the index of feature channels.

In the tracking step, a patch $z$ with the same size of $x$ is cropped out in the new frame. With the learned coefficient $\hat{\boldsymbol{a}}$ and the target appearance model $\hat{\boldsymbol{x}}$, the matching scores for all the circular shifts of $\boldsymbol{z}$ can be calculated via

$$\bar{\boldsymbol{y}}=\mathcal{F}^{-1}(\hat{\boldsymbol{a}}\odot\hat{\boldsymbol{k}}^{xz}). \quad (4)$$

Finally, the target's new position is estimated by searching for the location of the maximal value of $\bar{\boldsymbol{y}}$.

We present the technical details of the proposed algorithm. As shown in Fig.1, our method consists of two stages, which are the base KCF tracking stage and the template consensus KCF tracking stage.

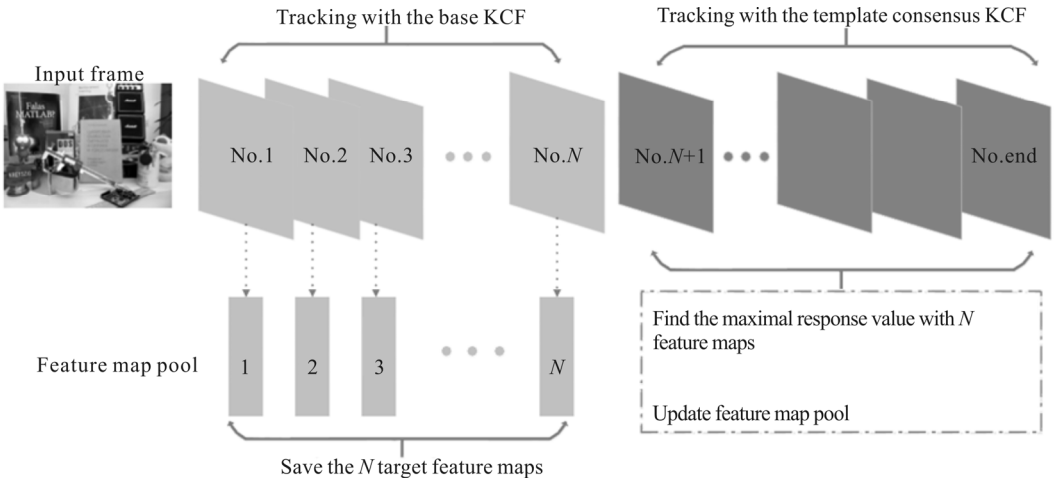



**Fig.1 Main steps of the proposed method, which consists of the base KCF tracking stage and the template consensus KCF tracking stage**

During the base KCF tracking stage, from the second frame to the $N$ frame, the new target position is predicted by the base KCF tracker via Eq.(4). However, a little different from the original KCF, when the target position is updated, we save the target feature map $\hat{\boldsymbol{x}}^{(1)}$ and the corresponding coefficient $\hat{\boldsymbol{a}}^{(1)}$ into the feature map pool at the same time. When the base KCF tracking stage is over, there will be $N$ target feature maps and coefficients saved in the feature map pool. Algorithm 1 summarizes the main steps of this stage.

**Algorithm 1 The base KCF tracking stage**

• *win_sz*: size of the tracked region.
• *pos*: center location of tracking target in spatial domain.
• *patch*: region of *img* centered at *pos* with *win_sz*.
• *feature(x)*: extract the feature of *x* (eg. HOG).
• *cos_window*: cosine window weights.

**for** *img*=1, … , *N*
  **if** *img* >1
    *patch* <— *region* (*img, pos, win_sz*)
    $\hat{\boldsymbol{z}}$ <— $\mathcal{F}$(*feature*(*patch*) *Θ cos_window*)
    $\hat{\boldsymbol{k}}^{\hat{z}\bar{x}}$ <— $\mathcal{F}$(correlation $(\hat{\boldsymbol{z}}, \bar{\hat{\boldsymbol{x}}})$ ) —>Eq.(3)
    *pos*<—*pos*argmax$_{loc}(\bar{\boldsymbol{y}})$ —>Eq.(4)
  **end if**
  *patch* <— *region* (*img, pos, win_sz*)
  $\hat{\boldsymbol{x}}$ <— $\mathcal{F}$(*feature*(*patch*) *Θ cos_window*)
  $\hat{\boldsymbol{k}}^{xx}$ <— $\mathcal{F}$(correlation ( $\hat{\boldsymbol{x}}, \hat{\boldsymbol{x}}$ )) —>Eq.(3)
  $\hat{\boldsymbol{a}} = \hat{\boldsymbol{y}} / (\hat{\boldsymbol{k}}^{xx} + \lambda)$ —>Eq.(2)
  $\bar{\hat{\boldsymbol{x}}}^{(i)} = \hat{\boldsymbol{x}}$ ;
  $\bar{\hat{\boldsymbol{a}}}^{(i)} = \hat{\boldsymbol{a}}$ ;
  **if** first image :$\eta$<—1 **else** $\eta$<— *inter_factor*
  $\bar{\hat{\boldsymbol{a}}}^{(i)} = \eta \times \hat{\boldsymbol{a}} + (1-\eta) \times \bar{\hat{\boldsymbol{a}}}$
  $\bar{\hat{\boldsymbol{x}}} = \eta \times \hat{\boldsymbol{x}} + (1-\eta) \times \bar{\hat{\boldsymbol{x}}}$
**end for**

From the $N$+1 frame to the end of the sequence, the tracking target position is updated by the template consensus KCF tracker. During the target detection stage, we have $N$ target feature maps $\bar{\hat{\boldsymbol{x}}}^{(i)}$ and the corresponding coefficients $\bar{\hat{\boldsymbol{a}}}^{(i)}$ stored in the feature map pool compared with the original KCF which only keeps one target appearance model.

Firstly, a patch $\boldsymbol{z}$ with the same size of $\boldsymbol{x}$ is cropped out in the new frame, and then we calculate all the matching scores of $\boldsymbol{z}$ and $\bar{\hat{\boldsymbol{x}}}^{(i)}$ via

$$\bar{\boldsymbol{y}}^{(i)} = F^{-1}(\bar{\hat{\boldsymbol{a}}}^{(i)} \odot \hat{\boldsymbol{k}}^{\hat{z}x(i)}), i = 1, \cdots, N \quad . \tag{5}$$

Then we obtained $N$ response maps.

We choose the location of the maximal value of $\bar{\boldsymbol{y}}^{(i)}$ as the new target position via

$$\text{argmax}_{loc}\, \text{argmax}_i\, \bar{\boldsymbol{y}}^{(i)}, \quad i=1, 2, \ldots, N. \tag{6}$$

After we got the new target position, we need to update $\bar{\hat{\boldsymbol{x}}}^{(i)}$ and $\bar{\hat{\boldsymbol{a}}}^{(i)}$ in the feature map pool via

$$\bar{\hat{\boldsymbol{a}}}_t^{(i)} = \eta \times \hat{\boldsymbol{a}}_t + (1-\eta) \times \bar{\hat{\boldsymbol{a}}}_{t-1}^{(i)}, \tag{7}$$

$$\bar{\hat{\boldsymbol{x}}}_t^{(i)} = \eta \times \hat{\boldsymbol{x}}_t + (1-\eta) \times \bar{\hat{\boldsymbol{x}}}_{t-1}^{(i)} \quad . \tag{8}$$

Algorithm 2 summarizes the main steps of this stage.

**Algorithm 2 The template consensus KCF tracking stage**

• *win_sz*: size of the tracked region.
• *pos*: center location of tracking target in spatial domain.
• *patch*: region of *img* centered at *pos* with *win_sz*.
• *feature(x)*: extract the feature of *x* (eg. HOG).
• *cos_window*: cosine window weights.

**for** *img*=*N*+1, … , *end*
  *patch* <— *region* (*img, pos, win_sz*)
  $\hat{\boldsymbol{z}}$ <— $\mathcal{F}$(*feature*(*patch*) *Θ cos_window*)
  **for** *i* = 1, ... , *N*
    $\hat{\boldsymbol{k}}^{\hat{z}\bar{x}(i)}$ <— $\mathcal{F}$(correlation $(\hat{\boldsymbol{z}}, \bar{\hat{\boldsymbol{x}}}^{(i)})$ )
    $\boldsymbol{y}^{(i)} = -(\boldsymbol{a}^{(i)} \odot \boldsymbol{k}^{zx(i)})$—> Eq .(5)
  **end for**
  *pos* <— *pos*+argmax$_{loc}$ argmax$_i$ $\bar{\boldsymbol{y}}^{(i=1:N)}$—>Eq.(6)
  *patch* <— *region* (*img, pos, win_sz*)
  $\hat{\boldsymbol{x}}$ <— $\mathcal{F}$(*feature*(*patch*) *Θ cos_window*)
  $\hat{\boldsymbol{k}}^{xx}$ <— $\mathcal{F}$(correlation $(\hat{\boldsymbol{x}}, \hat{\boldsymbol{x}})$ )—>Eq.(3)
  $\hat{\boldsymbol{a}} = \hat{\boldsymbol{y}} / (\hat{\boldsymbol{k}}^{xx} + \lambda)$ —>Eq.(2)
  **for** *i*=1, … ,*N*
  $\bar{\hat{\boldsymbol{a}}}_t^{(i)} = \eta \times \hat{\boldsymbol{a}} + (1-\eta) \times \bar{\hat{\boldsymbol{a}}}^{(i)}$
  $\bar{\hat{\boldsymbol{x}}}^{(i)} = \eta \times \hat{\boldsymbol{x}} + (1-\eta) \times \bar{\hat{\boldsymbol{x}}}^{(i)}$
  **end for**
**end for**

For the experiment, CVPR2013 visual tracking benchmark dataset contains 50 fully annotated sequences and covers a variety of scenarios for visual tracking. These sequences are annotated with the 11 attributes, including illumination variation, scale variation, occlusion, deformation, motion blur, fast motion, in plane rotation, out-of-plane rotation, out-of-view, background clutters and low resolution. Over the past several years, many tracking literature were evaluated on the dataset.

To analyze the performance of our algorithm, we follow the evaluation metrics proposed in Ref.[2], two evaluation methods are used, which are precision plot and success plot. The precision plot counts the percentage of successfully tracked frames based on the center location error (CLE). The success plot presents the percentage of successfully tracked frames, by measuring the intersection over union (IOU) metrics, and the threshold of IOU is varied from 0 to 1. To rank the trackers, two types of ranking metrics are used, which are the precision score whose threshold is set as 20 for the precision plot, and the area under the curve (AUC) metric for the success plot. Both plots and ranking metrics are computed using the software provided by Ref.[2].

The number of feature maps $N$ proposed above controls the balance of sensitivity and precision of the target model. We determine this value based on the experiment results evaluated on the dataset[2]. As it can be seen in Fig.2, the precision score tends to get maximum when $N$ is set to 20, so we determine to set $N$=20 in this paper. The other parameters are the same with the original KCF and keep fixed for all evaluation sequences.

We evaluate our method on the benchmark[2] compared with 7 state-of-the-art trackers, including KCF[8], Struck[17], CN[7], TLD[18], CXT[19], CSK[6] and MIL[20]. For fair evaluations, we compare all the methods following the protocol of the benchmark. We report the results in OPE, SRE and TRE using the distance precision and overlap success rate in Fig.3. It can be seen that

our method attains the best overall performance in all evaluation settings. The proposed method outperforms the original KCF by 8.17% in the success plot and 8.11% in the precision plot. And the results in OPE are in general consistent with those in SRE and TRE.

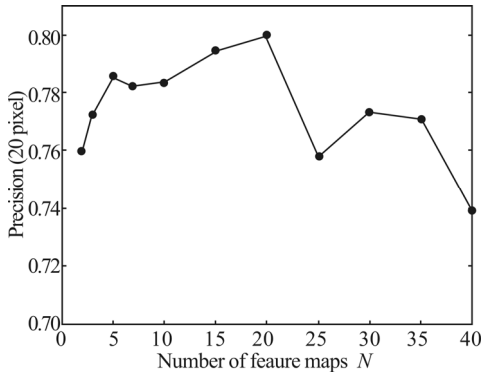
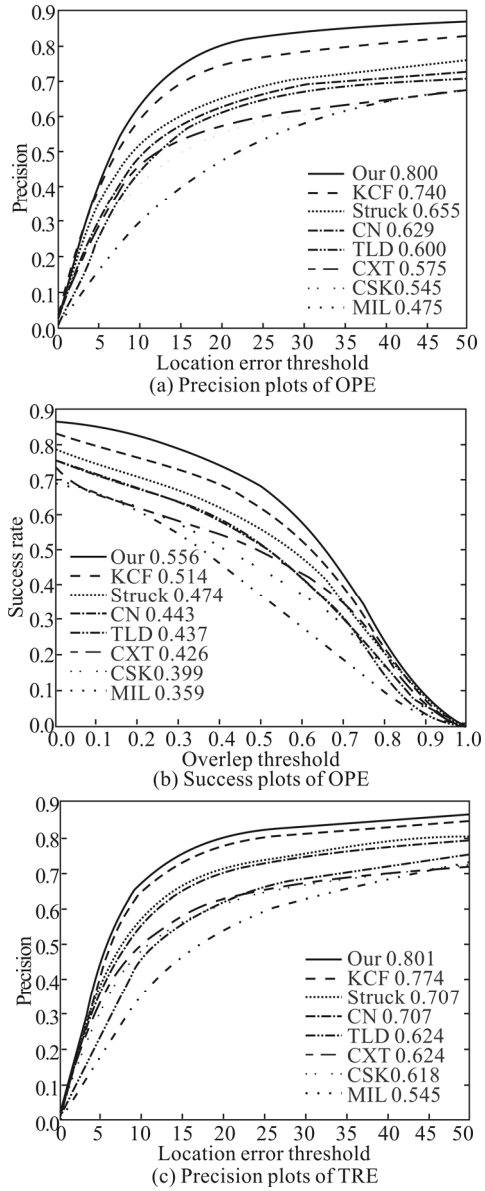



**Fig.2 Precision scores (20 pixels) obtained on the CVPR2013 visual tracking benchmark dataset with different numbers of feature maps**



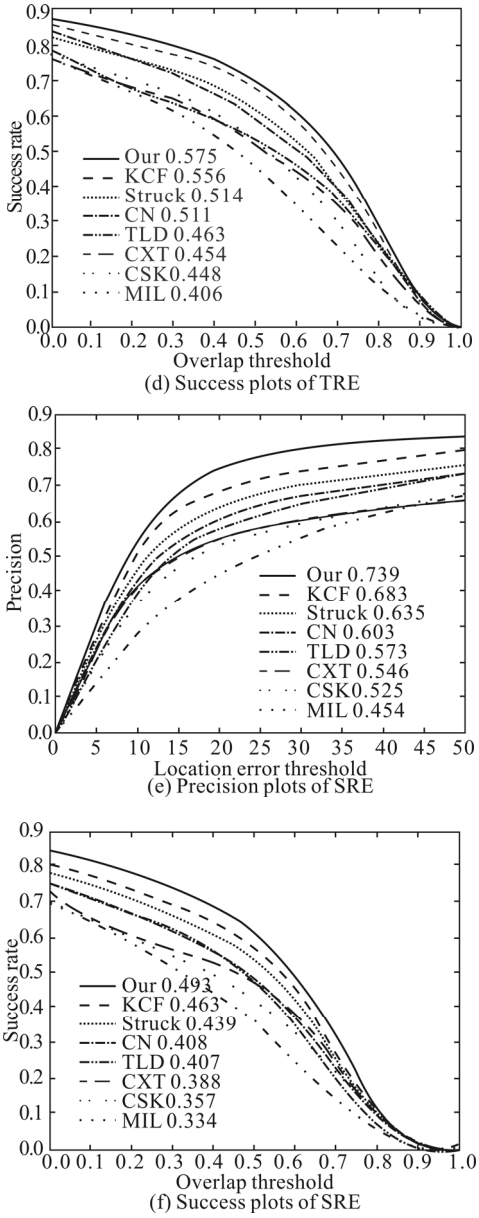




**Fig.3 Distance precision and overlap success plots over 50 sequences using OPE, TRE and SRE**

We also present some tracking results on qualitative evaluation in Fig.4. Ten challenging sequences are selected to validate the effectiveness of our tracker. Overall, our tracker can track the targets more precisely while other methods almost can not deal with these complicated scenarios. The proposed tracker performs well in presence of in-plane rotation, out of view and out-of-plane rotation, which can be explained that our method maintains more target appearance models in the feature map pool.

In addition to the tracking accuracy, the greatest advantage of KCF is the computational efficiency, making it especially suitable for a variety of real-time applications. In our method, saving more target appearance models in the feature map pool will certainly make the

tracker slower, so we make a real time performance comparison between our method, KCF tracker and other excellent trackers evaluated on the CVPR2013 tracking dataset. Tab.1 lists the mean frames per second (FPS) of other trackers and our method in OPE running on a computer with Intel i7 6700K CPU (4.0GHz). We can see that our tracking algorithm also satisfies the requirement of real-time performance.

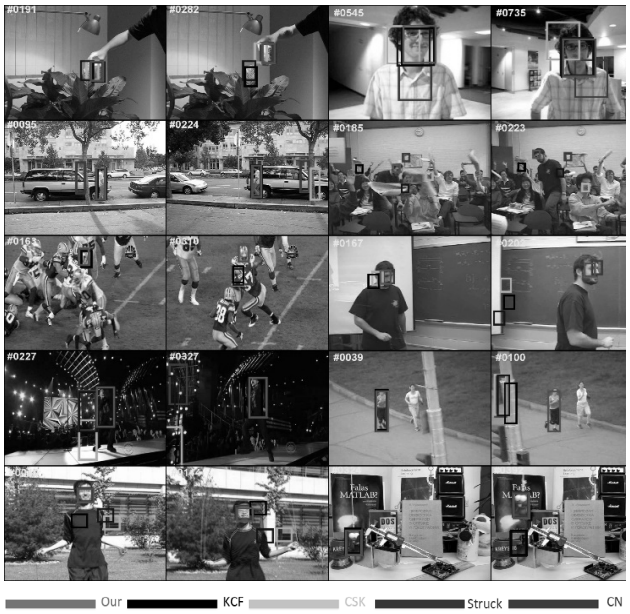



**Fig.4 Tracking results of our method, KCF[8], CSK[7], Struck[12] and CN[6] on 10 challenging sequences (Photos from left to right and top to down are Coke, David, David3, Freeman4, Football, Freeman, Singer2, Jogging, Jumping and Lemming respectively.)**

**Tab.1 Comparison of real time performance**

| Algorithm | Precision (20 pixel) | Mean FPS |
|---|---|---|
| Our | 0.800 | 94 |
| KCF | 0.740 | 437 |
| Struck | 0.656 | 21 |
| CN | 0.629 | 269 |
| CSK | 0.545 | 558 |

In this paper, we propose a novel tracking algorithm based on the KCF. Compared with the original KCF tracker which keeps only one target appearance model and updates with a fixed learning rate, we propose to use the feature map pool to keep more target appearance models and predict the new target position by searching for the location of the maximal value of the response maps. The experimental results on the CVPR2013 tracking benchmark dataset show the superior performance of our method compared with the original KCF tracker.

## References

[1] Alper Yilmaz, Omar Javed and Mubarak Shah, ACM Computing Surveys **38**, 4 (2006).

[2] Yi Wu, Jongwoo Lim and Ming Hsuan Yang, Online Object Tracking: A Benchmark, IEEE Conference on Computer Vision and Pattern Recognition, 2411 (2013).

[3] Arnold W. M. Smeulders, Dung M. Chu, Rita Cucchiara, Simone Calderara, Afshin Dehghan and Mubarak Shah, IEEE Transactions on Pattern Analysis and Machine Intelligence **36**, 1442 (2014).

[4] David S. Bolme, J. Ross Beveridge, Bruce A. Draper and Yui Man Lui, Computer Vision and Pattern Recognition **119**, 2544 (2010).

[5] Martin Danelljan, Gustav Hager, Fahad Shahbaz Khan and Michael Felsberg, Accurate Scale Estimation for Robust Visual Tracking, British Machine Vision Conference, 223 (2014).

[6] Jo. F. Henriques, Rui Caseiro, Pedro Martins and Jorge Batista, Exploiting the circulant structure of tracking-by-detection with kernels, European Conference on Computer Vison, 702 (2012).

[7] M. Danelljan, F. S. Khan, M. Felsberg, and J. Van, de Weijer, Adaptive Color Attributes for Real-Time Visual Tracking, IEEE Conference on Computer Vision and Pattern Recognition, 1090 (2014).

[8] Joao F. Henriques, Caseiro Rui, Pedro Martins and Jorge Batista, IEEE Transactions on Pattern Analysis and Machine Intelligence **37**, 583 (2015).

[9] Yang Li and Jianke Zhu, A Scale Adaptive Kernel Correlation Filter Tracker with Feature Integration, European Conference on Computer Vision, 254 (2015).

[10] A. S. Montero, J. Lang and R. LaganiRe, Scalable Kernel Correlation Filter with Sparse Feature Integration, IEEE International Conference on Computer Vision Workshop, 587 (2015).

[11] Asha C S and Narasimhadhan A V, Procedia Computer Science **89**, 614 (2016).

[12] Ji Zhangjian and Wang Weiqiang, Journal of Visual Communication & Image Representation **55**, 354 (2018).

[13] Xu F, Wang H, Song Y and Liu J, A Multi-Scale Kernel Correlation Filter Tracker with Feature Integration and Robust Model Updater, 29th Chinese Control and Decision Conference, 1934 (2017).

[14] Pan Z, Zhu Y, Computer Engineering & Applications **53**, 196 (2017). (in Chinese)

[15] Bibi A and Ghanem B, Multi-template Scale-Adaptive Kernelized Correlation Filters, IEEE International Conference on Computer Vision Workshop, 613 (2016).

[16] Hanzi Wang and David Suter, Pattern Recognition **40**, 1091 (2007).

[17] Sam Hare, Amir Saffari and Philip H. S Torr, Struck: Structured Output Tracking with Kernels, IEEE International Conference on Computer Vision, 263 (2011).

[18] Zdenek Kalal, Krystian Mikolajczyk and Jiri Matas, IEEE Transactions on Pattern Analysis and Machine Intelligence **34**, 1409 (2012).

[19] Thang Ba Dinh, Nam Vo and G Medioni, Context Tracker: Exploring Supporters and Distracters in Unconstrained Environments, IEEE Conference on Computer Vision and Pattern Recognition, Colorado Springs, 1177 (2011).

[20] Boris Babenko, Ming Hsuan Yang and Serge Belongie, IEEE Transactions on Pattern Analysis and Machine Intelligence **33**, 1619 (2011).