



Channel-independent spatially regularized discriminative correlation filter for visual object tracking

A. Varfolomieiev¹

Received: 25 September 2019 / Accepted: 26 March 2020 / Published online: 22 April 2020
© Springer-Verlag GmbH Germany, part of Springer Nature 2020

Abstract

The study proposes the improvements for visual object trackers based on discriminative correlation filters. These improvements consist in the development of the channel-independent spatially regularized method for filter calculation, which is based on the alternating direction method of multipliers as well as in the use of additional features that are the result of the back-projection of normalized weighted object histogram. The VOT Challenge 2018 benchmark has confirmed that the proposed approaches allow to increase the tracking robustness. Particularly, by the value of expected average overlap (EAO = 0.1828), the tracker that uses these approaches (CISRDCF) can reach the level of more computationally complex competitors that utilize convolutional neural features. At the same time, the software-optimized version of the CISRDCF tracker, which implements the suggested improvements has moderate computational complexity and can operate in the real-time both on the PC and on the mid-range ARM-based processors, making the CISRDCF tracker promising for embedded applications.

Keywords Visual object tracking · Discriminative correlation filter · Alternating direction method of multipliers · Embedded devices

1 Introduction

Visual object tracking is the area in computer vision, which is widely used in robotics, machine–human interfaces, medicine, security systems, advanced driver assistance systems (ADAS), video editing and post-production, etc. In many of the mentioned applications, the tracking should perform in the real-time, and sometimes even on embedded hardware. This requirement restricts a tracking algorithm to have reasonable computational complexity.

Currently, the most promising in terms of robustness and accuracy are the trackers based on discriminative correlation filters (DCF) or on convolutional neural networks, as well as the methods that combine these two approaches [1–4]. In particular, according to the [1–4] benchmarks, the most accurate and robust trackers utilize either convolutional neural networks directly or discriminative correlation filters, which use responses from some layers of convolutional

neural networks as features. At the same time, the mentioned approaches have high computational complexity and usually able to process only several frames per second, even on the most modern PCs with high-end GPUs [5–7]. The only exceptions here are the Siamese neural network trackers [8, 9] that can process up to 100 fps, but also requiring high-performance GPUs. Methods which employ the discriminative correlation filters with so-called “handcrafted” features such as HOG [10–14], colour names or attributes [14–16] have lower robustness and accuracy, but are much faster and allowing to achieve hundreds of frames per second using only CPU of a conventional PC.

Consequently, the discriminative filter-based methods which use handcrafted (non-convolutional) features appear to be still of interest for implementation in embedded systems. Therefore, in this paper, we focus on the improvement of these methods by increasing their robustness and accuracy, while maintaining low computational complexity. The main contribution of this paper consists in fulfilling the following objectives:

1. To develop the approach for calculating the channel-independent spatially regularized discriminative correlation filter by using the alternating direction method

✉ A. Varfolomieiev
a.varfolomieiev@kpi.ua

¹ Igor Sikorsky Kyiv Polytechnic Institute, Faculty of Electronics, National Technical University of Ukraine, 37, Peremogy Prospect, Kyiv, Ukraine

of multipliers (ADMM). This approach combines the channel-independent calculation of the DCF described in paper [14] with the spatial regularization technique given in [12, 17], which results in a specific optimization problem solved within this paper.

2. To employ the additional features, which are based on backprojection of the normalized object histogram. These features are extracted similarly to the ones given in [18], but unlike to [18], we propose using the result of backprojection as an additional channel for DCF filter calculation, which, we believe, should increase the discriminative properties during the object localization.

It is worth noting that one of the peculiarities of the suggested modification is its ability to increase the tracking robustness without using the colour information, which may be important in some cases.

The paper is organized as follows. The section “Related Work” gives a brief overview of tracking methods that preceded and was the basis of the suggested approach. In the section “The Proposed Approach”, we discuss the methods for calculating the modified discriminative filter and extracting the features based on the object histogram backprojection as well as the peculiarities of combining correlation responses obtained from different kinds of features. The section “Experimental Results” presents the details of the implementation of proposed approach (We use CISRDCF below in the paper to refer to this implementation), the analysis of its tracking quality based on VOT Challenge 2018 benchmark [4], and lastly the evaluation of the CISRDCF tracker speed on different computational platforms (including embedded ones).

2 Related work

A discriminative correlation filter (DCF) is a special type of filter, whose correlation (or convolution, in some formulations) with the image gives a sharp response in the region, where the object encoded in this filter is located. Based on the response peak, it is possible to locate the object in each frame of video sequence, which actually makes tracking feasible. The DCF calculation implies solving the least squares minimization problem, which is done in the frequency domain to find this solution fast.

DCFs have become widely used in the visual object tracking since the MOSSE implementation [19] where the simple single-channel filter has been calculated from a grayscale image. Later, the similar approach known as the kernelized correlation filter (KCF) [10, 20] was suggested. This approach introduced the SVM-like feature space kernelization and the use of multichannel FHOG features [21], which increased the discriminative capabilities of the filter and led to significant

tracking quality improvement. The main advantage of both approaches that made them popular was their high frame rate, namely about 700 fps for the MOSSE [19] and 170–290 fps for the KCF [10] with FHOG features. Such high frame rates were achieved mainly due to the calculation of filters in the frequency domain.

Nevertheless, the filter calculation in the frequency domain has a significant drawback, as it was established later in [12, 22, 23]. Due to the periodic nature of Fourier transform, it is impossible to obtain the spatially equivalent filter having non-zero values only in the part that encodes an object. It is believed that such a filter is inaccurate, more sensitive to background variation (since it captures background or so-called “context” information), and consequently gives lower tracking quality. To overcome this difficulty, some modifications of the optimization problem were proposed in [12, 22, 23]. The authors of papers [13, 22, 23] reformulated the minimization problem by introducing the additional constraints, which restricted the filter to have a desirable form (to have non-zero values wherever it is necessary). The problem itself was solved by the proximal gradient descent in [22] and also by the use of the alternating direction method of multipliers (ADMM) in [13, 23]. The constraints in the mentioned approaches are implemented by the special proximal operator that transforms the filter from the frequency to the spatial domain, excludes unnecessary components from the filter (setting them to zero), and returns it back to the frequency domain. This slightly increases the amount of computation, but the asymptotic complexity remains at the level of the KCF and the MOSSE methods. In [12], the authors propose suppressing unnecessary filter components using the regularization. The optimization problem in this case is solved by the Gauss-Seidel method, or in later implementations [5, 6], by the conjugated gradients method. It should be noted that in practice ADMM-based approaches are slightly faster and use less memory, therefore, they are more expedient for fast trackers implementation.

Another way to improve the DCF-based trackers is the use of alternative features [4]. Here, one should note the approaches, which employ colour information [14, 16, 18] as well as the CNN features extracted from certain layers of convolutional neural networks [6, 17, 24, 25]. The CNN features provide a better tracking quality, but at the same time have higher computational complexity making them less appropriate for the real-time application, especially on embedded hardware.

3 The proposed approaches

This section describes in details the algorithm of DCF filter calculation using the ADMM method and the way of extraction and application of the features based on the normalized histogram backprojection.

3.1 Calculation of channel-independent spatially regularized DCF

In order to obtain the DCF filter in the frequency domain so that its spatial counterpart will have suppressed components outside the object region, we used the optimization method, which was employed in [13, 14, 17], particularly the alternating direction method of multipliers (ADMM).

The object region for the tracking is usually given as a rectangle, but, obviously, the shapes of real objects may defer from the rectangular ones. Therefore, it could be reasonable to make the region, which encodes an object in the filter with fuzzy boundaries. It can be done by suppressing the background filter components using the regularization, similarly to the approach given in [17]. However, in contrast to [17], we suggest calculating the filter channels independently. We believe that this should slightly simplify the computations as well as the control of the response merging for different kind of features during the convolution calculation (relying on the experience in [14]). Thus, in this paper, we suggest combining the approaches of channel-independent DCF filter calculation similar to the one from [14], and applying spatial regularization in the formulation given in papers [12, 17]. In particular, it leads to the following optimization problem:

$$\begin{aligned} \arg \min_h \sum_{i=1}^d (\|t_i \star h_i - r\|^2 + \|w \cdot g_i\|^2), \\ \text{s.t. } h - g = 0, \end{aligned} \tag{1}$$

here, $\|\cdot\|^2$ denotes the square of Frobenius norm; “ \star ” denotes convolution between the filter h and the template t , from which this filter is calculated; r is the desired response; w is the regularization matrix that defines which filter components should be suppressed; g also denotes the filter, and the optimization constraint guaranties that $g = h$; operator “ \cdot ” denotes the element-wise multiplication between w and g . The template t is the multichannel image, d is the total number of channels, and subscript i defines the particular channel number.

For the problem (1), the augmented Lagrangian in case of using the scaled dual variable u [26] has the following form:

$$\begin{aligned} \mathcal{L}(h, g, u) = \sum_{i=1}^d (\|t_i \star h_i - r\|^2 + \|w \cdot g_i\|^2 \\ + \frac{\rho}{2} \|h_i - g_i + u_i\|^2 + \frac{\rho}{2} \|u_i\|^2), \end{aligned} \tag{2}$$

where ρ is the penalty parameter; u is the scaled dual variable, which is proportional to Lagrange multipliers [26].

The ADMM method in the scaled form [26] implies the iterative solution of the following optimization sub-problems:

$$\begin{aligned} h^{(k+1)} &= \arg \min_h \sum_{i=1}^d (\|t_i \star h_i - r\|^2 + \frac{\rho}{2} \|h_i - g_i^{(k)} + u_i^{(k)}\|^2), \\ g^{(k+1)} &= \arg \min_g \sum_{i=1}^d (\|w \cdot g_i\|^2 + \frac{\rho}{2} \|h_i^{(k+1)} - g_i + u_i^{(k)}\|^2), \\ u^{(k+1)} &= u^{(k)} + (h^{(k+1)} - g^{(k+1)}), \end{aligned} \tag{3}$$

where superscript (k) denotes the values of respective variables at the k th iteration of the ADMM.

Sub-problem h. Because of convolution, which is present in sub-problem h (3), it is expedient to search the solution in the frequency domain. Using the Parseval’s and the convolution theorems, as well as denoting the respective variables in the frequency domain by the capital letters, the sub-problem h may be rewritten as follows:

$$\begin{aligned} \arg \min_H \sum_{i=1}^d \frac{1}{mn} \sum_{y=1}^m \sum_{x=1}^n (|T_{x,y,i} \cdot H_{x,y,i} - R_{x,y}|^2 \\ + \frac{\rho}{2} |H_{x,y,i} - G_{x,y,i}^{(k)} + U_{x,y,i}^{(k)}|^2), \end{aligned} \tag{4}$$

where m , n and d are the sizes of arrays along each dimension; “ \cdot ” is the element-wise multiplication, which corresponds to the cyclic convolution in the spatial domain; $|\cdot|$ denotes the absolute value of the complex number.

The solution of (4) can be found taking into account the optimization specifics of real functions of complex arguments. One of the most essential things here is that the differentiation of (4) may be performed by the complex-conjugated variable H^* , while the resulting equation may be solved for simple non-conjugated variable H [19, 27]. Moreover, since all operations in (4) are element-wise, the differentiation and solutions of equations may be found for each element independently. Thus, after the derivation and solving the respective equations, we can write the final result for sub-problem h :

$$H^{(k+1)} = \frac{RT^* + \frac{\rho}{2}(G^{(k)} - U^{(k)})}{TT^* + \frac{\rho}{2}}, \tag{5}$$

here, T^* is the complex-conjugated template t in the frequency domain; the operations of multiplication and division are element-wise. Subscripts in formula (5) are omitted for simplicity.

Sub-problem g. Sub-problem g in (3) can be directly solved in spatial domain. For this, we expand the norms:

$$\begin{aligned} \arg \min_g \sum_{i=1}^d \sum_{y=1}^m \sum_{x=1}^n ([w_{x,y} \cdot g_{x,y,i}]^2 \\ + \frac{\rho}{2} [h_{x,y,i}^{(k+1)} - g_{x,y,i} + u_{x,y,i}^{(k)}]^2). \end{aligned} \tag{6}$$

All operations in formula (6) are element-wise, thus, we can again differentiate and solve equations for each array element independently, as we did for previous sub-problem (4). The final result of minimization has the following form (subscripts omitted):

$$g^{(k+1)} = \frac{\rho(h^{(k+1)} + u^{(k)})}{2w^2 + \rho}. \quad (7)$$

It is interesting to note that the solution for the sub-problem g (7) is equivalent to the one, obtained in [17].

Since we search the solution for h in the frequency domain, and the convolution of g during the object localization can be also computed faster in the same domain, it is expedient to transform the solution for g as follows:

$$G^{(k+1)} = \mathcal{F}\left[\mathcal{F}^{-1}\left[H^{(k+1)} + U^{(k)}\right] \cdot \frac{\frac{\rho}{2}}{w^2 + \frac{\rho}{2}}\right], \quad (8)$$

where $\mathcal{F}[\cdot]$ and $\mathcal{F}^{-1}[\cdot]$ denote direct and inverse Fourier transforms respectively. Thus, the formula (8) implements the solution of the sub-problem g in the frequency domain.

Update of the scaled dual variable u can also be performed directly in the frequency domain:

$$U^{(k+1)} = U^{(k)} + (H^{(k+1)} - G^{(k+1)}). \quad (9)$$

In order to achieve the faster convergence, the step scale ρ is usually updated at every iteration of the algorithm. In most existing trackers, this update is performed via the following formula [13, 14, 17]:

$$\rho^{(k+1)} = \min(\tau \cdot \rho^{(k)}, \rho_{\max}), \quad (10)$$

where ρ_{\max} is the maximally allowed penalty parameter (step size); τ is the coefficient of penalty parameter change. Note that if the ADMM method is used in the scaled form, the dual scaled variable U should also be rescaled, when updating the parameter ρ ; particularly if ρ increases τ times, U should be decreased, respectively: $U = U/\tau$ [26].

After performing the required number of iterations, the approximation of the final result is obtained as the solution of sub-problem g , i.e. the filter with the suppressed components will be stored in the $G^{(k+1)}$ variable at the last ADMM iteration.

The computational complexity of the above algorithm [formulas (5), (8–10)] is defined by the most computationally expensive operations, which are direct and inverse Fourier transforms used in (8). Thus, the asymptotic complexity of calculation of the DCF filter is $\mathcal{O}(k \cdot d \cdot mn \cdot \log(mn))$, where mn denotes the resolution (width and height) of the template (search region) in the feature space [the term $mn \cdot \log(mn)$ corresponds to the Fourier transform complexity]; d is the number of

the feature channels (depth); and k is the number of the ADMM iterations.

3.2 Features based on object normalized histogram backprojection

The idea of using the features that utilize the object histogram backprojection was borrowed from the Staple tracker [18]. Generally, this tracker consists of two components running in parallel. The first of them is actually DCF-based tracker with FHOG features, which essentially repeats the approach from [11]. The second component employs histogram of pixel features (namely of quantized RGB colour space). Since we apply the ideas from the second component of the Staple tracker, let's consider it in more details.

This component implies the computation of two histograms: the object histogram and the histogram of the background around the object. Normalizations of both of these histograms give two features distributions for the object $p(O)$ and for the background $p(B)$, respectively. Then, the normalized object histogram β (distribution of feature weights) is calculated using these distributions:

$$\beta_j = \frac{p_j(O)}{p_j(O) + p_j(B) + \lambda}, \quad (11)$$

where j is the histogram bin that is associated with a particular feature (colour); $p_j(R) = N_j(R)/|R|$ is the relative appearance frequency of the feature associated with the j th histogram bin in the region R , here $N_j(R)$ denotes the number of pixels (features) in the region R , which corresponds the j th histogram bin, and $|R|$ is the total number of pixels (features) in the region R ; O and B are the sets of pixels in the object and the background areas, respectively; λ is the regularization parameter, which prevents division by a small value if $p_j(O) + p_j(B) \approx 0$. Accordingly to [18], formula (11) can be considered as the solution of the regression problem, where the β_j values minimize the weights of background and maximize the weights of object features simultaneously; for more details reader may refer to the original paper [18].

In order to locate the object in the current frame using the feature weights β , the weights of every single pixel in the region that is centred on the last known object position are evaluated. It can be done by the replacing each pixel with its respective histogram weight β_j . This procedure is also known as *histogram backprojection* and gives the likelihood map. The search of the object through the likelihood map (backprojection) is performed by the localization of the densest area within this map. The density itself is estimated by the averaging of the map regions using a sliding window of the same size as the object. To achieve higher performance, the average values for each window position are calculated using the integral images [18].

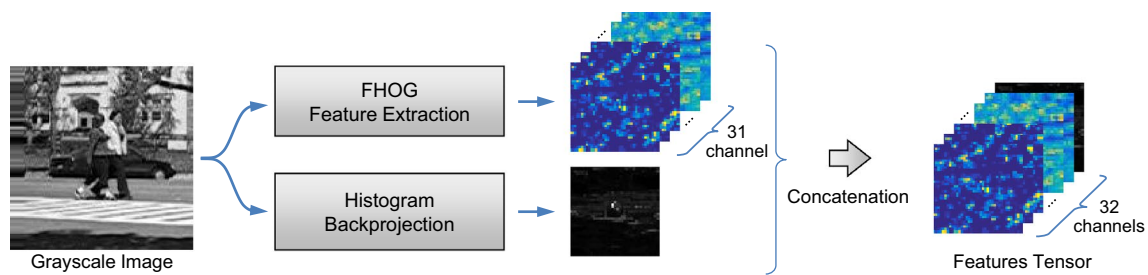


Fig. 1 Feature calculation procedure in the suggested approach: FHOG and histogram backprojection features are extracted from grayscale image and concatenated to a single tensor. This procedure

The averaging of elements in the likelihood map using the sliding window is actually equivalent to the application of the conventional averaging box filter. The averaging makes the tracking invariant to the shape of the object, since the filtering results (mean values) do not depend on any weights β_j (and thus the features) permutations within the filter window. At the same time, this approach neglects the relative spatial positions of features, which, in turn, may decrease the discriminative properties. Therefore, in this paper, we propose replacing the averaging filter with the DCF filter calculated using the ADMM method from the previous subsection. In our opinion, the DCF can take into account spatial positions of the features in the object and thus may deliver a better performance during the tracking. The transition from the old approach to the new one is straightforward: it implies transforming the likelihood map into the frequency domain and considering its frequency image as the template T for the DCF filter calculation procedure; during object localization, the obtained DCF filter should be convolved with the likelihood map extracted from the current frame. In this case, the peak of the response defines the object position, as for the general DCF localization technique. Of course, the likelihood map (backprojected features) can also be considered as one of the channels of the multichannel template image. Thus, the aforementioned approach may be easily integrated into the channel-independent DCF calculation procedure described in the previous subsection.

When applying the suggested backprojected features in conjunction with some features of different type, for instance FHOG, we need to merge the responses taking into account that the FHOG features usually include a set of channels, while the likelihood map (backprojection result) has single channel only. Thus, to obtain the balanced response, we perform the merging similarly to the one given in the original paper [18]:

$$c = c_{\text{HOG}} + \gamma \cdot c_{\text{HBP}}, \quad (12)$$

where c_{HOG} is the total response over all channels of FHOG features; c_{HBP} is the convolution response given by a likelihood map that is obtained from the backprojection of object

is repeated several times per frame: (1) to locate the object at different scales, and (2) when the object is found to calculate a new DCF filter for the region of object location

normalized histogram β ; γ is the merging coefficient, which allows us to control the importance of responses [18].

For a better adaptation to the changes of object appearance during the tracking, it is expedient to update the object normalized histogram β in each frame [18]:

$$\beta^{(t+1)} = (1 - \eta_h) \cdot \beta^{(t)} + \eta_h \cdot \beta, \quad (13)$$

where $\beta^{(t+1)}$ and $\beta^{(t)}$ are the updated normalized histograms for the next and the current frames respectively; β is the normalized histogram for the object located in the current frame; η_h is the update coefficient.

4 Experimental results

4.1 Implementation details

To evaluate the suggested improvements, we implemented the tracker with parameters that are described in details in this section. Below, we refer to this implementation as the CISRDCF tracker.

Our implementation uses both FHOG [21] features and the described above features that are based on the backprojection of the object normalized histogram. FHOG features were calculated for the cell size of 4×4 pixels and 9 orientations. Thus, the complete feature array (tensor) consists of $d = 32$ channels (31 channels of FHOG features and 1 channel obtained from the object histogram backprojection). The detailed procedure of feature extraction is shown in Fig. 1.

We calculate the DCF filter by using (5), (8–10) formulas for each feature channel separately. In these formulas, we employ the Gaussian with the standard deviation of $\sigma = 0.1$ as the desired response r in the spatial domain. We also utilize the matrix with the high values elsewhere outside the object-bounding box as the regularization matrix w (Fig. 2). These high values are taken as equal to 100, while the values inside the bounding box of the object are equal to 0.01. We blur the margin between the high and low values in the matrix w using the Gaussian filter with the size of 7×7 pixels and the

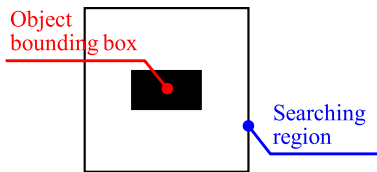


Fig. 2 Generalized view of the regularization matrix w : the dark region contains the low values and corresponds to the object-bounding box in the feature space: the light region contains the high values and defines the area in the filter that have to be suppressed

standard deviation of $\sigma = 0.25$. This technique is applied to make the margin of the suppressed area in the filter smoother, which, we hope, should help when the object position is set slightly inaccurately or when the object has fuzzy edges.

We also use the following parameters in (5), (8–10): initial step size (penalty parameter) $\rho^{(0)} = 1$; maximally allowed step size $\rho_{\max} = 1000$; step size change factor $\tau = 30$. The number of ADMM iterations for each filter calculation is 2. The initial values $G^{(0)}$ and $U^{(0)}$ for the first ADMM iteration were taken from the results of the filter calculation of the previous frame. (In the paper, [26] this approach is called the “warm start”.) We believe that such a technique is applicable because objects tend to change their appearance slightly in adjacent frames during the tracking. For the very first frame, we take $G^{(0)}$ and $U^{(0)}$ equal to 0.

We also apply the exponential filtering of the DCF in each frame, similarly to the commonly used practice [10, 12, 18, 19]. Such an approach allows adapting more efficiently to the changes of object appearance during the tracking:

$$G^{(t+1)} = (1 - \eta_f) \cdot G^{(t)} + \eta_f \cdot G, \quad (14)$$

where $G^{(t+1)}$ and $G^{(t)}$ are the averaged filters for the next and the current frames, respectively; G is the filter, which is calculated for the region where the object was found in the current frame (It is obtained directly from (8) at the last ADMM iteration); η_f is the exponential filtering coefficient. In this paper, we use $\eta_f = 0.025$. For the very first frame, $G^{(0)} = G$.

We calculate the normalized histogram for the localized object position in the current frame and update the histogram for the next frame using the formula (13) in which we use $\eta_h = 0.125$ update coefficient.

We localize the object by calculating convolution (according to the definition of minimization problem (1) in the frequency domain with subsequent transition to the spatial domain:

$$c_i = \mathcal{F}^{-1}[F_i \cdot G_i], \quad (15)$$

where $\mathcal{F}^{-1}[\cdot]$ as earlier denotes the inverse Fourier transform; c_i is the calculated convolution for the i th feature channel in the spatial domain; F_i is the i th feature channel of the search region of the object in the frequency domain; G_i is

the i th channel of the DCF in the frequency domain (filter G here is the $G^{(t+1)}$ from formula (14)).

We merge the responses of the individual channels (calculated by (15)) applying (12), in which we use the coefficient: $\gamma = 0.55 \cdot \max(c_{\text{HOG}}) / \max(c_{\text{HBP}})$. This value is obtained empirically.

To adapt to the change of the object size, we calculate the filter convolution with exemplars of the search region at $n_s = 3$ scales. The scale factors are taken to be $s = 1.03^p$, where $p = \{-(n_s - 1)/2, \dots, +(n_s - 1)/2\}$.

In all further experiments, the search region had the square shape. It was taken approximately 4.25 times larger than the object, and also was downscaled so that in the feature space its resolution was not larger than 38×38 pixels.

4.2 Tracking accuracy and robustness evaluation

We evaluated the quality of the suggested CISRDCF tracker using the VOT Challenge 2018 [4] benchmark intended for the short-term trackers. This benchmark includes 60 annotated video sequences with different objects and with different visual complexity attributes: object occlusion, illumination change, object motion change, scale change, and camera motion [4]. The quality of tracking is estimated by three measures: the accuracy (A), the robustness (R), and the expected average overlap (EAO) [1, 4]. The overlap in the benchmark is assessed by the intersection-over-union value (also known as Jaccard index), which is $\text{IoU} = \frac{|r_t \cap r_{GT}|}{|r_t \cup r_{GT}|}$, where r_t is the location of the object (rectangle) reported by the tracker, and r_{GT} is the ground-truth location [18]. The *accuracy* measures the average overlap for all frames and all video sequences in the benchmark where the tracking was acknowledged successful (robust). The *robustness* is the mean number of tracker re-initializations (fails) caused by the loss of object. The fact of the object loss is established on the basis of the zero overlap value between the bounding box predicted by the tracker and the ground-truth object location. The *expected average overlap* is the average overlap, which the tracker is expected to achieve on a large set of video sequences of some identical length and visual properties [1, 4]. This measure accounts for the increase in the variance and bias of the average overlap for video sequences of variable lengths. The VOT Challenge 2018 results are arranged in accordance with the EAO measure [4].

The VOT Challenge 2018 benchmark includes three sub-challenges: the baseline, the unsupervised, and the real-time ones. The unsupervised sub-challenge does not imply re-initialization when tracker fails, therefore, the only no-reset average overlap (AO) is estimated within this sub-challenge. In the real-time sub-challenge, the frames of video sequences are sent to the tracker with constant framerate of 20 fps, and if the tracker does not respond in time, the last

Table 1 The comparison results of the tracking quality for the suggested CISRDCF tracker and the similar trackers, which were tested in VOT Challenge 2018 benchmark

Tracker	Baseline			Real-time	Unsup.
	EAO \uparrow	A \uparrow	R \downarrow	EAO \uparrow	AO \uparrow
DCFNet	0.1827	0.470	0.543	0.1803	0.3269
DensSiam	<i>0.1732</i>	0.462	0.688	<i>0.1731</i>	0.3037
SAPKLTF	0.1713	0.488	0.613	0.1174	0.3521
Staple	0.1694	0.530	0.688	0.1696	<i>0.3327</i>
ASMS	0.1692	0.494	0.623	0.1669	0.3356
ANT	0.1684	0.464	0.632	0.0590	0.2770
STBACF	0.1548	0.461	0.740	0.0618	0.2447
KCF	0.1349	0.447	0.773	0.1336	0.2671
SRDCF	0.1189	0.490	0.974	0.0584	0.2445
CISRDCF	0.1828	0.426	0.566	0.1774	0.2685

Cells denoted by bold, bolditalic and italic fonts are the first, the second and the third best values in the columns, respectively

\uparrow means higher better

\downarrow means lower better

object position is considered as the tracker output for the current frame [4].

The results of our CISRDCF tracker on the VOT Challenge 2018 benchmark are shown in Table 1. For comparison, we also extended the table with the results of some other trackers of the same class or similar performance. These results are completely consistent with the ones in [4] and were downloaded from the official VOT Challenge web page.

As can be seen from Table 1, the suggested approach (CISRDCF) by the EAO measure is able to surpass some state-of-the-art trackers of the same category: Staple, SRDCF, STBACF, and KCF. In addition, by the same parameter the CISRDCF tracker approaches and even goes

slightly beyond the more powerful neural network-based trackers, such as DCFNet and DensSiam. At the same time, the accuracy of the CISRDCF tracker is in the last place in the baseline sub-challenge (Table 1, A column of the Baseline experiment), while it takes the second place by robustness: right after the DCFNet tracker (Table 1, R column of the Baseline sub-challenge). In the real-time sub-challenge, the EAO of the suggested approach is in the second place (again, after the DCFNet tracker). In the unsupervised sub-challenge, the CISRDCF tracker takes place slightly ahead of STBACF, but behind the ANT tracker (see, last column of the Table 1). It should be noted that the SAPKLTF, ASMS and Staple trackers, which share the first places in the unsupervised sub-challenge, explicitly use colour information, while the suggested CISRDCF tracker employs only the greyscale information.

The expected average overlaps for the tested trackers depending on the sequence length are shown in Fig. 3. From these curves, one can see that the CISRDCF tracker has one of the highest values of EAO for sequences longer than 100 frames. It means that the suggested approach tends to have relatively high precision of object localization for longer sequences in comparison with other tested trackers.

Figure 4 illustrates some examples of the tracking of objects in video sequences from the VOT Challenge [4] that were obtained by the CISRDCF tracker. The first row (a–d) of Fig. 4 shows the sequence with the object, which significantly changes its size during the tracking. The second row (e–h) of Fig. 4 gives the example of the tracking of the small object (drone), which rotates in- and out- of plane of frame during the motion, the object also periodically falls into the regions with a cluttered background (frames g, h). The third row (i–l) illustrates the sequence, where the object moves behind the plant and undergoes significant partial occlusion (frame k). In all these cases, the suggested CISRDCF approach managed to track the objects without failures.

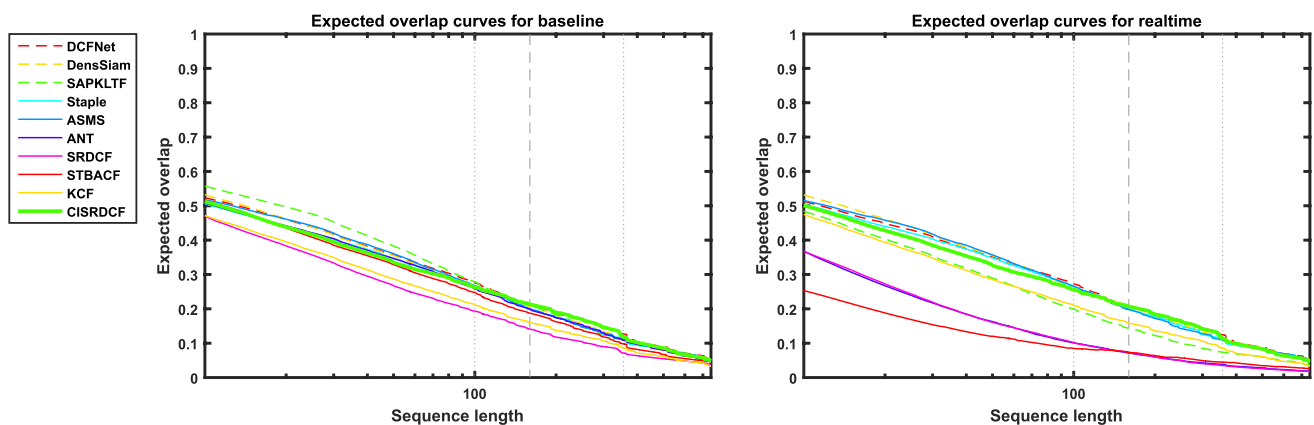


Fig. 3 The expected average overlap curves (EAO depending on the video sequence length) for the trackers from Table 1. The curves are generated by the VOT 2018 benchmark for the baseline and real-time sub-challenges



Fig. 4 Examples of some video sequences from VOT Challenge 2018, where the CISRDCF algorithm tracked objects without fails: green rectangles are the object-bounding boxes reported by the

CISRDCF tracker; red rectangles are the ground-truth; the numbers of the frames in the respective video sequences are shown in the top right-hand corners

The dependencies of the accuracy and robustness measures on the visual attributes for the CISRDCF tracker are shown in Fig. 5. We used the sensitivity parameter of $S = 100$ for robustness estimation similarly to the [4]. This diagram allows us to judge about the influence of visual attributes (parameters of video sequence and object motion) to the work quality of a given tracker. Namely, as it can be seen in Fig. 5, the tracking robustness of CISRDCF evidently drops when illumination changes. At the same time, occlusions and changes of object motion affect the robustness less. Camera motion or change of the object size have the smallest influence on the CISRDCF robustness. We also observe that none of attributes significantly affects the accuracy of the CISRDCF.

During the benchmarking, we used the MATLAB implementation of the CISRDCF tracker. Since MATLAB is an interpreted language, this implementation has reduced performance; thus, we decided to re-implement the tracker in the C++ to estimate its speed more adequately. The next section is dedicated to this issue.

4.3 Speed evaluation

To estimate the speed of the CISRDCF tracker, we re-implemented it using C++. This implementation uses the same parameters that are described in the “Implementation Details” section.

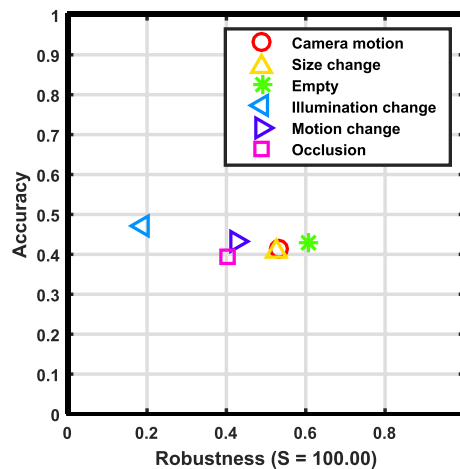


Fig. 5 The VOT Challenge 2018 accuracy–robustness diagram for the CISRDCF tracker depending on different visual attributes

We used the OpenCV library (version 3.4.5) for software implementation of the tracker on both PC and ARM-based platforms. To achieve the highest possible performance, we employed the following software optimization techniques. Firstly, we implemented the modified FHOG features extraction procedure, which defers from the original one¹, mainly

¹ <https://github.com/rbgirshick/voc-dpm/blob/master/features/feature.cc>.

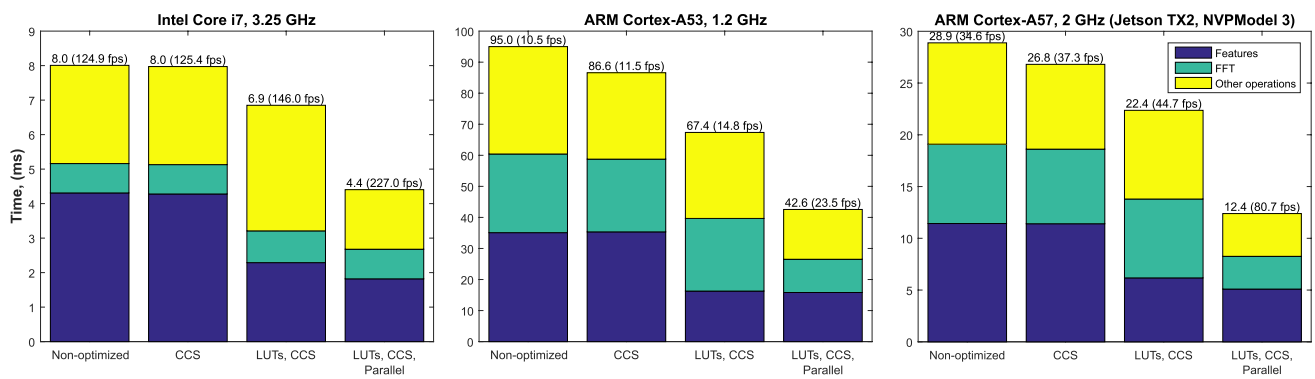


Fig. 6 The impact of suggested optimizations on the average frame processing speed of CISRDCF tracker. The bars in the diagrams denoted as “Non-optimized”, “CCS”, “LUTs, CCS”, and “LUTs, CCS, Parallel” correspond the basic unimproved implementation, the implementation that uses reduced complex-conjugated symmetric spectrum format (CCS format), the implementation that employs

by applying the lookup tables (LUTs) for arctangent and square root calculation during the estimation of orientations and magnitudes of the gradients. It should be noted that the image pixels are usually 8-bit unsigned integers and the number of gradients orientations is small (up to few tens), thus the arctangents and square roots have limited values, which allows making the LUTs of a compact size. Also, we used SIMD instructions for the inverse square root calculation during the histogram bins normalization. For more details on the optimization of FHOG features extraction procedure, reader may refer to [28, 29]. Secondly, we used the twice reduced complex-conjugated symmetric spectrum representations (CCS format), which allowed us to simplify the most matrix operations by the factor of two. And thirdly, since the DCF filter may be calculated for each channel independently, we applied the per-channel parallelization. Moreover, we also computed the features for each scale in parallel during the object localization.

We implemented and tested the CISRDCF tracker on a few platforms including the embedded ones. Fig. 6 illustrates the impact of the mentioned software optimizations on the tracker performance for some selected platforms. The complete results for all tested platforms including parallel implementation are shown in Table 2. During the tests, we used the video sequence, in which the object in the feature space had maximally allowed resolution of 38×38 pixels (according to the implementation details described above). Note that the average frame processing time in Table 2 does not include the time for the image input–output.

As can be seen in Fig. 6, the use of CCS spectrum representation format does not give any notable advantages on a PC, while it increases the performance roughly by 10% on the ARM platform. The application of look-up tables (LUTs) during the feature extraction brings 15–30% improvement

CCS format and look-up tables (LUTs) during feature calculations, and the multi-threaded (parallel) implementation, which uses CCS format and look-up tables, respectively. Each section in the bar is a portion of the time taken by some principal operations: feature extraction, FFT computations, and other operations, including matrix manipulations during DCF filter calculation

in performance on all tested platforms. Again, the results of optimization are more evident on the ARM. Furthermore, the portion of FFT calculations is larger for the ARM platform, which is probably caused by the better OpenCV optimization for the Intel processors (OpenCV employs components of the IPP library). The parallel implementation, together with all mentioned above improvements, increases the tracker performance approximately 2 times on all platforms (compare first and last bars of each diagram).

The performance of CISRDCF tracker on PC in Table 2 is given for the reference. Generally, there are no doubts that the tracker is able to operate in the real-time on modern, but not the most powerful PCs. It was confirmed on the x86 64-bit processor downclocked to 0.78 GHz: when only one thread of this processor is used, the frame rate of 37 fps is achieved (see Table 2, third row, column 1). At the same time, the CISRDCF tracker can operate at approximately 15 fps using a single thread of the mid-range ARM Cortex-A53 processor, which runs at 1.2 GHz (Table 2, fourth row, column 1). When multiple threads in this processor are employed, the frame rate can be increased up to 21–23 fps (see Table 2, ARM Cortex-A53, 4 Cores 1.2 GHz row, column 2), and when it is additionally overclocked to 1.3–1.4 GHz, it is possible to achieve up to 25–26 fps. The more productive platform, such as ARM Cortex-A57, is capable of 27 fps at 1.2 GHz and up to 40 fps at 2.0 GHz without parallelization (see first column of Table 2 for ARM Cortex-A57 processor rows).

It is interesting to note that the performance of the tracker on the Jetson TX2 platform becomes lower, when two additional Denver cores are engaged (see last two rows of the Table 2). It may be caused by the power limitations of the SoC or by some peculiarities of the scheduler of the Linux operating system.

Table 2 Average frame processing time by the C++ version of CISRDCF tracker, which is implemented on different computational platforms

Platform	Performance (ms)	
	Basic implementation	Using parallelization
Intel Core i7, 4 Cores 3.25 GHz	6.85 (146 fps)	4.4 (227 fps)
Intel Core i3, 2 Cores 2.1 GHz	10.9 (91.7 fps)	8.1 (123 fps)
Intel Core i7, 4 Cores 0.78 GHz	26.9 (37.1 fps)	16.9 (59.2 fps)
ARM Cortex-A53, 4 Cores 1.2 GHz	67.5 (14.8 fps)	42.5 (23.5 fps)
ARM Cortex-A53, 4 Cores 1.344 GHz	61 (16.4 fps)	38.8 (25.8 fps)
ARM Cortex-A57, 4 Cores 1.2 GHz*	35.8 (27.9 fps)	19.2 (52.1 fps)
ARM Cortex-A57, 4 Cores 2.0 GHz*	22.4 (44.7 fps)	12.4 (80.6 fps)
ARM Cortex-A57, 4 Cores + Denver, 2 Cores, 2.0 GHz*	23.3 (42.3 fps)	13.8 (72.5 fps)

The two columns of Performance part of the table present the results for the basic non-parallel (single-threaded) and parallel (multi-threaded) implementations of the algorithm, respectively

* Tested on Nvidia Jetson TX2

5 Conclusions

In this paper, we have suggested and tested the modifications for the tracking methods based on discriminative correlation filters (DCF). These modifications consist in introducing of the channel-independent spatially regularized method for calculation of the DCF filter using the alternating direction method of multipliers (ADMM), and also in applying the features based on the backprojection of normalized object histogram. The study has shown benefits of using the proposed modifications that allow increasing the tracking performance in terms of robustness, which was confirmed on the VOT Challenge 2018 benchmark. We have shown that the CISRDCF tracker that uses the proposed improvements achieves the expected average overlap measure of EAO = 0.1828, surpassing some state-of-the-art methods of the same category, and even putting it on a par with certain more powerful, but at the same time, more computationally complex approaches based on convolutional neural features. It should be emphasized that the CISRDCF tracker does not use any colour information, which may be important for some applications. The experiments also confirmed that the C++ optimized implementation of the suggested CISRDCF tracker can process frames in a real-time on modern PCs (from 37 fps when using single thread of downclocked x86 processor to 220 fps on the same processor at the full speed) as well as on embedded platforms, which are based on the mid-range ARM processors (at least at 15 fps in a single thread of Cortex-A53 processor).

Compliance with ethical standards

Conflict of interest The authors declare that they have no conflict of interest.

References

- Kristan, M., Matas, J., Leonardis, A., Felsberg, M., et al: The visual object tracking VOT2015 challenge results. In: IEEE International Conference on Computer Vision Workshop (ICCVW), pp. 564–586 (2015). <https://doi.org/10.1109/ICCVW.2015.79>
- Matej, K., Leonardis, A., Matas, J., et al: The visual object tracking VOT2016 challenge results. In: European Conference on Computer Vision – ECCV 2016 Workshops. ECCV 2016. Lecture Notes in Computer Science, vol. 9914, pp. 777–823 (2016). https://doi.org/10.1007/978-3-319-48881-3_54
- Kristan, M., Leonardis, A., Matas, J., Felsberg, M., et al: The visual object tracking VOT2017 challenge results. In: ICCV2017 Workshops, Workshop on visual object tracking challenge (2017). <https://doi.org/10.1109/ICCVW.2017.230>
- Kristan, M., Leonardis, A., Matas, J., Felsberg, M., et al: The sixth visual object tracking VOT2018 challenge results. In: Computer Vision – ECCV 2018 Workshops. ECCV 2018. Lecture Notes in Computer Science, vol. 11129, pp. 3–53 (2018). https://doi.org/10.1007/978-3-030-11009-3_1
- Danelljan, M., Robinson, A., Khan, F.S., Felsberg, M.: Beyond correlation filters: learning continuous convolution operators for visual tracking. In: 14th European Conference on Computer Vision – ECCV 2016. ECCV 2016. Lecture Notes in Computer Science, vol. 9909, pp. 472–488 (2016). https://doi.org/10.1007/978-3-319-46454-1_29
- Danelljan, M., Bhat, G., Khan, F.S., Felsberg, M.: ECO: efficient convolution operators for tracking. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 6931–6939 (2017). <https://doi.org/10.1109/CVPR.2017.733>
- Nam, H., Han, B.: Learning multi-domain convolutional neural networks for visual tracking. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 4293–4302 (2016). <https://doi.org/10.1109/CVPR.2016.465>
- Bertinetto, L., Valmadre, J., Henriques, J.F., Vedaldi, A., Torr, P.H.S.: Fully-convolutional siamese networks for object tracking. In: European Conference on Computer Vision – ECCV 2016 Workshops. ECCV 2016. Lecture Notes in Computer Science, vol. 9914, pp. 850–865 (2016). https://doi.org/10.1007/978-3-319-48881-3_56
- Li, B., Yan, J., Wu, W., Zhu, Z., Hu, X.: High performance visual tracking with siamese region proposal network. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 8971–8980 (2018). <https://doi.org/10.1109/CVPR.2018.00935>

10. Henriques, J.F., Caseiro, R., Martins, P., Batista, J.: High-speed tracking with kernelized correlation filters. *IEEE Trans. Pattern Anal. Mach. Intell.* **37**(3), 583–596 (2015). <https://doi.org/10.1109/TPAMI.2014.2345390>
11. Danelljan, M., HÅd'ger, G., Khan, F., Felsberg, M.: Discriminative scale space tracking. *IEEE Trans. Pattern Anal. Mach. Intell.* **39**(8), 1561–1575 (2017). <https://doi.org/10.1109/TPAMI.2016.2609928>
12. Danelljan, M., HÅd'ger, G., Khan, F., Felsberg, M.: Learning spatially regularized correlation filters for visual tracking. In: *IEEE International Conference on Computer Vision (ICCV)*, pp. 4310–4318 (2015). <https://doi.org/10.1109/ICCV.2015.490>
13. Galoogahi, H.K., Fagg, A., Lucey, S.: Learning background-aware correlation filters for visual tracking. In: *IEEE International Conference on Computer Vision (ICCV)*, pp. 1144–1152 (2017). <https://doi.org/10.1109/ICCV.2017.129>
14. Lukežič, A., Vojřĩ, T., Čehovin, Z.L., Matas, J., Kristan, M.: Discriminative correlation filter tracker with channel and spatial reliability. *Int. J. Comput. Vis.* **126**, 671–688 (2018). <https://doi.org/10.1007/s11263-017-1061-3>
15. van de Weijer, J., Schmid, C., Verbeek, J., Larlus, D.: Learning color names for real world applications. *IEEE Trans. Image Process.* **18**(7), 1512–1523 (2009). <https://doi.org/10.1109/TIP.2009.2019809>
16. Danelljan, M., Khan, F.S., Felsberg, M., Weijer, J.: Adaptive color attributes for real-time visual tracking. In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1090–1097 (2014). <https://doi.org/10.1109/CVPR.2014.143>
17. Li, Feng, Tian, C., Zuo, W., Zhang, Lei, Yang, M.-H.: Learning spatial-temporal regularized correlation filters for visual tracking. In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 4904–4913 (2018). <https://doi.org/10.1109/CVPR.2018.00515>
18. Bertinetto, L., Valmadre, J., Golodetz, S., Miksik, O., Torr, P.H.S.: Staple: Complementary learners for real-time tracking. In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1401–1409 (2016). <https://doi.org/10.1109/CVPR.2016.156>
19. Bolme, D.S., Beveridge, R.J., Draper, B.A., Lui, Y.M.: Visual object tracking using adaptive correlation filters. In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2544–2550 (2010). <https://doi.org/10.1109/CVPR.2010.5539960>
20. Hannuna, S., Camplani, M., Hall, J., et al.: DS-KCF: a real-time tracker for RGB-D data. *J. Real-Time Image Proc.* **16**, 1439–1458 (2019). <https://doi.org/10.1007/s11554-016-0654-3>
21. Felzenszwalb, P.F., Girshick, R.B., McAllester, D., Ramanan, D.: Object detection with discriminatively trained part based models. *IEEE Trans. Pattern Anal. Mach. Intell.* **32**(9), 1627–1645 (2010). <https://doi.org/10.1109/TPAMI.2009.167>
22. Fernandez, J.A., Boddeti, V.N., Rodriguez, A., Kumar, B.V.K.V.: Zero-aliasing correlation filters for object recognition. *IEEE Trans. Pattern Anal. Mach. Intell.* **37**(8), 1702–1715 (2014). <https://doi.org/10.1109/TPAMI.2014.2375215>
23. Galoogahi, H.K., Sim, T., Lucey, S.: Correlation filters with limited boundaries. In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 4630–4638 (2015). <https://doi.org/10.1109/CVPR.2015.7299094>
24. Danelljan, M., HÅd'ger, G., Khan, F., Felsberg, M.: Convolutional features for correlation filter based visual tracking. In: *IEEE International Conference on Computer Vision Workshop (ICCVW)*, pp. 621–629 (2015). <https://doi.org/10.1109/ICCVW.2015.84>
25. Gundogdu, E., Alatan, A.A.: Good features to correlate for visual tracking. *IEEE Trans. Image Process.* **27**(5), 2526–2540 (2018). <https://doi.org/10.1109/TIP.2018.2806280>
26. Boyd, S., Parikh, N., Chu, E., Peleato, B., Eckstein, J.: Distributed optimization and statistical learning via the alternating direction method of multipliers. *Found. Trends Mach. Learn.* **3**(1), 1–122 (2010). <https://doi.org/10.1561/22000000016>
27. Messerschmitt, D.: Stationary points of a real-valued function of a complex variable. *Tech. Report No. UCB/EECS-2006-93*. EECS, U.C. Berkeley (2006)
28. Varfolomieiev, A., Lysenko, O.: Modification of the KCF tracking method for implementation on embedded hardware platforms. In: *International Conference Radio Electronics & Info Communications (UkrMiCo)* (2016). <https://doi.org/10.1109/UkrMiCo.2016.7739644>
29. Varfolomieiev, A., Lysenko, O.: A simple way to broaden objects search area for tracking methods based on discriminative correlation filters. In: *IEEE First Ukraine Conference on Electrical and Computer Engineering (UKRCON)*, pp. 1149–1154 (2017). <https://doi.org/10.1109/UKRCON.2017.8100430>

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Anton Varfolomieiev Received Ph.D. degree in Computer Systems and Components (2014) from the Vinnytsia National Technical University, M.Sc. degree (2010) and B.Sc. degree (2008) in Electronic Devices from the National Technical University of Ukraine “Kyiv Polytechnic Institute”. Currently he is associate professor in the Department of Design of Electronic Digital Equipment (DEDEC) at the Faculty of Electronics of NTUU “Igor Sikorsky Kyiv Polytechnic Institute”. Scientific interests: visual object recognition and tracking, digital image and signal processing, embedded devices design.