Aerial tracking, which has received widespread attention and exhibited excellent performance, is one of the most active applications in the remote sensing field. In particular, an unmanned aerial vehicle (UAV)-based remote sensing system equipped with visual tracking has been widely used in aviation, navigation, agriculture, transportation, public security, and so on. The UAV-based aerial tracking platform has gradually developed from the research stage to the stage of practical application, establishing itself as one of the main aerial remote sensing technologies of the future. However, due to severe real-world situations, e.g., harsh external challenges, the vibration of a UAV’s mechanical structure (especially under strong wind conditions), its maneuvering flight in complex environments, and its limited onboard computational resources...
as well as its accuracy, robustness, and high efficiency are all crucial for onboard tracking methods.

Recently, discriminative correlation filter (DCF)-based trackers have stood out for their high computational efficiency and appealing robustness on a single CPU and have flourished in the UAV visual tracking community. In this article, the basic framework of DCF-based trackers is first generalized, based upon which 23 state-of-the-art DCF-based trackers are summarized according to their innovations for solving various issues. Also, exhaustive and quantitative experiments have been performed on various prevailing UAV tracking benchmarks, i.e., UAV123, UAV123@10fps [19], UAV20L [19], UAVDT [20], DTB70 [18], and VisDrone2019-SOT [21], which contain 371,903 frames in total. The experiments show the performance, verify the feasibility, and demonstrate the current challenges of DCF-based onboard UAV trackers. This work also implements DCF-based trackers on a typical CPU-based onboard PC to achieve real-flight UAV tracking tests to further validate their real-time capabilities and robustness under challenging scenes and provides a concise summary of future research trends in the area of DCF-based methods for UAV tracking as well as a comprehensive survey of directions for future research.

**MOTIVATION AND CONTRIBUTION**

Aerial visual object tracking is attracting considerable attention and developing rapidly in the field of remote sensing [1]-[5]. In particular, when equipped with visual tracking techniques, widely used UAV platforms [6]-[8] having small size, flexible motion, and high safety have flourished in many applications, e.g., wildlife rescue [9], target following [10]-[12], vehicle tracking [1], disaster response [13], [14], cinematography [15], infrastructure inspection [16], [17], and so forth. Specifically, a mobile UAV usually needs to continuously locate (and follow) one certain object, where a real-time, robust single-object tracking (SOT) algorithm is essential. Nevertheless, under complex scenes on board UAVs, achieving robust, accurate, and real-time tracking is a very challenging task. Compared with general tracking scenes, where the camera is usually static or slow moving and fewer geometric and photometric variations exist, UAV tracking is confronted with more onerous challenges:

- **Inadequate sampling resolution:** The large visual scope of UAVs results in more background information, leading to reduced object resolution and, hence, weak model representations. Weak model representation can make it easy for the tracker to lose the object because of its poor discriminative ability.

- **Fast motion (FM) issue:** UAVs have a large degree of freedom and a high degree of mobility; thus, the FM of both the UAV and the tracking object brings even greater challenges to the tracking task. Moreover, in the process of flight, UAVs usually inevitably encounter mechanical vibration, especially under the influence of strong winds, which may even result in motion blur. Such rapid changes to the object’s location can be more challenging for trackers.

- **Severe visual occlusion (OCC):** A common phenomenon in UAV tracking, partial or even full OCC (FOC) may cause severe degradation of the object, leading to tracking failure.

- **Acute illumination variation (IV):** The illumination conditions for UAVs can change rapidly, including from bright to dim and from indoors, canopy, and shadow regions to bright outdoors or even direct sunlight. Such scenes can cause a large variety of object appearances, thus making tracking challenging. Further, complex and harsh scenes are often encountered, such as the poor illumination conditions at night or during rainy or foggy days, which make it difficult for the tracker to distinguish the object from the background.

- **Viewpoint change (VC):** Generally, UAVs may fly 360° around the object, which makes the onboard camera capture different aspects of the 3D object, e.g., the back and front sides, where the object’s appearance can undergo severe variations. Under such scenes, trackers may lose the object without timely online learning and model updates.

- **Scarce computational resources:** Because of their limited power supply and payload issues, most UAVs use only a single CPU as the computing platform, which greatly limits the processing speed. To meet the real-time requirements of UAV tracking, the methods need to be carefully designed to realize robust tracking without casting aside high efficiency. In addition, the algorithms need to be lightweight enough to leave more power supplies for energy-consuming applications, like maneuvering the flight of a UAV in complex environments.

Figure 1 shows representative and challenging UAV-based aerial tracking scenes. Due to the challenges mentioned previously, the R&D of a fast and robust visual tracking approach is extremely critical and valuable for the prospect of UAV-based remote sensing applications.

Among the various tracking methods, DCF-based trackers usually possess both high speed and accuracy. The most important characteristic as well as an amazing highlight of DCF-based methods is that they transform the calculation of cyclic correlation or convolution in the spatial domain to element-wise multiplication in the frequency domain through discrete Fourier transform (DFT). Such a strategy greatly enhances the operation speed of DCF-based trackers, most of which reach more than 30 fps on a single CPU platform, thus meeting the real-time requirements of a UAV. Due to the excellent and extraordinary performance of DCF-based trackers on a single CPU platform, the past years have witnessed the rapid expansion and favorable results of DCF-based trackers’ onboard UAV tracking [5], [22]-[27].

Even though there has been some research mentioning or summarizing DCF-based object tracking [28], [29], it has paid little attention to UAV-based aerial tracking scenarios (which are more complex, challenging, and resource limited). Besides, until now, most studies reviewing UAVs in
the field of remote sensing have either been general reviews concerning the applications of the UAV platforms [30], [31] or detailed strategies used to control UAVs [32], [33], neither of which focuses on the robust, accurate, and real-time tracking methods under complex scenes on board UAVs.

To the best of our knowledge, there exist very few reviews about UAV real-time tracking in recent years, let alone any that focus on the performance of DCF-based trackers in UAV scenarios. In other words, a systematic and comprehensive review concerning the DCF-based tracking algorithms applied to a UAV platform has not yet been conducted. DCF-based trackers have been utilized widely in the UAV tracking community, and the amount of coverage in related publications is currently increasing remarkably, so it appears that a systematic summary and analysis is necessary to gain a comprehensive and objective understanding of the superiority and practicability of DCFs for UAV tracking. Therefore, this article offers an overall review of the DCF-based tracking algorithms of the past decade, introducing their special contributions. This work has also performed extensive experiments and analyzed the advanced representative DCF-based trackers on various authoritative UAV benchmarks [18]–[21] to demonstrate their reliability and superiority in aerial tracking.

The extensive experiment results confirm the excellence of DCF-based algorithms in UAV tracking, and even more exciting prospects can be expected for DCFs in UAV tracking. Based on the results, we also provide the key bottlenecks of DCF-based methods. DCF-based trackers [22], [24] are also implemented on a typical CPU-based onboard PC, i.e.,

![Representative scenes in six well-known UAV tracking benchmarks.](image)

**FIGURE 1.** Representative scenes in six well-known UAV tracking benchmarks. The benchmarks are (a) DTB70 [18], (b) UAV20L [19], (c) UAV123 [19], (d) UAV123@10fps [19], (e) UAVDT [20], and (f) VisDrone2019-SOT [21]. The target ground truths are marked with red rectangles. The common challenges in UAV tracking are shown, e.g., low resolution, FM, similar objects, VC, IV, and OCC in UAV123@10fps (from left to right).
an Intel NUC8i7HVK, to achieve the UAV tracking tests. The potential directions are also indicated in this work, guiding further research into DCFs for UAV tracking.

The main contributions of this work are fourfold, formally summarized as follows:

- **Comprehensive review**: This work offers a general framework of DCF-based trackers and summarizes many state-of-the-art DCF-based trackers according to their innovations and contributions.
- **Code library**: This work integrates most of the publicly available DCF-based trackers into one code library. In addition, our experimental results are also organized for convenient reference. The code library and experimental evaluation results are located at https://github.com/vision4robotics/DCFTracking4UAV.
- **Experiment evaluation**: This work provides exhaustive experiments of DCF-based trackers on six authoritative UAV benchmarks, i.e., UAV123, UAV123@10fps, UAV20L, UAVDT, DTB70, and VisDrone2019-SOT, to demonstrate their performance under complex scenes and their superiority against other types of trackers. The code library and experimental evaluation results are located at https://github.com/vision4robotics/DCFTracking4UAV.
- **Onboard test**: This work further implements DCF-based trackers onto a typical CPU-based onboard PC to achieve real-flight UAV tracking tests, where their real-time capabilities and robustness under harsh scenes are validated.

**RELATED WORKS**

**OBJECT TRACKING METHODS**

According to different representation schemes, object-tracking methods can be generally divided into two types: generative [34]–[36] and discriminative [37]–[39]. The following sections detail both methods.

**GENERATIVE METHOD**

The main idea of the generative method is to learn a feature template from the target area in the first frame and to find the most similar matching area to the template appearance in the search region of the subsequent frames as the tracking result. As is known, most of the early object tracking methods are generative. For instance, Lucas and Kanade [40] proposed holistic templates, which are based on raw intensity values. To cope with appearance changes, subspace-based tracking methods emerged [34], [41], [42]. As a well-known branch in the generative method, numerous tracking approaches based on sparse representation have also grabbed researchers’ attention [35], [43]–[45].

Nevertheless, the weaknesses of generative methods are obvious. The first is that a large number of training samples requires abundant computational resources, which makes it difficult to meet real-time requirements on board a UAV. Then, traditional generative approaches have neglected the background information, which may help ensure more robust tracking. Third, the generative methods usually suppose that the appearance of the object will not undergo large changes in a period while appearance variation happens frequently in UAV tracking scenes. Recently, the discriminative method has become common within the visual tracking community.

**DISCRIMINATIVE METHOD**

Unlike the generative method, the core idea of the discriminative method (also called the tracking-by-detection method) is to train a classifier that can distinguish the tracking object from the background.

Belonging to the discriminative method, support vector machine (SVM)-based approaches are the first ones to flourish in the visual tracking community [37], [46]–[48]. More concretely, Avidan [37] constructed pyramids from the support vectors and used a coarse-to-fine approach in the classification stage in consideration of large motion issues. Bai and Tang [48] presented an online Laplacian-ranking support vector tracker to robustly locate the object.

Ning et al. [46] proposed a simple yet effective dual linear-structured SVM to boost tracking efficiency. STRUCK [47], proposed by Hare et al., achieved great tracking results due to its kernelized structured-output SVM, which is utilized to provide adaptive tracking. Although the performances of the trackers based on an SVM are promising, large-scale training samples will consume much of the machine’s memory and computing time, greatly limiting their real-time capabilities on board a UAV.

Another competitive branch in the discriminative method is based on multiple-instance learning (MIL) [49]–[52]. In [49], Babenko et al. showed that, compared with traditional supervised learning methods, MIL can enable a more robust tracker with even fewer parameter tweaks. However, a serious problem in MIL is the instability of the sample labels. In other words, if there is a slight change in the training set (which is frequent in UAV tracking situations), the output sample labels can undergo drastic changes, ending up with poor robustness.

It is worth mentioning that among the previous trackers, Zhang et al. proposed CT [53], which employed a simple yet efficient appearance method and used a Bayes classifier in the compressed domain for tracking. The speed of CT is appreciable; nevertheless, its robustness and accuracy gradually fell behind the ranks.

With the development of the convolutional neural network (CNN) in recent years, applying one for object tracking, which we call a deep tracker, has become a research hot spot [54]. Specifically, such methods usually train a CNN specifically for object tracking using a large number of labeled images offline.

While tracking the object, for one type of deep trackers, the object’s template and search region are input into the network simultaneously, and the object’s location and size in the search region are predicted directly end to end, e.g., in [38] and [55]–[60]. Held et al., to name a few [38], have used a simple feedforward network structure that was amazingly efficient but still made it difficult to achieve real-time
requirements on the UAV’s platform. Apart from the past work, Bertinetto et al. introduced a novel fully convolutional Siamese network (SiamFC) [57]. The network is completely trained end to end offline, avoiding online updates of its parameter. Constructed on the general framework of the SiamFC [57], Guo et al. proposed a dynamic Siam (DSiam) [58], where the fast transformation learning model can handle an object’s appearance variation efficiently.

Unlike previous anchor-free solutions [57], [58], Li et al. proposed a Siamese region proposal network (RPN) [55], where the Siamese network is combined with the RPN proposed in a Fast R-CNN object detector [61]. In this earliest anchor-based Siam, traditional multiscale testing and online fine-tuning can be abandoned, which greatly improves its speed and settles the aspect ratio change issue.

It is worth mentioning that Danelljan et al. creatively proposed ATOM [56], where target tracking is divided into two stages, classification and estimation, unlike the Siamese series methods. The first stage distinguishes the object from its background for rough localization. Aimed at fine bounding-box estimation, the second stage creatively utilizes the intersection over union (IoU)-net [62], which is trained offline with large-scale data sets to maximize the IoU of ground truth. Among them, some brilliant trackers are designed especially for long-term tracking. Recently, Yan et al. [59] proposed a novel tracker based on skimming and perusal modules. The innovative perusal module estimates target state, enabling the tracker to determine whether the object disappears or not and so decide whether to perform a global or local search.

In [63], Dai et al. creatively offline trained a metaupdater, which learns binary outputs to inform the tracker whether to update or not, greatly settling an updating issue in long-term tracking. The other type utilizes a CNN to extract deep features of the object for model training and object detection, e.g., in [64]–[68].

Although deep trackers have achieved promising results recently, they are generally implemented on high-performance GPUs due to the high complexity of convolutional operation in the networks, which is unable to be supported on board a UAV. Besides, the offline training process requires a large number of pretreated UAV tracking images with annotations, which are hard to obtain. Moreover, the deep network easily loses efficacy when faced with imperceptible noises [69]. Therefore, it is not an ideal approach for UAV-based aerial tracking.

Among the various trackers, DCF-based ones [39], [66], [70]–[73] stand out due to their efficiency and accuracy, making them suitable for UAV tracking. Compared with end-to-end methods, DCF-based approaches [5], [22]–[27] are more applicable to UAV platforms due to their high computational efficiency. Figure 2 displays DCF-based trackers’ performances against deep trackers in terms of success rate, precision, and tracking speed with the UAVDT [20] benchmark. The next section presents the core idea, major steps, and basic framework of DCF-based approaches.

### DCF-BASED TRACKERS

#### CORE IDEA

The core idea of DCF-based trackers is to train a filter with the ability to classify and score search samples by minimizing the loss between the labels and the cyclic correlation between the samples and filters. The classification results of the filter on the search samples can be obtained using the following formula:

\[
g = w \cdot x,\]

where \( \cdot \) symbolizes the cyclic correlation operator and \( g \) is the cyclic correlation between signals; e.g., in DCF-based tracking methods, \( w \) denotes the filter, \( x \) indicates the search samples, and \( g \) is the response map. To speed up the cyclic convolution calculation, cyclic correlation is computed in the Fourier domain using a DFT, which can be expressed as

\[
\hat{g} = \mathcal{F}(w) \odot \mathcal{F}(x),
\]

where \( \odot \) denotes element-wise multiplication and \( \cdot \) indicates the complex conjugate. A feature common to all DCF-based trackers is that, by substituting the operation with an efficient element-wise multiplication, the computational complexity can be substantially reduced.

#### GENERAL STRUCTURE AND MAJOR STEPS

In general, nearly all DCF-based trackers follow a similar structure, which mainly includes three steps: a training stage, model update, and detection stage. As presented in Figure 3, in the training stage, the training image patch is first cropped near the current object’s center. Second, the feature of the image patch is extracted. Then the extracted feature \( x_t \) is used as the training samples to get the filter in the \( f \)th frame \( w_f \) by solving the regression equation. Having gone through the model update step, the filter model \( w_{f,\text{model}} \) is obtained. During the detection stage, in the \((f+1)\)th frame, the tracker first crops the region of interest (ROI), which is centered at the current location. Then the response map is generated by calculating the cyclic correlation between the filter model \( w_{f,\text{model}} \) and the feature of the ROI \( z_{t+1} \) in the frequency domain, and the new location of the object in the \((f+1)\)th frame is determined according to the peak value of the response map. When the new location of the object is determined, the tracker extracts samples from the new location as new training samples. Thus, in the following frames, the training stage, model update, and detection stage are carried out in order.

The main differences in various filters generally lie in the three aforementioned steps, which are introduced in the “Development of DCF-Based Methods” section comprehensively. In particular, the feature-extraction strategy is the universal key component for all correlation filters (CFs), which can be generally divided into two camps: handcrafted and deep features. The commonly used handcrafted features
include gray scale, histograms of the gradient (HOG) [74], a fast version of HOG (fHOG) [75], color names (CNs) [76], and so on, which are not only easy to acquire but also robust to appearance changes. Unlike handcrafted features, deep features (used in deep trackers) are extracted from a multilayer CNN, e.g., the Visual Geometry Group Network [77]. Deep features are usually more discriminative than handcrafted ones and bring too much computational burden to the UAV platform as well.

BASIC FRAMEWORK

Based on the structure described in the previous section, Bolme et al. [78] first proposed training a filter that minimizes the sum of the squared error between preset labels and the correlation between samples and filters. As the MOSSE tracker in [78] showed a high fps rate in tracking performance with a simple structure, plenty of DCF-based trackers with improvements based on [78] have come to the fore, whose tracking strategy can be summarized as follows.

**TRAINING STAGE**

Focusing on ridge regression, one of the goals of the DCF-based tracker is to train a filter in the fth frame $w_f$ that minimizes the squared error $\mathcal{E}$ of the correlation response in training samples $x_i \in \mathbb{R}^{N \times D}$, which is the feature extracted from the image patch $O_{fN}$, compared to their regression targets $y_i \in \mathbb{R}^N$, i.e.,

$$\mathcal{E}(w_f) = \sum_{i=1}^{D} w_f \cdot x_i - y_i \sum_{i=1}^{D} \|w_f\|_2^2,$$  \hspace{1cm} (3)

where $w_f \in \mathbb{R}^N$ and $x_i \in \mathbb{R}^N$, respectively, denote the filter and training samples in the cth feature channel, which

![FIGURE 2. A comparison of the performance of DCF-based and deep trackers under UAVDT, the UAV tracking benchmark [20]. The trackers with an * in the legend are the result of running on a GPU, which utilizes the GPU to accelerate the convolution and pooling calculations. When the tracking speed reaches the red dotted line (30 fps) on a single CPU, it meets the requirements of UAV real-time tracking. The AUC is related to the success rate, and the distance precision (DP) depends on precision, whose exhaustive explanation is carried out in the “Experimental Evaluation and Analysis” section. AUC: area under the curve; CACF: context-aware CF; BACF: background-aware CF; CN: color names; STRCF: spatial-temporal regularized CF; CVPR2020: 2020 Conference on Computer Vision and Pattern Recognition; ICCV2019: 2019 International Conference on Computer Vision; IJCV2018: International Journal of Computer Vision (2018); AAAI2018: 2018 AAAI Conference on Artificial Intelligence; ICCV2017: 2017 IEEE International Conference on Computer Vision Workshop; BMVC2014: 2014 British Machine Vision Conference; ECCV2015: 2015 European Conference on Computer Vision; TPAMI2017: 2017 IEEE Transactions on Pattern Analysis and Machine Intelligence.](image-url)
contain a total of $D$ channels. $\lambda$ is a regularization parameter that controls overfitting.

**Remark 1:** For the convenience of derivation, this work considers the training samples and filter of a certain feature channel as 1D, i.e., $x^c, w^c \in \mathbb{R}^N$, in most cases. In the implemented code, where the samples and filter are 2D matrices with length and width, the derived results can be generalized to 2D.

For convenience, the following introduces only the calculation in the $c$th channel. The minimizer has a closed-form resolution:

$$w^c_f = (X^c_f X^c_f + \lambda I_N)^{-1} X^c_f y,$$

where $I_N$ is the identical matrix and $X^c_f$ denotes the data matrix. Moreover, $X^c_f$ is a circulant matrix, i.e.,

$$X^c_f = C(x^c) = [x^c_1, x^c_2, ..., x^c_N],$$

where all of the columns in $X^c_f$ are actually from the same original training sample, $x^c_1$, using a circulant shift matrix $P \in \mathbb{R}^{N \times N}$:

$$P^i = \begin{bmatrix} 0 & 0 & 0 & \cdots & 1 \\ 1 & 0 & 0 & \cdots & 0 \\ 0 & 1 & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 & 0 \end{bmatrix}.$$

Thus, (5) can also be represented as

$$X^c_f = [P^0 x^c_1, P^1 x^c_2, P^2 x^c_3, ..., P^{N-1} x^c_N].$$

The circulant matrix $X^c_f$ can be diagonalized by DFT matrix $F_N$, which can be expressed as

$$X^c_f = F_N^H \text{diag}(\hat{x}^c_f) F_N,$$

where $^H$ is the Hermitian transpose, i.e., $F_N^H = (F_N)^T$. $\hat{x}^c_f$ denotes the DFT of the generating vector, i.e., $\hat{x}^c_f = \mathcal{F}(x^c_f) = \sqrt{N} F_N x^c_f$. Therefore, (4) can be transformed into a simpler solution:

$$\hat{w}^c_f = \frac{\hat{x}^c_f \odot \hat{y}}{\sum_{k=1}^{N} \hat{x}^c_k \odot \hat{x}^c_k + \lambda I_N}.$$

**Remark 2:** The filter $w^c$ and training samples $x^c$ here are all in the $c$th feature channel, and the calculations among each of the channels do not interfere with each other.

### Model Update

Generally, to avoid overfitting, the filter used for detection does not directly take the calculation result of each frame but obtains it by linear interpolation. When learning rate $\eta$ is introduced, the most common model-updating strategy for the $f$th frame of DCF-based trackers is to update the filter model $w^c_{f-1, \text{model}}$ using linear interpolation as follows:

\[ \text{FIGURE 3. The general tracking structure of the DCF-based methods on board the UAV platform, which can be divided into the training stage, model update, and detection stage. (Source: [19].)} \]
focused on the training samples and proposed (11), over the years, differ frame.
frame represents the feature of the ROI in the frame and thus r(z) denotes the response map.

**Remark 3:** The change in the position of the peak of the response map relative to the center indicates the displacement of the object, which can be calculated to obtain the object’s location in the (f + 1)th frame.

Arriving here, DCF-based trackers use (9) for filter training, (10) for the model update, and (11) for object detection. As most of the calculations in the equations are element wise, DCF-based trackers reduce storage and computation by several orders of magnitude compared with the traditional solution of the ridge regression problem. The DCF-based approach brings the tracking algorithms to a new level, greatly promoting robustness and accuracy with satisfying speed, thus becoming the mainstream method in the UAV tracking field.

Based on the basic framework, a series of methods have been developed to deal with various challenges. For example, Danelljan et al. [79] proposed a creative one that can solve the scale change issue faster. Galoogahi et al. [80] made use of the negative samples generated by the real shift to include a larger search region and real background information and applied the alternating-direction method of multiplier (ADMM) to solve the filter and so forth. Specially, recent years have witnessed a number of CF trackers designed for real-time UAV tracking scenes, e.g., in [5], [22]–[25], and [81], that process not only excellent performance but also wide recognition. This kind of method can reduce the calculation load on UAVs, thus reducing the power consumption to extend the valuable UAV’s endurance time. The remaining computational resources can be put into the high-level control algorithms’ multisensor information fusion, path planning, and so on. Therefore, these advantages have resulted in great advances on the platform of UAVs and ultimately enhance the overall performance of UAVs.

**CFs ONBOARD UAVS**

Generally speaking, there are three reasons that a DCF-based tracker outperforms most of the other tracking methods on UAVs:

1) **Adaptability:** DCF is an online learning method. As mentioned in the previous section, the tracking model is usually updated once every frame, which enables the tracker to respond to changes in the object’s appearance and scale in time. In UAV tracking, due to the frequent changes of view angle, height, and distance, the online updating and training ability of DCF-based trackers ensures their adaptability to object appearance changes, which is of vital competitiveness onboard a UAV.

2) **Robustness:** The DCF-based method belongs to the discriminative methods. It not only learns the object’s information but also the background information. The high discriminability of the filter enables the UAV to maintain high tracking robustness even when it encounters severe environmental changes, similar object interference, and other adverse conditions in the tracking process.

3) **Efficiency:** Most of the operations involved in DCF are element-wise products in the frequency domain, which have an impressive running speed compared with most of the other tracking algorithms. The high speed of the DCF tracker not only makes the UAV realize the real-time tracking function on a single CPU but also saves the power for the UAV. Thus, the redundant computing power can be used to handle other processes and broaden the use scenarios of the UAV.

Based on the aforementioned priorities, recent research has further improved the performance of CFs, raising the application of the CF method in UAVs to an even higher level [5], [22]–[25], [81], [82]. Specifically, Li et al. [22] proposed a novel approach to automatically and adaptively adjust online the spatiotemporal regularization term (AutoTrack), thus greatly reducing the workload of tuning predefined parameters. Huang et al. [24] creatively trained a filter that can learn to repress the aberrances emerging in the detection stage [the aberrance-repressed CF (ARCF) algorithm], resulting in its favorable robustness. Fu et al. [5] applied saliency detection in the filter training process and utilized dual-regularization CFs, thereby accentuating the object’s appearance and achieving a promising result. Lin et al. [23] put forward a novel bidirectional incongruity-aware CF that is not only able to track the object forward but is also able to locate the object in the previous frame, showing its advantage in both adaptability and robustness. Li et al. [82] focused on the training samples and proposed an original time slot-based distillation approach to optimize the training samples’ quality.

As mentioned previously, as an efficient and robust discriminative object tracking strategy, DCF-based methods have outstanding performance and have become the mainstream method in the UAV tracking community. In the next section, the contributions of various DCF-based methods are summarized in detail.

**DEVELOPMENT OF DCF-BASED METHODS**

Even though the general structure of most DCF-based trackers is the same, every tracker possesses its own special priority and contribution. This section focuses on 23 DCF-based trackers’ creative contribution in detail (some are in the same section, i.e., the faster DSST (fDSST) and DSST trackers. In general, as is shown in Figure 4, over the years, different trackers proposed their innovation focusing on different issues and achieved better and better results. Generally, the
trackers introduced in this section are categorized into foundation, scale estimation, feature representation, boundary effect, temporal consistency, and extra types. Table 1 lists most the symbols used in this work for the sake of convenience. Table 2 shows all of the state-of-the-art DCF-based trackers in their categories, including venue, features used, and other characteristics, in corresponding categories.

**Remark 4:** Considering their implementability on board UAVs, only the trackers using handcrafted features are considered practical, which ensures promising speed.

![DCF-Based Tracking Methods Diagram](image)

**FIGURE 4.** The development of DCF-based tracking methods over the years. Here we focus on their most surprising contributions, which are introduced in detail in the “Development of DFC-Based Methods” section. The terms in red denote the trackers’ names, and the terms in black briefly summarize their innovations.
on a single CPU and saves the power supply on board UAVs. Some trackers adopting CF structure but CNN features, e.g., the HCFT tracker \cite{67}, are classified as deep trackers, which are not the mainstream in this work.

**FOUNDATION**

The MOSSE \cite{78}, CSK \cite{86}, and KCF trackers \cite{39}, which are considered the cornerstones of DCF-based methods, built the basic framework and proposed the key idea of DCFs.

**MOSSE TRACKER**

The first tracker that utilized CFs, MOSSE \cite{78}, proposed by Bolme et al., aimed to train a filter that minimizes the squared error between the output of cyclic correlation and the designed labels using the following regression equation:

$$\mathcal{E}(w) = \|w \cdot x - y\|^2.$$ \hfill (12)

Note that the filter $w$ and training samples $x$ are all in a single-feature channel here, i.e., $w = w^\top$, $x = x^\top$.

Using a partial derivative in the frequency domain, a closed-form solution for (12) can be obtained for the MOSSE tracker:

$$\hat{w} = \frac{x^\top \odot Y}{x^\top \odot x^\top}.$$ \hfill (13)

Note that, during the model update stage, the MOSSE tracker uses the following updating scheme for more robust estimation:

\[
\begin{array}{ll}
\text{TABLE 1. A LIST OF FREQUENTLY USED SYMBOLS WITH THEIR MEANING AND DIMENSIONS. FOR EXPRESSING CONCISENESS AND CLARITY, THE DIFFERENT MEANINGS OF SOME SYMBOLS ARE EMPHASIZED IN THE CORRESPONDING SECTIONS.}
\end{array}
\]

| SYMBOL | MEANING | DIMENSION |
|--------|---------|-----------|
| $w$ | CF in the cth channel | $\mathbb{R}^{N \times D}$ |
| $w^\top$ | CF; $w = [w^1, w^2, \ldots, w^N]$ | $\mathbb{R}^N$ |
| $\mathcal{E}$ | Squared error | $\mathbb{R}$ |
| $G$ | Original image patch without feature extraction | $\mathbb{R}^N$ |
| $x^c$ | Training image samples in the cth channel | $\mathbb{R}^{N \times D}$ |
| $x$ | Training samples; $x = [x^1, x^2, \ldots, x^N, \ldots, x^N]$ | $\mathbb{R}^{N \times N \times D}$ |
| $D$ | Total number of feature channels | $\mathbb{R}$ |
| $x_c$ | Elements in training samples $x_c = [x_c^1, x_c^2, \ldots, x_c^N, \ldots, x_c^N]$ | $\mathbb{R}$ |
| $F_N$ | DFT matrix | $\mathbb{C}^{N \times N}$ |
| $R$ | Training samples in the frequency domain | $\mathbb{R}^N$ |
| $C(x^c)$ | Gaussian-shaped regression labels | $\mathbb{R}^N$ |
| $P$ | Circulant data matrix for $x^c$ | $\mathbb{R}^{N \times N}$ |
| $C$ | Circulant data matrix | $\mathbb{R}^{N \times N}$ |
| $L$ | Cropping matrix | $\mathbb{R}^{N \times L}$ |
| $M$ | Identity matrix | $\mathbb{R}^{N \times N}$ |
| $R^-$ | Response map | $\mathbb{R}^N$ |
| $\omega$ | Coordinate vector in the cth channel in dual space | $\mathbb{R}^N$ |
| $\kappa$ | Kernel correlation vector | $\mathbb{R}^N$ |
| $z$ | Samples in the search region (for single scale, $K = 1$) | $\mathbb{R}^{N \times D \times K}$ |
| $z_c$ | Samples corresponding to scale $s_c$ in the search region | $\mathbb{R}^{N \times D}$ |
| $r(z)$ | Response map generated by the search region samples | $\mathbb{R}^N$ |
| $x_s$ | Training samples extracted in the search region | $\mathbb{R}^N$ |
| $\alpha_t$ | Filter parameters learned in the $t$th frame | $\mathbb{R}^N$ |
| $x_{(\text{model})}$ | Training template updated after the $t$th frame | $\mathbb{R}^N$ |
| $\alpha_{(\text{model})}$ | Filter parameters updated after the $t$th frame | $\mathbb{R}^N$ |
| $\eta$ | Total number of frames involved in the regression equation | $\mathbb{R}$ |
| $\eta^h$ | Learning rate | $\mathbb{R}$ |
| $s_i$ | Scaling pool containing $K$ sizes | $\mathbb{R}^K$ |
| $s_i$ | Scale rate in the scaling pool | $\mathbb{R}$ |
| $K$ | Total number of possible scales | $\mathbb{R}$ |
| $s_i$ | Object’s template size | $\mathbb{R}^2$ |
| $w$ | Width of the object's template | $\mathbb{R}$ |
| $h$ | Height of the object's template | $\mathbb{R}$ |
| $\tau$ | Threshold | $\mathbb{R}$ |
| $\omega$ | Spatial punishment weight | $\mathbb{R}^N$ |
| $\gamma$ | Weight for training samples in the $f$th frame | $\mathbb{R}$ |
| $\gamma^*$ | Interpolation weight for different response maps | $\mathbb{R}$ |
| $\gamma^*$ | Cyclic correlation operator | $-$ |
| $\gamma^{-1}$ | DFT | $-$ |
| $\gamma^{-1}$ | Inverse DFT | $-$ |
| $\gamma^{-1}$ | Complex conjugate | $-$ |
| $L^2$ norm known as the **Euclidean norm** | $-$ }

Number of elements in a matrix

Kronecker product

Dot product

A diagonal matrix with elements in the vector

Element-wise multiplication

Hermitian transpose

Mapping function

Shifting operation

Feature transform function

Different tracking models in LCT, Staple, and so forth

134
The object is randomly generated near the current object's center in the search region. On the one hand, this method cannot extract sufficient samples, and, on the other, the candidate boxes overlap with each other, causing inefficiency. The CSK tracker explained for the first time that the cyclic correlation used in CFs is a kind of dense sampling in a mathematical sense and introduced a circulant matrix to calculate the cyclic correlation. Such a strategy first extracts the single-channel feature of an image patch as a basic sample, $x = \{x^i\} \in \mathbb{R}^N$, and cyclically shifts it using (5) to obtain the circulant data matrix (7) as training samples.

In addition to dense sampling and a circulant matrix, the CSK tracker also improves the regression equation in the MOSSE tracker. The improved equation can be expressed as the single-channel form of (3), which is known as regularized least square (RLS). The proposed regularization term can prevent the filter $w$ from overfitting and has gained much better results compared to the MOSSE tracker.

Apart from that, another highlight of the CSK algorithm is its application of a kernel trick, which maps the

\[
A_{f, \text{model}} = \eta A_f + (1 - \eta) A_{f-1, \text{model}}, \quad (14)
\]

\[
B_{f, \text{model}} = \eta B_f + (1 - \eta) B_{f-1, \text{model}}, \quad (15)
\]

where $A = \hat{x} \odot \hat{y}$ and $B = \hat{x} \odot \hat{x}$ indicate the numerator and the denominator, respectively, in (13). Therefore, the filter model obtained after the $f$th frame is $w_{f, \text{model}} = A_{f, \text{model}} / B_{f, \text{model}}$.

Having calculated the filter model $w_{\text{model}}$, the MOSSE tracker adopts (11) for detection in new frames. Due to its simplicity, the MOSSE algorithm has achieved a compelling tracking speed of hundreds of fps.

Large-scale variation remains difficult for the MOSSE tracker to adapt to, and its single-channel, gray-scale features are not powerful enough. In the MOSSE tracker, the object does not consider the linear separability of the samples in high-dimensional spaces.

**CSK TRACKER**

The previous outstanding trackers, such as the SVM tracker [37], used sparse sampling to obtain the training samples. To be specific, several candidate boxes with the same size as the object are randomly generated near the current object’s center in the search region. On the one hand, this method cannot extract sufficient samples, and, on the other, the candidate boxes overlap with each other, causing inefficiency. The CSK tracker [86] explained for the first time that the cyclic correlation used in CFs is a kind of dense sampling in a mathematical sense and introduced a circulant matrix to calculate the cyclic correlation. Such a strategy first extracts the single-channel feature of an image patch as a basic sample, $x = \{x^i\} \in \mathbb{R}^N$, and cyclically shifts it using (5) to obtain the circulant data matrix (7) as training samples.

In addition to dense sampling and a circulant matrix, the CSK tracker also improves the regression equation in the MOSSE tracker. The improved equation can be expressed as the single-channel form of (3), which is known as regularized least square (RLS). The proposed regularization term can prevent the filter $w$ from overfitting and has gained much better results compared to the MOSSE tracker.

Apart from that, another highlight of the CSK algorithm is its application of a kernel trick, which maps the

\[
A_{f, \text{model}} = \eta A_f + (1 - \eta) A_{f-1, \text{model}}, \quad (14)
\]

\[
B_{f, \text{model}} = \eta B_f + (1 - \eta) B_{f-1, \text{model}}, \quad (15)
\]

where $A = \hat{x} \odot \hat{y}$ and $B = \hat{x} \odot \hat{x}$ indicate the numerator and the denominator, respectively, in (13). Therefore, the filter model obtained after the $f$th frame is $w_{f, \text{model}} = A_{f, \text{model}} / B_{f, \text{model}}$.

Having calculated the filter model $w_{\text{model}}$, the MOSSE tracker adopts (11) for detection in new frames. Due to its simplicity, the MOSSE algorithm has achieved a compelling tracking speed of hundreds of fps.
classification procedure to a high-dimensional feature space to acquire even better performance. Using a kernel trick, $w$ can be written as $w = \sum a_i \phi(x_i)$, where the mapping function $\phi(\cdot)$ can map the data to the high-dimensional feature space, and sample $x_i$ is from the original sample set $x$ using circulant shift matrix $P$, e.g., $x_i = P^k x$. The RLS with kernels has a simple closed-form solution:

$$\alpha = (K + \lambda I_n)^{-1} y,$$  

(16)

where $K$ is the kernel matrix with elements $K_{ij} = k(x_i, x_j)$. Kernel $k$ is defined as

$$k(x_i, x_j) = \langle \phi(x_i), \phi(x_j) \rangle,$$  

(17)

where $\langle \cdot, \cdot \rangle$ denotes a dot product.

**Remark 5:** Although Henriques et al. achieved amazing tracking results using the CSK tracker, the algorithm still uses illumination intensity features, which are inferior robust.

**KCF TRACKER**

Standing as the basic framework of most of the subsequent DCF-based trackers, the main contribution of the KCF tracker is that Henriques et al. [39] formulated their prior work into the CSK tracker into a DCF-based tracking structure and greatly boosted the tracker’s performance.

First, the KCF algorithm proposed a nonlinear regression equation for filter training:

$$E(w) = \frac{1}{2} \sum_{i=1}^{D} \| w^* \phi(x_i^*) - y_i \|^2 + \lambda \sum_{i=1}^{D} \| w_i \|^2.$$  

(18)

Assuming that the training sample is a single-feature channel, i.e., $x \in \mathbb{R}^{N \times D}$, by adopting the kernel tricks $w = \sum a_i \phi(x_i)$, their solution to the filter parameter $\alpha$ was first obtained in the same way as the CSK tracker, i.e., (16). Proving that the kernel matrix $K$ is circulant, a faster version of (16) is proposed by

$$\hat{\alpha} = \frac{\hat{y}}{k^{xx} + \lambda I_N},$$  

(19)

where $k^{xx}$ denotes the kernel correlation vector. For two arbitrary vectors, $x$ and $x'$, their kernel correlation is the vector $k^{xx}$ with elements

$$k^{xx} = \phi^T(x') \phi(P^{-1} x).$$  

(20)

The KCF tracker gave two commonly used kernel functions:

1) a Gaussian kernel:

$$k^{xx} = \exp\left(-\frac{1}{\sigma^2}(\|x\|^2 + \|x\|^2 - 2F^{-1}(x \odot x'))\right).$$  

(21)

2) a polynomial kernel:

$$k^{xx} = \left(\mathcal{F}^{-1}(\hat{x} \odot \hat{x'}) + a\right)^\eta.$$

(22)

If the training sample $x$ is a multichannel sample, $x \in \mathbb{R}^{N \times D}$, e.g., fHOG [75], the multichannel version of (21) and (22) is as follows:

3) a Gaussian kernel:

$$k^{xx} = \exp\left(-\frac{1}{\sigma^2}(\|x\|^2 + \|x\|^2 - 2F^{-1}(\sum \hat{x} \odot \hat{x'})\right).$$  

(23)

4) a polynomial kernel:

$$k^{xx} = \left(\mathcal{F}^{-1}(\sum \hat{x} \odot \hat{x'}) + a\right)^\eta.$$  

(24)

In this case, it is a linear filter but trained in the dual space $\alpha$.

For the model update, instead of updating filter $w$, the KCF method updates the filter parameter and the sample:

$$\tilde{\alpha}_{\text{model}} = (1 - \eta) \tilde{\alpha}_{\text{old}, \text{model}} + \eta \tilde{\alpha}_{\text{train}}.$$  

(25)

Thus, the sample model $x_{t, \text{model}}$ instead of the original sample $x_t$ is used for filter training in the $t$th frame.

In the detection stage, the response map can be obtained using

$$\mathbf{r}(z) = \mathcal{F}^{-1}(\sum \hat{x} \odot \tilde{\alpha}_{\text{model}}),$$  

(27)

where $z$ denotes the samples in the search region.

The KCF tracker formulated the kernel tricks used in the CSK tracker into a DCF-based tracking structure and put forward a multichannel sample’s training strategy, thus becoming the basic framework for most of the DCF-based trackers.

**SCALE ESTIMATION**

The three aforementioned fundamental trackers [39], [78], [86] can achieve target localization, but they adopted single scale during the tracking process, i.e., assuming that the scale of the object is fixed. Considering that the target scale changes in real-world tracking, how best to accurately and effectively estimate the scale of the object has become an urgent problem to be solved.

**SAMF TRACKER**

The main contribution of the SAMF tracker is the proposal of an effective scale-estimation method. In the SAMF algorithm, we assume that the original fixed size of the object template is $s_t = (w, h)$, where $w$ and $h$ represent the width and height of the search region, respectively. The scale pool can be defined as $S = \{s_1, s_2, \ldots, s_k\}$. During the detection stage, the image patches in $k$ different
sizes in the scale pool are first cropped. Having extracted the feature of the resized image patches \( \{ z'_{i}, z'^{2}_{i}, \ldots, z'^{n}_{i} \} \), the SAMF tracker can determine the optimal scale \( s \) by solving the following optimization problem:

\[
\arg\max_{s} r(z^s). \tag{28}
\]

Moreover, the location of the peak value can be used to estimate the new location of the object. Thus, the SAMF tracker achieves both location and scale prediction at the same time.

**Remark 6**: Taking into account that the new samples extracted in each frame have different scales, the SAMF tracker first converts the sample bilinear interpolation to the same size and then uses the same linear-interpolation template update strategy as the KCF tracker, that is, (12) and (26).

**DSST AND fDSST TRACKERS**

The scaling pool algorithm given in the SAMF tracker [83] coped with the scale variations, but its operation speed and robustness left room for further improvement. Danelljan et al. made an innovative contribution to the scale-estimation algorithm.

In [84], the training regression equation adopted the ridge regression in the KCF tracker [39], i.e., (18), and applied a linear kernel, i.e., (25). Their solution is

\[
\hat{w} = \frac{\hat{x} \cdot \hat{y}}{\sum_{i=1}^{N} \hat{x}_{i} \cdot \hat{y}_{i} + \lambda I_{N}}. \tag{29}
\]

Then, the core idea of the DSST tracker in [84] is to train two filters, namely, a translation \( w_{\text{trans}} \) and a scale filter \( w_{\text{scale}} \). When a new frame arrives, the translation filter is first used to search for the new object’s location; then, the samples of the different scales \( \{ z_{i}, z'_{i}, \ldots, z'^{n}_{i} \} \) are extracted near the new center. The samples in various scales are used for the scale filter to predict the proper scale. Then, the translation filter uses the samples centered at the predicted location for training, while the scale filter utilizes the samples in different scales that were centered at the predicted scale for training. Such a scale-evaluation method combines both robustness and speed.

**Remark 7**: In the implemented code, the translation filter adopts the promoted 2D results of (29), where the training samples and filter in a certain feature channel are 2D. Differently, the DSST tracker pulls the 2D matrices into a 1D vector to train the 1D scale filter, adopting (29) directly. Such a strategy greatly boosts the processing speed of the DSST tracker. Another reason that the DSST tracker achieves fast scale estimation is that the scale filter crops the patches about the size of the object, which is relatively small, making feature extraction more efficient.

Based on the DSST tracker, Danelljan et al. further proposed a faster version, called the fDSST tracker [79], which improves tracker performance while obtaining an even higher tracking speed. The strategy can be summarized into three points. First, the fDSST tracker uses sub-grid interpolation to reduce the size of training samples and search samples. Second, the fDSST tracker performs principal component analysis (PCA) on the features of the sample (similar to the PCA of the CN in the CN tracker [87]) to achieve feature dimensionality reduction. Finally, the scale filter is compressed by reducing the total number of features to the number of features after PCA dimensionality reduction, which greatly reduces redundant information. Under these three acceleration strategies, an fDSST tracker can expand its search region to obtain a better tracking performance.

**FEATURE REPRESENTATION**

For the sake of better robustness and favorable discriminative ability of the trackers, it is of vital importance to learn sufficient and effective object features. The trackers that introduced this part [87]–[89] put forward a variety of new methods to learn and effectively utilize the expressive features, thus promoting DCF algorithms to a new level.

**CN TRACKER**

The CN tracker [87], which was proposed by Danelljan et al., put forward an innovative color feature based on the CSK tracker [86], which is another powerful handcrafted feature. Further, the CN tracker also settled the multichannel training problem.

The color attribute [76], also known as the CN, directly denotes the 11 color language labels defined by human beings. The CN feature first maps the value of each channel in a red-green-blue image to the 11-dimensional CNs’ probability, which, when summed up, equals one. Then, the 11-dimensional color space is mapped to the 10-dimensional orthogonal basis subspace; thus, dimension reduction and normalization are achieved at the same time (the CN tracker).

However, there is a linear relationship between the computational complexity of the CSK tracker and the number of dimensions of the feature used. To lessen the amount of calculation cost and to guarantee high speed, the CN tracker further proposed low-dimensional adaptive color attributes, where PCA is performed on 10-dimensional features to select the two dimensions with the most information (the analysis results of each frame are adaptive), that is, the CN_2 tracker.

Except for proposing the color feature, another improvement in the CN tracker [87] is that it considers the information in the previous frames in their training progress, i.e.,

\[
E(w_f) = \sum_{j=1}^{n} \theta_{j} \left( \frac{1}{2} \left\| \sum_{i=1}^{m} w_{j} \cdot \phi(x_{i}) - y_{j} \right\|^{2} + \lambda \sum_{i=1}^{m} \| w_{j} \|^{2} \right). \tag{30}
\]

where \( y_{j} \) denotes the values in the \( j \)th frame, e.g., \( \theta_{j} \) indicates the weight for the \( j \)th frame. We assume that the training sample is a single-feature channel, i.e., \( x \in \mathbb{R}^{3 \times 1} \). The solution for the filter parameter \( \alpha_{j} \) in the \( 7 \)th frame is
\[
\hat{a}_f = \frac{\sum_{j=1}^{L} \theta_j \hat{y} \otimes \hat{k}^a_j}{\sum_{j=1}^{L} \theta_j \hat{k}^a_j \otimes (\hat{k}^a_j + \lambda_1)}.
\]

Such a strategy, where previous frames are taken into account, can improve the robustness of the tracker.

**Remark 8:** The CN feature not only improved the performance of the original CSK algorithm but also brought DCF-based trackers to a new level, making them more robust under challenging scenes.

**STAPLE TRACKER**

The samples used to train the filter can be various features, such as fHOG [75], CN [76], color histogram, and so forth. In the Staple tracker [88], Bertinetto et al. found that different features have diverse advantages under various tracking scenarios. For example, the fHOG feature has a good expression effect under IV while the color histogram is more powerful under deformation and rotation. Based on this discovery, how best to make the two complementary features fuse in tracking becomes the focus of the Staple algorithm.

In [88], Bertinetto et al. proposed a complementary fusion tracking method, which simultaneously uses the main formula of the DSST tracker, i.e., (18) and (25), to train template filters \( \mathbf{w} \) based on the fHOG feature, i.e., (29), and another ridge regression equation to train histogram weights: \( \mathbf{h} \)

\[
\mathcal{E}_\text{hist}(\mathbf{h}) = \frac{1}{T_1} \sum_{O_{i},O} (\mathbf{h}^T \mathbf{\psi}[O_i] - 1)^2 + \frac{1}{T_2} \sum_{O_{i},\mathcal{B}} (\mathbf{h}^T \mathbf{\psi}[O_i] - 1)^2,
\]

where \( O \) and \( \mathcal{B} \) represent the object and background region, respectively. \( | \mathcal{B} | \) denotes the total samples \( O \) in the background region in the image patch \( O \). \( \mathbf{\psi}[O] \) indicates an M-channel feature transform function, which maps image patch \( O \) to an M-dimensional color feature space. Then the elements \( h_j \) in histogram weight \( \mathbf{h} \) can be obtained:

\[
h_j = \frac{\rho(O)}{\rho(O) + \rho(B) + \lambda},
\]

where \( \rho(O) \) denotes the frequency of color bin \( j \) in the object region, and \( \rho(B) \) is similar. Thus, in the search stage, the histogram score \( \mathbf{r}_\text{hist} \) can be calculated by

\[
r(O)_\text{hist} = \frac{1}{|S|} \mathbf{h}^T \mathbf{\psi}[O] \otimes \mathbf{y},
\]

where \( S \) is the search region, \( 1/|S| \) denotes the likelihood map, \( \mathbf{O} \) represents the original search region image patch, and \( \mathbf{y} \) indicates the Gaussian-shaped labels.

**Remark 9:** The response maps obtained by the two methods are linearly superimposed with the following formula for the final object detection:

\[
r(z) = \gamma_\text{hist} r(O)_\text{hist} + \gamma_\text{template} r(z)_\text{template}.
\]

**MCCT-H TRACKER**

The features used by the DCF-based trackers that achieve real time on UAVs can be summarized as fHOG [75], CN [76], and so on. Now the main processing methods for multichannel features are 1) a direct overlay by layer, such as the KCF [39], SAMF [83], and DSST trackers [84] and so forth, and 2) using PCA to reduce dimension, like the ECO [95], CN [87], and fDSST trackers [79]. Nevertheless, these methods have obvious limitations. The stability and reliability of each feature channel under different frames are usually unequal, and the number of reliable features is not necessarily constant. Thus, both methods—directly overlaying by layer (also parallel tracking) and using PCA to reduce dimensionality—may cause information redundancy, missing information, and even unreliability.

Based on the aforementioned considerations, Wang et al. proposed a multicon joint tracking scheme (MCCT) [89], introducing feature and expert pools (different combinations of features in the feature pool) to select the most reliable expert as the features for tracking in different frames, which, compared with previous fusion methods, has achieved better results.

In the MCCT tracker, the basic training structure of the DSST tracker [84], i.e., (18), (25), and (29), is adopted. Innovatively, it introduces feature and expert pools. The feature pool consists of three types of features, \{Low, Middle, High\}, which correspond to \{fHOG, fHOG, and CN\}, respectively (MCCT-H). Optionaly combining the features into \( C_1 \) gives an expert \( E_1, E_2, \ldots, E_7 \). To rate the experts, the authors have a pair-evaluation score,

\[
R_{\text{pair}}(E_i) = \frac{M_{E_i} + V_{E_i}}{V_{E_i}},
\]

where \( M_{E_i} \) and \( V_{E_i} \) are both calculated based on the overlap ratio of the bounding box generated by expert \( E_i \) and others. Differently, \( M_{E_i} \) directly denotes the consistency of expert \( E_i \) and others, while \( V_{E_i} \) is the temporal stability of the overlap ratio. \( \xi \) is used to avoid zero denominators. The bigger the score \( R_{\text{pair}}(E_i) \) is, the more pair judging reliable expert \( E_i \) is. The authors also introduced the self-evaluation score,

\[
R_{\text{self}}(E_i) = \frac{1}{N} \sum_{T} W_T S_T E_i,
\]

where \( S_T E_i \) is calculated according to the Euclidean distance of the bounding-box temporal change, and \( 1/N \Sigma_T W_T \) denotes the average in frames \( W \). The self-evaluation score \( R_{\text{self}}(E_i) \) actually measures the degree of smoothness. The final robustness score \( R(E_i) \) is defined as

\[
R(E_i) = \mu \cdot R_{\text{pair}}(E_i) + (1 - \mu) \cdot R_{\text{self}}(E_i).
\]

Thus, the expert with the highest robustness score is selected for the current tracking.

Another innovative highlight in the MCCT-H tracker is that its update scheme uses an adaptive learning rate.
By computing the average PSR $P = (R_{\text{max}} - m)/\sigma$ of different features,

$$P_{\text{mean}} = \frac{1}{3}(P_{t}^{f} + P_{s}^{f} + P_{l}^{f})$$  \hspace{1cm} (39)

is used to evaluate the tracking result, and the learning rate $\eta$ is changed according to the PSR, which is no longer a constant.

**BOUNDARY EFFECT**

The circulant samples used by DCF-based trackers have a huge disadvantage, that is, the boundary effect. The aforementioned filters typically use a cosine window to weaken the boundary effect, which can reduce the boundary effect only to a certain extent. Such a method cannot solve the problem of an inaccurate response value to the circulant samples that centered to a large extent at the boundary. Under such considerations, trackers such as the KCF [39] and DSST trackers [84] can sample only two to three times the original object’s size. To settle this issue, trackers [71], [80], [90], [91] that enabled sufficient negative samples learned while alleviating the boundary effect were proposed, thus ensuring better tracking robustness.

**SRDCF AND SRDCFDECON TRACKERS**

The most favorable benefit of the SRDCF tracker [71] is the introduction of a spatial punishment weight in the filter training process. Its improved main training formula for filters in the $T$th frame $w_{T}$ is

$$E(w_{T}) = \sum_{j=1}^{T} \sum_{c=1}^{D} \theta_{j} \left[ \sum_{i=1}^{D} w_{i}^{f} \cdot x_{j}^{f} - y_{j}^{f} \right]^{2} + \sum_{c=1}^{D} \| \omega \odot w_{c}^{f} \|_{l}^{2}. \hspace{1cm} (40)$$

where $\omega$ is the given regularization weight, which has a larger value far from the center point and a smaller value near the center point. It can therefore increase the weight at the center of the filter, making the more reliable center part of the samples possess a higher impact on the response map and shield the noise or unreliable negative samples far from the center. $\theta_{j}$ denotes the weight of the training samples in the $f$th frame, and $w^{c} \in \mathbb{R}^{k}$ indicates the $c$th feature channel of the filter $w$.

The SRDCF tracker is also unique for solving new regression equations. First, the Parseval formula is used, which turned (40) into

$$E(w_{T}) = \sum_{j=1}^{T} \sum_{c=1}^{D} \theta_{j} \left[ \sum_{i=1}^{D} w_{i}^{f} D(x_{j}^{f}) - y_{j}^{f} \right]^{2} + \sum_{c=1}^{D} \| C(\omega) w_{c}^{f} \|_{l}^{2}. \hspace{1cm} (41)$$

Here $D(x^{f})$ denotes a diagonal matrix with elements in $x^{f}$, and $C(\omega)$ is an $N \times N$ matrix. Then they use a Gauss–Seidel iteration to solve the simplified calculation.

The SRDCF tracker represents a milestone in the development of CF trackers. It supplements the KCF’s ridge regression main formula with spatial penalty terms and uses the Gauss–Seidel iteration method to solve it. The experiments have confirmed that the SRDCF algorithm enables the filter to learn more negative samples, which improves the robustness of tracking; thus, the SRDCF algorithm has become a popular baseline for subsequent CF trackers.

**Remark 10:** Moreover, the improvement of the SRDCF tracker’s performance is also due to the addition of all the previous samples, which slowed its calculation speed as well.

Danelljan et al. further proposed the SRDCFdecon tracker [90] on the basis of the SRDCF tracker. The SRDCFdecon tracker adopted a main formula similar to the SRDCF’s:

$$E(w_{T}, \theta) = \sum_{j=1}^{T} \sum_{c=1}^{D} \theta_{j} \left[ \sum_{i=1}^{D} w_{i}^{f} \cdot x_{j}^{f} - y_{j}^{f} \right]^{2} + \frac{1}{\mu} \sum_{j=1}^{T} \sum_{c=1}^{D} \theta_{j} \left[ \sum_{i=1}^{D} \| \omega \odot w_{c}^{f} \|_{l}^{2} \right. \left. + \sum_{i=1}^{D} \| \omega \odot w_{i}^{f} \|_{l}^{2} \right], \hspace{1cm} (42)$$

where $\mu > 0$ represents the flexibility parameter and $\rho_{j} > 0$, satisfying $\sum_{j} \rho_{j} = 1$, denotes the previous sample weights.

The difference is that the main formula of the SRDCFdecon algorithm is a binary regression, namely, filter $w$ and weight $\theta$. Therefore, for SRDCFdecon, the weight $\theta$ of the sample used for filter training in the previous $T$ frames is no longer fixed but is solved dynamically according to the regression equation. As a result, the online adjustment of the weight can continuously influence previous $T$ frames, which ensures that the impact of the frames with high confidence is greater and that the impact of the contaminated samples is reduced.

**CSR-DCF TRACKER**

Note that the spatial punishment $\omega$ in (40) is fixed, while the importance of each pixel does not necessarily subject itself to a fixed uniform that declines from the middle to the surrounding area. In addition, the reliability of each feature layer of the object sample is also different; in other words, the previous tracking methods learned some fake information that may disturb the tracking.

To ensure that the filter learns more accurate object information with irregular shapes and weakens the unreliable feature channels, Lukezic et al. made worthwhile improvements to the SRDCF tracker [90] in the CSR-DCF tracker [91] to change the fixed spatial punishment item $\omega$ to a pixel-by-pixel confidence map $m$ and introduced the channel confidence score to modify the generated response map. In the CSR-DCF tracker, Lukezic et al. introduced a spatial confidence map $m$, where the value of each element $m_{i} \in m$ represents the probability that each pixel is the object. The calculation method is divided into three steps: the prior layer, the Bayesian probability under the color model, and the Epanechnikov kernel:

$$p(m_{i} = 1|x, x_{i}) \propto p(m_{i} = 1)p(x|m_{i} = 1, x)p(x|m_{i} = 1), \hspace{1cm} (43)$$

where the prior $p(m_{i} = 1)$ is defined by the ratio among the region sizes for the color histogram; the Bayesian probability under the color model $p(x|m_{i} = 1, x_{i})$ adopts a method similar to the Staple tracker [88], which is actually a...
likelihood map; and $p(x_i|m_i = 1)$ is calculated using a modified Epanechnikov kernel.

Having obtained the spatial confidence map, the authors proposed their main training formula:

$$E(w) = \sum_{i} \| m \odot w_i \cdot x_i - y_i \|^2 + \frac{\lambda}{2} \| m \odot w \|^2. \quad (44)$$

To perform an augmented Lagrange iteration, (44) can be first transformed into

$$E(w) = \| w^H \Omega (x_i) - y_i \|^2 + \frac{\lambda}{2} \| w \|^2 + \| w^H \Omega (x_i) - y_i \|^2 + \lambda \| w \|^2, \quad (45)$$

where $w = m \odot w$, and $w$ is a dual variable satisfying $w \odot m = 0$. Note that $\Omega$ is a complex Lagrange multiplier, and $\lambda > 0$. Using an augmented Lagrange iteration, (45) can be solved.

Moreover, the CSR-DCF algorithm also proposed a per-channel detection-reliability score:

$$S = 1 - \min \left( \frac{r(z)_{mean}}{r(z)_{max}} \right), \quad (46)$$

where $r(z)_{mean}$ and $r(z)_{max}$ denote the second and first major mode in the response map, respectively. In this way, it can surpass the response map containing noise or a distractor.

BACKGROUND-AWARE CF TRACKER

The background-aware CF (BACF) tracker [92] proposed by Kiani Galoolghi et al. also enables the filter to learn abundant background information while reducing boundary effect.

**Remark 11:** The BACF tracker utilizes a cropping matrix to automatically crop the samples in the large ROI into multiple small samples of the same size as the object. To be more specific, these small samples are generated by a circulant shift plus a cropping operator, which are all sub-regions of the ROI. This makes it possible to introduce more background samples without introducing too much boundary effect.

In [92], the authors first introduced the cropping matrix $C \in \mathbb{R}^{N \times L}$, which can select and extract the pixels of fixed size $\mathbb{R}^L$ in the center area of the sample $x_i \in \mathbb{R}^N$, and the circulant shift operator $P$ is the same in the KCF tracker [39]. Thus, $C x_i P^i$ can be regarded as the central area of the circulant sample $x_i$. In different circulant samples, such a center area corresponds to positive or negative samples, which is consistent with the size of the corresponding position value of the expected Gaussian label $y(j)$. Note that since $L \geq N$, the samples used in the BACF tracker are far larger than with other trackers, ensuring abundant negative samples learning. Based on this idea, the BACF tracker put forward its main training formula as

$$E(w) = \frac{1}{2} \sum_{j=1}^{N} \| w^c (C x_i P^j - y(j)) \|^2 + \frac{\lambda}{2} \| w \|^2, \quad (47)$$

where the regression label $y$ is in $\mathbb{R}^L$ instead of $\mathbb{R}^N$. In this way, the search region is enlarged, the real negative samples increase, and the circularity of samples is ensured at the same time. To apply the ADMM algorithm, (47) can first be divided into

$$E(w, g) = \frac{1}{2} \| \bar{X} g - y \|^2 + \frac{\lambda}{2} \| w \|^2, \quad (48)$$

s.t. $g = \sqrt{F}(I_r P^T \odot I_d)w,$

where $\odot$ denotes the Kronecker product.

Due to it possessing both amazing tracking speed and impressive tracking performance, the BACF tracker’s core architecture has become another milestone in the history of CF tracking. Since then, many DCF-based trackers have chosen a BACF tracker as their baseline and have made even more innovative contributions.

TEMPORAL CONSISTENCY

The appearance of the object during tracking sometimes undergoes a short-term large variation, such as partial OOC, similar object interference, and so on. These tracking scenarios usually cause filter degradation, which leads to subsequent tracking failures. To ensure temporal consistency of the filter [66] of the response map, [22] and [24] are effective methods to counter the unexpected abrupt degradation.

SPATIOTEMPORAL REGULARIZED CF TRACKER

Li et al. put forward the spatiotemporal regularized CF (STRFCF) [66], which introduced the temporal regularization term based on the previous main training formula of the SRDCF tracker [90], i.e., (40), to suppress excessive short-term object-appearance changes and to obtain a more stable tracking effect on the basis of the SRDCF tracker:

$$E(w_i) = \frac{1}{2} \| \sum_{j=1}^{N} w_j \cdot x_{ij} - y_i \|^2 + \frac{\lambda}{2} \| \sum_{j=1}^{N} w_j \odot y_i \|^2,$$

$$+ \frac{\mu}{2} \| \sum_{j=1}^{N} w_{ji} - w_{ji-1} \|^2. \quad (49)$$

where $\| w_{ji} - w_{ji-1} \|^2$ represents the temporal regularization term, according to passive-aggressive (PA) algorithm. Also, the STRCF tracker uses an ADMM iteration for the efficient solution of (49).

**Remark 12:** Due to the introduction of the temporal regular term, the main training formula of the STRCF no longer needs the information from all the previous $T$ frames, which greatly reduces the memory required for training and significantly improves the calculation speed compared to the SRDCF tracker.

The experiments have proved that the filter in the STRCF algorithm has strong temporal stability. It is especially robust under scenes where the object undergoes rapid appearance variation, such as FM and IV. The time regularization term under the PA algorithm is also very useful for further reference.
ARCF-H AND ARCF-HC TRACKERS

Based on the aforementioned abundant DCF-based tracking research, the DCF-based tracker used specifically for UAV tracking has finally landed on the stage [22], [24]. A huge shortcoming of the existing DCF-based trackers in the UAV tracking environment is their poor anti-interference ability. First, the UAV tracking scene is more complex and usually encounters large environmental disturbances, such as IV, similar objects, and so forth, which affects the tracking results. Second, the previous CF trackers usually expand the search region and impose a spatial penalty term to solve the boundary effect, which would inevitably cause more background information and is thus more likely to introduce environment noise.

To solve this problem and to improve the performance of the CF tracker in the tracking environment with serious background interference on board a UAV, Huang et al. proposed the ARCF algorithm [24]. The core idea of the ARCF tracker is to use the aberrance of the successive response maps to suppress and learn the environmental noise.

**Remark 13**: Unlike the previous methods, which utilize the reliability of the current response map to suppress the corresponding position in the subsequent frames, e.g., the CSR-DCF tracker [91], the ARCF tracker introduced the response map’s aberrance into the training formula for the first time, suppressing any possible noise in the training stage, and has achieved quite considerable results.

In [24], the authors first defined the noise evaluation standard:

\[
||M_1[\psi_{p,q}] - M_2||_F^2, \tag{50}
\]

where \(M_1\) and \(M_2\) represent two response maps, and \(\psi_{p,q}\) indicates the shifting operation that makes the locations of the peaks in \(M_1\) and \(M_2\) the same. Thus, when aberrance occurs, the Euclidean distance of \(M_1[\psi_{p,q}]\) and \(M_2\) rises, resulting in the increase of (50).

Based on this standard, the training equation of the ARCF algorithm can be expressed as

\[
E(w_f) = \frac{1}{2}\sum_{i=1}^{\alpha} w_i^2 \cdot C_{w_i} - y_i^2 + \frac{1}{2}\sum_{i=1}^{\alpha} w_i^2
\]

\[+ \frac{\tilde{\gamma}}{2}\sum_{i=1}^{\alpha} (w_{i-1} \cdot C_{w_{i-1}})[\psi_{p,q}] - \frac{\alpha}{2}\sum_{i=1}^{\alpha} w_i^2 \cdot C_{w_i}]^2, \tag{51}
\]

where matrix \(C\) in the first item is the cropping matrix in the BACF tracker [92], the second item is a regular term (the same as the KCF tracker) [39], and the third item is meant to suppress the possible response map aberrance in the \(f\)th frame relative to the \(f-1\)th frame. Finally, the ARCF tracker adopts the ADMM iterative algorithm to solve (51).

The experiments have confirmed that the ARCF tracker’s suppression effect on response map aberrance is very significant on most UAV tracking benchmarks. For scenes with frequent noises, such as fast movement and OCC, the ARCF tracker stands out among the state of the art, particularly under small object (SO) tracking scenes. Due to low object resolution and insufficient information, aberrance is more likely to appear due to background environment interference. The ARCF tracker has demonstrated outstanding performance under such scenes and has become one of the most preferred algorithms for UAV-based aerial tracking.

**AUTOTRACK TRACKER**

Another well-known CF tracker specifically for UAV tracking uses the STRCF tracker [66] as its baseline. Although the stability and robustness of the STRCF tracker are excellent, its spatiotemporal regularization term introduces too many parameters that need to be manually set, such as spatial punishment \(\omega\) and temporal parameters \(\mu\). On the one hand, these parameters usually take considerable time to adjust to find the best one during the experiment. On the other hand, the determined parameters cannot always perform well in every sequence.

Based on these considerations, Li et al. proposed a DCF-based tracker with automatic spatial-temporal regularization (AutoTrack) [22], which uses local and global response maps, respectively, to automatically adjust the spatial and temporal weights dynamically, which is both highly adaptable to different sequences and efficient at the same time.

In [22], the authors first defined a response map reliability vector \(\Pi\), whose element \(\Pi_i\) can be expressed as

\[
\Pi_i = \frac{\{M_1[\psi_{p,q}] - M_{f-1}\}}{M_{f-1}}, \tag{52}
\]

where \(M_{f-1}\) and \(M_f\) denote the response map in the \(f-1\)th frame and the \(f\)th frame, respectively. \(\psi_{p,q}\) is the same shifting operator that makes two peaks in two response maps positionally coincide in the ARCF tracker [24]. Then, the automatic spatial weight \(\omega\) can be defined as

\[
\omega = C_N^\delta \log(\Pi + 1) + \omega, \tag{53}
\]

where \(C_N^\delta\) is used to crop the center region \(R_N\), which denotes the object’s template size; \(\delta\) is a constant aimed to adjust the weight; and \(\omega\) is the same fixed spatial punishment term used to settle the boundary effect as in the SRDCF tracker. Thus, an automatic spatial regularization term according to local response map can be calculated, where the higher the value is, the less reliable the pixel is, resulting in a smaller weight in filter \(\omega\).

Meanwhile, the authors defined automatic temporal parameter as

\[
\mu = \frac{\xi}{\log(\nu ||\Pi||_\infty + 1)}, ||\Pi||_\infty \leq \tau, \tag{54}
\]

where \(\xi\) and \(\nu\) are all hyperparameters. Only when \(||\Pi||_\infty\) is smaller than the threshold \(\tau\) (denoting that the samples are reliable) does the CF learn. When \(||\Pi||_\infty \leq \tau\), the higher
\( |I| \) is, the less reliable the samples are, resulting in a smaller \( \bar{\mu} \), making the filter \( \mathbf{w} \) change by very little.

Based on this, the AutoTrack tracker gave its overall objective as

\[
\mathcal{E}(\mathbf{w}_f, \mu_f) = \frac{1}{2} \sum_{i=1}^{D} \left( \mathbf{w}_f^\top \mathbf{x}_i^f - y_i \right)^2 + \frac{1}{2} \sum_{i=1}^{D} \| \tilde{\omega} \odot \mathbf{w}_f^i \|_2^2 + \frac{1}{2} \| \mu_f - \bar{\mu} \|_2^2,
\]

where \( \tilde{\omega} \) denotes the automatic spatial weights obtained by (53). \( \mu_f \) and \( \bar{\mu} \) indicate the optimized temporal and reference parameters, respectively.

Abundant experiments have demonstrated that the AutoTrack tracker is extremely robust to tracking scenes where an object undergoes appearance changes, such as IV, VC, and so on.

**EXTRA TYPES**

Some DCF-based trackers provide extra solutions for long-term tracking [93], computational acceleration [95], background suppression [96], and calculation tricks [85], which make outstanding contributions to DCF-based tracking communities.

**LCT TRACKER**

Compared to short-term tracking, long-term tracking is usually more challenging. Due to more extreme challenges like full OCC and out of view, the DCF-based trackers in long-term tracking scenes are more likely to encounter tracking failures. To address this problem, Ma et al. gave an innovative tracking idea in the long-term correlation (LCT) tracker [93]. The idea in the LCT tracker is similar to a famous previous work: the tracking, learning, and detection (TLD) tracker [97]. In the TLD tracker, long-term tracking is divided into TLD phases. The tracker learns frame by frame and determines the position of the object in each frame, while the detector learns the large appearance change of the object and searches for the object globally when the tracking fails.

The first innovation is that two training models are introduced in the LCT tracker, i.e., the temporal regression model \( R_t \) and the object-appearance regression \( R_o \). Between them, \( R_t \) used both the object and the surrounding context to train the model, which is the same as with the KCF tracker [39].

As a remedy, the main purpose of \( R_t \) is to predict the current state of the object and make an appropriate scale estimation, which is similar to the scale filter in the DSST tracker [84]. Therefore, \( R_t \) needs to learn only the information of the object and updates only when the peak value of the corresponding response map \( r(z) \) exceeds the fixed threshold \( \tau \) (indicating that the object’s state is reliable).

When the peak value of the response map is \( \max(r(z)) \), the LCT tracker judges that tracking failure has occurred. Then it turns on a random fern redetector to search for the object and updates \( R_t \) only when the peak of the corresponding response map exceeds the threshold to ensure the reliability of \( R_t \). Note that other tracking-by-detection methods usually perform redetection frame by frame, while the LCT tracker redetects only when tracking failure occurs, which ensures both speed and robustness.

Later, Ma et al. further proposed the LCT2.0 tracker [94], where the random fern classifier in the LCT tracker is substituted for an SVM classifier, and applied a histogram of local intensity as an additional expression.

**ECO-HC TRACKER**

Although the results of the previous work [72] are considerable, it has two serious problems. First, the operation speed is slow due to excessive calculation. Second, too many parameters are required, e.g., too many feature dimensions and too many frames involved, made the tracker prone to overfitting. Therefore, on the basis of their previous work [72], Danelljan et al. further proposed a faster and more robust ECO tracker [95], which improved the performance of the CF tracking method to a new level. Note that, here, we only discuss the ECO tracker using handcrafted features instead of CNN features, i.e., ECO-HC.

In [95], Danelljan et al. first analyzed the time complexity of the C-COT tracker, which can be expressed as \( O(N_{CC}DTK) \). \( N_{CC} \) denotes the number of conjugate gradient iterations, \( D \) indicates the number of feature channels of the training samples, \( T \) is the total number of frames involved in the operation, and \( K \) represents the average number of Fourier coefficients of each filter channel. Based on such analysis, three methods, respectively, are proposed to reduce computational complexity.

First, the ECO tracker reduces the number of feature dimensions for higher speed. The original \( D \) dimension is reduced to \( E \) dimension samples, and a matrix \( \mathbf{G} \) is introduced to represent the original samples as

\[
\mathbf{x} = \begin{pmatrix} \mathbf{x}^1 \\ \vdots \\ \mathbf{x}^E \end{pmatrix} = \begin{pmatrix} G_{x1} & \cdots & G_{xe} \\ \vdots & \ddots & \vdots \\ G_{e1} & \cdots & G_{ee} \end{pmatrix} \begin{pmatrix} \mathbf{x}^1 \\ \vdots \\ \mathbf{x}^E \end{pmatrix} = \mathbf{G} \mathbf{x},
\]

which realized dimensionality reduction and then updated the loss function to binary nonlinear regression as

\[
\mathcal{E}(\mathbf{w}_f; \mathbf{G}) = \sum_{f=1}^{E} \theta_f \sum_{c=1}^{E} \| \mathbf{w}_f^c \cdot \mathbf{G} \mathbf{x}_c^f - y_c^f \|_2^2 + \lambda \| \mathbf{G} \|_F^2.
\]

Note that \( \mathcal{J}(\{x\}) \) denotes the new training samples in the transformed continuous time-space domain [72]. \( \| \cdot \|_F^2 \) indicates the Frobenius norm. Utilizing a conjugate gradient method, \( \mathbf{w} \) and \( \mathbf{G} \) can be solved.

Second, the ECO tracker simplifies and compresses the prior sample model using the Gaussian mixture model to prevent overfitting. The main equation can be expressed as...
\[
E(w, G) = \sum_{i=1}^{N} \pi_i \left[ \sum_{j=1}^{M} w_j^i \cdot f(G \mu_j^i) - y \right]^2 \\
+ \sum_{i=1}^{N} \|w^i \|^2 + \lambda_1 \sum_{i=1}^{N} \|w^i \|^2, 
\]

where the previous samples in the \(fth\) frame \(x_f\) are replaced by Gaussian means \(\mu_f\), weight \(y_i\) is substituted by \(\pi_i\), and the number of total frames \(T\) is reduced to \(V\).

Finally, in the ECO algorithm, the filter model is updated every five frames instead of frame by frame. Thus, the ECO tracker achieved both more favorable performance and amazing speed.

**CONTEXT-AWARE CF TRACKERS**

Although the spatial punishment in the SRDCF tracker [90] can repress background noises, it utilizes a fixed global punishment, that is, artificially reducing the weight of the filter periphery to suppress the background response. As a means of improving the discriminative ability of the filter, learning background information, especially, can usually achieve a better effect. Aiming to learn context information, Mueller et al. proposed the context-aware CF (CACF) tracker [96].

In the CACF tracker, during the training stage, the authors changed the original ridge regression formula (3) to

\[
E(w) = \left[ \sum_{i=1}^{N} w^i \cdot \bar{x}^i - \bar{y} \right]^2 + \lambda_1 \sum_{i=1}^{N} \|w^i \|^2 \\
+ \lambda_2 \sum_{i=1}^{N} \|w^i \|^2, 
\]

where \(x_0\) denotes the training samples from the object region, and \(x^i\) represents the samples from the context around the object region.

**Remark 14**: Innovatively, to obtain a structure similar to the ridge regression equation, the authors further derive the deformation of the main formula as

\[
\bar{E} = \left[ \sum_{i=1}^{N} w^i \cdot \bar{x}^i - \bar{y} \right]^2 + \lambda_1 \sum_{i=1}^{N} \|w^i \|^2, 
\]

where \(\bar{x}^i\) and \(\bar{y}\) can be expressed as

\[
\bar{x}^i = \begin{bmatrix} x_0^i \\ \sqrt{\lambda_2} x_1^i \\ \vdots \\ \sqrt{\lambda_2} x_N^i \end{bmatrix}, \quad \bar{y} = \begin{bmatrix} y \\ 0_N \end{bmatrix},
\]

where \(0_N \in \mathbb{R}^N\) denotes a zero vector. The solution to (60) can be obtained:

\[
\hat{w}^i = \frac{\hat{x}^i \odot \hat{y}}{\hat{x}^i \odot \hat{x}^i + \lambda_1 + \lambda_2 \sum_{i=1}^{N} \hat{x}^i \odot \hat{x}^i}. 
\]

The experiments have shown that trackers with the context-aware algorithm in the CACF tracker, i.e., the Staple_CA and SAMF_CA trackers, can be more robust when compared with the original tracker in most scenes.

**Remark 15**: Although the CACF tracker [96] can make the filter learn the background information near the object’s bounding box, it has an obvious shortcoming: the background information learned by the CACF tracker is the four fixed background boxes in the relative position around the object, which is not enough for the tracker to tackle complex scenes.

**KCC TRACKER**

The kernel trick used in the KCF tracker [39] can map samples \(x\) to a high-dimensional space for division and thus has a relatively good shielding effect on noise and distractors. However, there are two obvious limitations in the KCF tracker: 1) due to the application of (9), the training samples must be circulant; and 2) the kernel function used, \(k^{xx}\), is also required to have the same weight for each pixel in the samples.

In the KCC tracker [85], Wang et al. unearthed the huge potential of the kernel method, which not only eliminates the two limitations in the KCF tracker but also further expands the kernel method to the calculation of changes such as scale and rotation (not just translation), and proved to be more robust than the scaling pool in the SAMF tracker [83] and the scale filter in the DSST tracker [84].

In [85], the authors first defined a kernel vector \(x_k(z) = [x(z, x_1), \ldots, x(z, x_N)]\), where \(x\) denotes any affine transform of \(x\), which is not a limited translation. \(x()\) is the kernel function, the same as \(k\). Then, the kernel cross correlator (response output) is defined as

\[
\hat{r} = \sum_{i=1}^{N} \hat{x}_k(x^i) \odot \hat{w}^i. 
\]

Based on this definition, the KCC tracker proposed their training regression equation as

\[
E(w) = \sum_{i=1}^{N} \| \hat{x}_k(x^i) \odot \hat{w}^i - \hat{y} \|^2 + \lambda \sum_{i=1}^{N} \| \hat{w}^i \|^2. 
\]

By applying a partial derivative, a closed solution to (64) can be obtained:

\[
\hat{w}^i = \frac{\hat{y} \odot \hat{x}_k(x^i)}{\hat{x}_k(x^i) \odot \hat{x}_k(x^i) + \lambda}. 
\]

**Remark 16**: As this result was derived without any theorems or restrictions, it is theoretically possible to predict any affine transform of \(x\), e.g., translation (the KTC tracker), scale change (the KSC tracker), and rotation (the KRC tracker), and apply any kernel function.

**EXPERIMENTAL EVALUATION AND ANALYSIS**

This section presents the experimental results and analysis, divided into five sections. The “Implementation
Information” section first introduces some implementation information, including the commonly used evaluation metrics for object tracking, and the experiment platform, parameter settings, and benchmarks used in the experiment. Second, a comprehensive analysis of the overall performance and the tracking results of DCF-based trackers is given in the “Overall Performance of DCF-Based Trackers” section. Moreover, the “Performance Analysis by UAV Special Attributes” section proposes the redefined UAV tracking attributes and analyzes the performance of DCF-based trackers by attribute. Then, in the “Against Deep Trackers” section, the performance of the DCF-based tracker (using only the handcrafted feature) against the deep trackers is supplemented. Finally, on the basis of the comprehensive experiments, in the “Failure Cases and Challenges” section, we conclude the typical failure cases and challenges that have not been well addressed in DCF-based methods for UAV tracking.

IMPLEMENTATION INFORMATION

Before stepping into an experimental evaluation and analysis, some experimental implementation information is given first in this section, i.e., the two metrics adopted in the experiment for the trackers’ evaluation, the experiment platform where all the experiments were extended, the parameter settings in the implemented code, and the UAV benchmarks utilized in the experiment.

EVALUATION METRICS

The two authoritative and objective evaluation metrics commonly used in object tracking are introduced, namely, the center-location error (CLE) and overlap score (OS), based on a one-pass evaluation (OPE) [28]. An OPE refers to initializing the first frame with the location and size of the object in the ground truth and then running the tracking algorithm to obtain the bounding boxes in subsequent frames, which can be used to draw both the precision and success-rate plots.

To obtain the precision plot, the CLE needs to be calculated in every frame, which is defined by the distance between the center point of the bounding box estimated by the tracker and the center point in the ground truth, as presented in Figure 5. By calculating the percentage of all the video frames in a sequence where the CLE is lower than a given threshold, a pair of precision scores and the threshold are obtained. For an evaluation of the whole benchmark, the final precision score can be produced by averaging the scores from all the sequences. Different thresholds result in different percentages; therefore, a precision curve plot can be obtained, as exhibited in Figure 6. The threshold is set to 20 pixels during the general evaluation (also in this experiment) for the final ranking of all the trackers, i.e., distance precision (DP) at a CLE = 20 pixels.

For the success-rate plot, the OS is first calculated in every frame, as depicted in Figure 5(a) and (b). Utilizing the bounding box obtained by the tracking algorithm (indicated as “a”), and the box given by the ground truth (indicated as “b”), the OS can be obtained by

\[
OS = \frac{|a \cap b|}{|a \cup b|},
\]

where \(| \cdot |\) represents the number of pixels in the area. When the OS of a frame is greater than the set threshold, this frame is regarded as a successful one, and the percentage of the total successful frames to all frames is the success rate under one threshold. The value of the OS ranges from zero to one, so a curve plot can be drawn, which is the success-rate plot in Figure 6. In the general evaluation (and also in this experiment), the area under the curve (AUC) is calculated as the trackers’ ranking standard. For ease of understanding, Figure 5(a) and (b) graphically shows the two evaluation metrics.

EXPERIMENT PLATFORM

The large-scale evaluation experiments presented in this article were extended on MATLAB R2019a. The main hardware consists of a single Intel Core i7-8700K CPU,
FIGURE 6. The overall performance of handcrafted DCF-based trackers on (a) a UAVDT [20] and (b) UAV123 [19]. The ranking standard in the precision plot is the precision under CLE = 20 pixels, and the standard in the success-rate plot is the AUC. (Continued)
FIGURE 6. (Continued) The overall performance of handcrafted DCF-based trackers on (c) DTB70 [18] and (d) UAV123@10fps [19]. The ranking standard in the precision plot is the precision under CLE = 20 pixels, and the standard in the success-rate plot is the AUC (area under curve). (Continued)
FIGURE 6. (Continued) The overall performance of handcrafted DCF-based trackers on (e) UAV20L [19] and (f) VisDrone2019-SOT [21]. The ranking standard in the precision plot is the precision under CLE = 20 pixels, and the standard in the success-rate plot is the AUC (area under curve).
32 GB of random-access memory (RAM), and an NVIDIA RTX 2080 GPU.

PARAMETER SETTINGS
To ensure the fairness and objectivity of the experiment, all of the trackers evaluated have maintained their official initial parameters. For the trackers that use various features, such as the ECO [95] and ARCF trackers [24], the specific features used in the experiment are noted, e.g., the ARCF-H tracker utilizes a fHOG feature [75] only, and the ARCF-HC tracker uses fHOG, CN [76], and gray scale.

BENCHMARKS
The experiments used a total of six well-known authoritative UAV tracking benchmarks, i.e., UAV123, UAV20L, UAV123@10fps [19], UAVDT [20], DTB70 [18], and VisDrone2019-SOT [21]. In this section, we introduce the characteristics of each benchmark one by one.

Mueller et al. [19] compiled benchmark UAV123, which contains 123 fully annotated, high-definition video sequences captured from a low-altitude aerial perspective, including a total of 112,578 frames that cover a wide variety of scenes and objects. As a subset of UAV123, UAV20L is designed especially for long-term tracking and includes the 20 longest sequences. For investigating the impact of the camera’s capture speed on tracking performance, Mueller also temporally downsampled the UAV123 benchmark to 10 fps where most of the sequences are originally provided at 30 fps; thus, the benchmark UAV123@10fps was created. Note that, because the frame interval becomes larger, the object’s location changes between frames becomes bigger, increasing the difficulty of accurate tracking.

Consisting of 70 videos and totaling 15,777 frames, DTB70 was built by Li and Yeung [18]. The highlight of DTB70 is its attention to the severe camera motion (CM) issue, which achieved real time on a UAV platform.

UAVDT [20], constructed by Du et al., contains 50 sequences and 37,084 frames (here, it refers to their SOT task). UAVDT primarily focuses on cars under a variety of new challenges, e.g., various weather conditions, flying altitude, and camera view.

For VisDrone2019-SOT [21], VisDrone2019-SOT-test-dev, VisDrone2019-SOT-val, and VisDrone2019-SOT-train are combined into a total of 132 sequences and 109,909 frames. This benchmark is from the VisDrone2019-SOT challenge, which focused on evaluating SOT algorithms on drones and was held in conjunction with the International Conference on Computer Vision.

Table 3 lists the numbers of sequences; the minimum, maximum, and mean frames in each sequence; and the total frames in six benchmarks.

OVERALL PERFORMANCE OF DCF-BASED TRACKERS
To verify the performance of DCF-based trackers in UAV tracking scenarios, 21 famous DCF-based trackers, using handcrafted features only, are selected, i.e., AutoTrack [22], ARCF-H [24], ARCF-HC [24], STRCF [66], CSR-DCF [91], ECO-HC [95], MCCT-H [89], BACF [92], DSST [84], fDSST [79], SAMF [83], SRDCF [71], SRDCFdecon [90], Staple [88], CACF (SAMF_CA and Staple_CA) [96], KCF (adopting a Gaussian kernel) and DCF (adopting a linear kernel) [39], LCT2.0 [94], KCC [85], and CN [87]. Their tracking results were obtained on the same platform and using the same six authoritative benchmarks [18]–[21]. The results show that DCF-based trackers not only have amazing accuracy and robustness, but many of them also possess real-time tracking speed on a single CPU, and thus are ideal algorithms for UAV tracking.

As displayed in Figure 6, the tracker performance under different benchmarks is usually different. AutoTrack performs best under DTB70, followed by ARCF-HC. Under UAV123, the performance of ECO-HC is outstanding, followed by AutoTrack. In UAVDT, ARCF-HC occupies first place, followed by AutoTrack. It can be seen that, with the development of research in recent years, the performance of DCF-based trackers has been incrementally improved. Table 4 lists all of the state-of-the-art, handcrafted DCF-based trackers’ DP, AUC, and speed comparison (most of which achieved real time on a UAV platform).

From the results in Table 4, it can be seen that the early DCF-based trackers, e.g., the KCF [39], and CN trackers [87], due to their simplicity, usually possess high tracking speed, but this results in poor accuracy and robustness. As the development of DCF-based trackers has continued, after the various problems and shortcomings of early DCF-based trackers—such as the lack of scale estimation—were paid attention to, settled, and further improved upon one by one, the tracking results of DCF-based methods have

| BENCHMARK | UAV123 | UAV123@10fps | UAV20L | DTB70 | UAVDT | VISDRAONE2019-SOT |
|-----------|--------|-------------|--------|-------|-------|-------------------|
| Sequences | 123    | 123         | 20     | 70    | 50    | 132               |
| Each sequence | Min.  | Max. | Mean | Min. | Max. | Mean | Min. | Max. | Mean | Min. | Max. | Mean | Min. | Max. | Mean | Min. | Max. | Mean | Min. | Max. | Mean | Min. | Max. | Mean | Min. | Max. | Mean | Min. | Max. | Mean |
| Total frames | 112,578 | 3,085 | 915 | 37 | 1,029 | 308 | 1,717 | 5,527 | 2,934 | 68 | 699 | 225 | 82 | 2,969 | 742 | 90 | 2,970 | 833 |

TABLE 3. THE NUMBERS OF SEQUENCES; THE MINIMUM, MAXIMUM, AND MEAN FRAMES IN EACH SEQUENCE; AND THE TOTAL FRAMES IN SIX BENCHMARKS, I.E., UAV123, UAV123@10FPS, UAV20L [19], UAVDT [20], DTB70 [18], AND VISDRAONE2019-SOT [21]. RED, GREEN, AND BLUE DENOTE FIRST, SECOND, AND THIRD PLACE, RESPECTIVELY.
also been significantly enhanced. To name a few, after the DSST [84] and SAMF [83] trackers solved the scale-estimation problem for the KCF tracker [39], the tracking accuracy and robustness have improved. After the CACF tracker (Staple_CA) [96] added a context-aware strategy, its performance has been much better, compared to the original tracker Staple [88]. The AutoTrack tracker [22], which uses the STRCF tracker [66] as the baseline, achieves better results than the STRCF tracker in most benchmarks through the spatiotemporal regularization terms that are automatically and adaptively updated.

Among all the DCF-based trackers using handcrafTed features, in the case of unadjusted parameters, three trackers, i.e., the AutoTrack [22], ARCF-HC tracker [24], and ECO-HC trackers [95], have achieved more top-three values (indicated by red, green, and blue in Table 4) in terms of the success rate (AUC) and precision (DP) in the six benchmarks than others. This indicates that, even with the fixed parameter, they still possess good universality in various complex scenes in each benchmark, which further demonstrates their practicability in real-world UAV tracking.

### PERFORMANCE ANALYSIS BY UAV SPECIAL ATTRIBUTES

In object tracking, to evaluate the performance of the trackers in various challenging scenes, each benchmark puts forward a special series of tracking scenes, called the *attribute*. Whether they have or not, the defined attributes are indicated in each sequence for further comparison of the trackers in terms of special attribute. The attributes’ full names in Table 5 are listed as follows: FM, CM, camera rotation (CR), FOC, OCC, partial OCC (POC), large OCC (LO), LV, low resolution (LR), SO, and VC.

To better illustrate the performance of DCF-based trackers in responding to different challenges in UAV tracking scenarios, the six benchmarks’ most commonly encountered tracking scenarios are summarized into FM, VC, LR, OCC, and IV.

The original attributes of each benchmark were first mapped to the five attributes, and each sequence was relabeled.
for UAV123, UAV123@10fps, UAV20L, and VisDrone2019-SOT, whose original attributes are the same. Their attributes of CM and VC are classified as VC. The original attributes of FM, LR, and IV are the same as in this work. POC and FOC are classified as OCC. For DTB70, their original attribute, fast CM, is classified as VC, and OCC is classified as OCC in this work. For UAVDT, their CR, SO, IV, and LO attributes correspond to VC, LR, IV, and OCC, respectively. Table 5 lists the correspondence between the original attributes in the six benchmarks and the new attributes, as well as the serial number contribution of each benchmark to each new attribute. Above the horizontal line is the sequence number of each benchmark under the new attribute, and below is the original attribute(s) of each benchmark corresponding to the new attribute.

Based on this work, the tracking results of each tracker under the new attributes were also drawn into 10 plots containing success rate and precision plots, as depicted in Figure 7(a)–(e). Note that the calculation method adopted is used to average by sequence, that is, to count the sequences containing specific attributes in all the sequences in the six benchmarks, and the arithmetic average of the results of sequences involved is used as the final result. Figure 7(a)–(e) shows that the performance of the trackers under specific attributes is obviously worse than the overall performance in each benchmark, and the challenges that each tracker does well in are also different. Overall, the ARCF-HC tracker [24] performs the best in FM and VC, AutoTrack [22] is good at VC and IV, and ECO-HC [95] ranked first in OCC issues.

Figure 8 presents the capability comparison of the three best-performing trackers under different challenges, i.e., the ARCF-HC [24], AutoTrack [22], and ECO-HC trackers [95]. As far as each attribute is concerned, the performance of the three trackers under the two attributes, OCC and FM, is worse than that of IV, LR, and VC, indicating that the two major tracking scenarios of OCC and FM are currently more challenging in UAV tracking. As far as the three trackers are concerned, when comparing the success rates, the three have a similar overall performance in VC; the ECO tracker [95] has a higher success rate in IV, LR, and OCC; and the ARCF-HC tracker [24] does better in OCC. The ARCF-HC and AutoTrack trackers both do better in LR compared to the ECO-HC tracker. In the precision comparison, there is little difference among the three in IV, while the overall precision of the ARCF-HC [24] and AutoTrack trackers [22] is significantly higher than that of the ECO-HC tracker [95]. The ARCF-HC, AutoTrack, and ECO-HC trackers are more accurate under FM, LR, and OCC, respectively.

TABLE 5. THE CORRESPONDENCE BETWEEN THE ORIGINAL ATTRIBUTES (UNDER THE LINE) IN THE SIX BENCHMARKS AND THE NEW ATTRIBUTES AS WELL AS THE SERIAL NUMBER CONTRIBUTION OF EACH BENCHMARK TO EACH NEW ATTRIBUTE.

| BENCHMARK   | VC | FM | LR | OCC | IV |
|-------------|----|----|----|-----|----|
| UAV123      | 70 | 28 | 30 | 54  | 73 |
|             | CM | VC | FM | LR  | POC| FOC| FOC| IV |
| UAV123@10fps| 70 | 28 | 30 | 54  | 73 |
|             | CM | VC | FM | LR  | POC| FOC| FOC| IV |
| UAV20L      | 16 | 7  | 13 | 13  | 18 |
|             | CM | VC | FM | LR  | POC| FOC| FOC| IV |
| VisDrone2019-SOT| 109| 41 | X  | 20  | 79 |
|             | CM | VC | FM | LR  | POC| FOC| FOC| IV |
| DTB70       | 41 | FCM| X  | X   | 17 |
|             | X  | OCC|     |     |    |
| UAVDT       | 30 | X  | 23 | 20  | 28 |
|             | CR | SO | LO | IV  |
| Total sequences | 336| 91 | 116| 237 | 247 |

AGAINST DEEP TRACKERS

To better demonstrate the superiority of the DCF-based trackers using handcrafted features in the UAV tracking scenes, various deep trackers are selected, including those adopting a basic CF framework but using CNN features, i.e., the ASRCF [73], ECO [95], CFWR [64], MCCI [89], MPCF [65], DeepSTRCF [66], CoKCF [101], IBCCF [102], and HCF [67], and also the end-to-end CNN trackers, i.e., the UDT [98], SiamFC [57], CFNet_conv2 [99], TADT [100], UDT [98], and DSiam trackers [58].

Remark 17: All of the deep trackers used GPU acceleration in the experiment, and all of the DCF-based trackers using handcrafted features, e.g., the AutoTrack tracker, were evaluated on a single CPU using one core only.

Remark 18: According to Table 4 and focusing on the AUC and DP, the AutoTrack tracker [22], which achieved the most top-three scores and the most first-rank scores in six benchmarks, is considered the most outstanding DCF-based tracker. This section chooses AutoTrack to demonstrate the DCF-based trackers’ superiority against the deep trackers in UAV tracking.

Table 6 lists their DP and tracking speed under benchmark UAVDT [20]. As is demonstrated in their tracking performance, even with low-cost handcrafted features, AutoTrack tracker [22] still prevailed over most of the deep trackers, achieving both excellent precision and considerable tracking speed.

According to the experimental results, even compared with the deep trackers, the handcrafted DCF-based trackers still maintain their strong competitiveness in accuracy and possess real-time performance, which further confirms that the DCF-based trackers using handcrafted features, e.g., the AutoTrack tracker [22] and so forth, are the best choice for UAV-based aerial tracking.

FAILURE CASES AND CHALLENGES

Although DCF-based methods have exhibited their superiority in UAV tracking, there still exist tracking challenges that have not been well addressed. This section analyzes...
FIGURE 7. The tracking results of each tracker under different attributes. (a) VC and (b) FM. The number after the attribute indicates the total number of sequences with the specific attribute, e.g., the total number of sequences with the FM attribute in the six benchmarks is 69. The ranking standard in the precision plot is the precision (DP) under $CLE = 20$ pixels, and the standard in the success-rate plot is the AUC. (Continued)
FIGURE 7. (Continued) The tracking results of each tracker under different attributes. (c) LR and (d) OCC. The number after the attribute indicates the total number of sequences with the specific attribute, e.g., the total number of sequences with the FM attribute in the six benchmarks is 69. The ranking standard in the precision plot is the precision (DP) under CLE = 20 pixels, and the standard in the success-rate plot is the AUC. (Continued)
Figure 7. (Continued) The tracking results of each tracker under different attributes. (e) IV. The number after the attribute indicates the total number of sequences with the specific attribute, e.g., the total number of sequences with the FM attribute in the six benchmarks is 69. The ranking standard in the precision plot is the precision (DP) under CLE = 20 pixels, and the standard in the success-rate plot is the AUC.

Figure 8. The performance comparison of the three best-performing trackers. This figure uses radar charts to show their capabilities under different attributes, whose specific values are in Figure 7. Note that the overall data are also the average values of all the sequences in the six benchmarks.
the classic tracking failure cases of the five best-performing trackers, the AutoTrack [22], ARCF-HC [24], STRCF [66], ECO-HC [95], and BACF trackers [92], in each benchmark in the experiment to illustrate the current limitations and challenges of handcrafted DCF-based trackers. Figure 9(a)–(f) depicts the representative tracking failures in the different benchmarks.

1) Rapid scale changes and other appearance variations are currently difficult for DCF-based trackers to deal with. In Figure 9(d) and (e), when the object undergoes rapid appearance variations caused by VC or scale variations, the trackers cannot adapt to the appearance changes in time, making the wrong location and scale estimations. Such scenes usually cause the filter to learn the wrong object information and eventually lead to tracking failure.

2) The OCC problem is often hard to tackle by the trackers. In Figure 9(b), when the object is completely occluded and appears outside the search region of the trackers again, the trackers cannot predict the object’s location. In Figure 9(f), even POC can seriously affect the object template learned by the tracker, resulting in tracking failure.

3) LR objects are much more challenging to track than other objects. As presented in Figure 9(a) and (c), LR objects lead to insufficient training samples of the filter, which may reduce the filter’s ability to discriminate the object from the background. When such an object undergoes FM, it can more easily cause tracking failure due to the filter’s poor discriminating ability.

In Figure 9(e), the SO can also cause inaccurate scale estimation: when the object scale changes, the tracker can’t adapt in time.

4) Poor lighting conditions and IV can make tracking harder. As shown in Figure 9(e) and (f), in a relatively dim environment, the filter cannot learn enough representative object features; thus, it is difficult to distinguish the object from the environment. Under such conditions, the presence of scale variation, POC, or a similar object makes robust tracking even more difficult.

**ONBOARD EVALUATION**

Apart from the aforementioned large-scale evaluation experiments, this work also extended an onboard test to further validate the real-time capability and robustness of the outstanding DCF-based trackers [22], [24]. This evaluation adopted a typical CPU-based onboard PC for UAV, that is, an Intel NUC8i7HVK, which contains a single Intel Core i7-8809G CPU and 32 GB of RAM as the test platform.

The onboard tracking performance of six tests is shown in Figure 10 with the AutoTrack [22] and ARCF-H [24] trackers. The six tests, i.e., Figure 10(a)–(f), contain the challenges, e.g., LR, IV, VC, and so forth, commonly encountered in UAV tracking (including one long-term tracking sequence). Table 7 displays the challenges in each test and the running speed of the two trackers, where they both surpass 30 fps, realizing real-time processing. In Figure 10, the CLEs of the two trackers in six tests are all smaller than 20 pixels, indicating that the trackers maintained satisfying robustness in real-world, challenging UAV tracking scenes.

**Remark 19:** Onboard tracking has verified the promising efficiency of DCF-based methods. Such superiority in computation saves scarce power supply for other energy-consuming functions like self-control in strong wind on board a UAV.

**FUTURE WORKS**

Future investigations and improved work on DCF-based trackers can be summarized into the following four points:

1) Exploit more adaptive CFs, with different learning rates or adaptive parameters for various tracking scenes and objects.

2) Conduct research for a more intelligent search strategy, which can better adapt to the situations where objects move faster or reappear at a distant location after being occluded.

3) Embed image preprocessing strategies into basic tracking structures to boost the trackers’ discriminative ability under complex scenes. Such strategies include low-light image enhancers, object-segmentation methods, saliency-detection algorithms, and so on.

4) Study multimodal trackers, such as infrared modalities, to cope with tracking scenes with poor lighting conditions.

The prospect of DCF-based methods’ onboard UAV tracking is promising, which can promote the development
and application of UAVs, thus boosting the progress of the remote sensing field.

**CONCLUSIONS**
This article first introduced the tracking scenes on board a UAV, the uniqueness and challenge of UAV tracking compared with general tracking scenes, and what makes DCF-based methods suitable for UAV-based aerial tracking in comparison with other types of trackers. Next, the DCF-based tracking algorithms’ common, basic architecture was proposed for overall understanding. Third, this work introduced the highlights of the noted DCF-based trackers, focusing on their contributions, thus integrating DCF-based trackers’ developments over the years. Then, in the “Experimental Evaluation and Analysis” section, having introduced some implementation

![Figure 9](image-url)
FIGURE 10. The onboard tracking performance in terms of the CLE. The six tests [(a)–(f)] where the tracking objects are marked with red boxes contain the aforementioned common UAV tracking challenges. The satisfying CLE results displayed validate the robustness of the AutoTrack [22] and ARCF-H [24] trackers in challenging real-world UAV tracking.
information, exhaustive experiments were conducted on six UAV tracking benchmarks to estimate all of the aforementioned DCF-based trackers (both generally and by attribute) and demonstrate the superiority of their tracking. Based on the experiment results, this article further analyzed the current tracking challenges. Moreover, additional onboard tracking tests were extended to validate the real-time capability and robustness of the DCF-based trackers in challenging real-flight UAV tracking tasks. Finally, future research directions and ideas for improved work were summarized, enabling more benefits of DCF for UAV tracking.

ACKNOWLEDGMENTS
This work was supported by the National Natural Science Foundation of China under grant 61806148 and the Natural Science Foundation of Shanghai under grant 20ZR1460100. Changhong Fu is the corresponding author.

AUTHOR INFORMATION
Changhong Fu (changhongfu@tongji.edu.cn) received his Ph.D. degree in robotics and automation from the Computer Vision and Aerial Robotics Laboratory, Technical University of Madrid, Madrid, Spain, in 2015. He is currently an associate professor with the School of Mechanical Engineering, Tongji University, Shanghai, 201804, China, and is leading seven projects related to vision for unmanned systems (USSs). During his Ph.D. studies, he held two research positions, at Arizona State University, Tempe, USA, and Nanyang Technological University (NTU), Singapore. After receiving his Ph.D. degree, he worked at NTU as a postdoctoral research fellow. He has worked on more than 10 projects related to unmanned aerial vehicles (UAVs). In addition, he has published more than 60 journal and conference papers (including in IEEE Transactions on Geoscience and Remote Science, IEEE Transactions on Circuits and Systems for Video Technology, IEEE Transactions on Multimedia, IEEE/ASME Transactions on Mechatronics, and IEEE Transactions on Industrial Electronics and at the Conference on Computer Vision and Pattern Recognition, International Conference on Computer Vision, IEEE International Conference on Robotics and Automation, and IEEE/RSJ International Conference on Intelligent Robots and Systems) related to intelligent vision and control of UAVs. His research interests include intelligent vision and control of USSs in a complex environment. He is a Member of IEEE.

Bowen Li (1854152@tongji.edu.cn) is pursuing his B.Eng. degree with the School of Mechanic Engineering, Tongji University, Shanghai, 201804, China. His research interests include robotics, artificial intelligence, and computer vision.

Fangqiang Ding (1753436@tongji.edu.cn) is pursuing his B.Eng. degree with the School of Mechanical Engineering, Tongji University, Shanghai, 201804, China. His research interests include unmanned aerial vehicles and computer vision.

Fuling Lin (lin fuling5@tongji.edu.cn) received his B.Eng. degree in mechanical engineering from Tongji University, Shanghai, 201804, China, where he is currently pursuing his M.Sc. degree. His research interests include robotics, visual object tracking, and computer vision.

Geng Lu (lug@tsinghua.edu.cn) received his B.E., M.E., and Ph.D. degrees from the Department of Automation, Tsinghua University, Beijing, China, in 1999, 2002, and 2004, respectively. From 2004 to 2006, he was a postdoctoral scholar in the Department of Electrical Engineering, Tsinghua University. Since 2006, he has been with the Department of Automation, Tsinghua University, Beijing, 10084, China, where he is currently an associate professor. His research interests include robust control, nonlinear control, signal processing, and aerial robots.

REFERENCES
[1] Y. Cao, G. Wang, D. Yan, and Z. Zhao, “Two algorithms for the detection and tracking of moving vehicle targets in aerial infrared image sequences,” Remote Sens., vol. 8, no. 1, pp. 1–21, 2016. doi: 10.3390/rs8010028.
[2] F. Bi, M. Lei, Y. Wang, and D. Huang, “Remote sensing target tracking in UAV aerial video based on saliency enhanced MDnet,” IEEE Access, vol. 7, pp. 76,731–76,740, June 2019. doi: 10.1109/ACCESS.2019.2921315.
[3] B. Uzkent, A. Rangnekar, and M. J. Hoffman, “Tracking in aerial hyperspectral videos using deep kernelized correlation filters,” IEEE Trans. Geosci. Remote Sens., vol. 57, no. 1, pp. 449–461, 2019. doi: 10.1109/TGRS.2018.2856370.
[4] B. Uzkent, M. J. Hoffman, and A. Vodacek, “Integrating hyperspectral likelihoods in a multidimensional assignment algorithm for aerial vehicle tracking,” IEEE J. Sel. Topics Appl. Earth Observat. Remote Sens., vol. 9, no. 9, pp. 4325–4333, 2016. doi: 10.1109/JSTARS.2016.2560220.
[5] C. Fu, J. Xu, F. Lin, F. Guo, T. Liu, and Z. Zhang, “Object saliency-aware dual regularized correlation filter for real-time aircraft tracking,” IEEE Trans. Geosci. Remote Sens., vol. 56, no. 12, pp. 8940–8951, 2020. doi: 10.1109/TGRS.2020.2992301.
[6] W. Yi, C. Liming, K. Lingyu, Z. Jie, and W. Miao, “Research on application mode of large fixed-wing UAV system on overhead

| TEST | CHALLENGES | AUTOTRACK | ARCF-H | FPS |
|------|------------|-----------|--------|-----|
| (a)  | LR and OCC | 33.58     | 50.47  |     |
| (b)  | VC and LR  | 37.59     | 68.05  |     |
| (c)  | LR         | 36.95     | 81.22  |     |
| (d)  | LR, OCC, and IV | 36.42 | 43.47  |     |
| (e)  | VC, LR, and IV | 36.92 | 57.48  |     |
| (f)  | Long term  | 40.21     | 57.51  |     |

MARCH 2022 IEEE GEOSCIENCE AND REMOTE SENSING MAGAZINE 157
transmission line,” in *Proc. IEEE Int. Conf. Unmanned Syst. (ICUS)*, 2017, pp. 88–91.

[7] X. Gu, T. Liu, D. Yang, K. Huang, and Y. Tian, “Research on application of maritime navigation technology for fixed-wing UAV,” in *Proc. IEEE Inf. Technol., Netw., Electron. Automat. Control Conf. (ITNEC)*, 2019, pp. 2585–2589.

[8] Y. Shi, M. Bai, Y. Li, and Y. Li, “Study on UAV remote sensing technology in irrigation district information construction and application,” in *Proc. Int. Conf. Measuring Technol. Mechatron. Automation (ICMTMA)*, 2018, pp. 252–255.

[9] M. A. Oliveira-Mendes et al., “Towards an autonomous vision-based unmanned aerial system against wildlife poachers,” *Sensors*, vol. 15, no. 12, pp. 31,362–31,391, 2015. doi: 10.3390/s151229861.

[10] F. Vanegas, D. Campbell, N. Roy, K. J. Gaston, and F. González, “UAV tracking and following a ground target under motion and localisation uncertainty,” in *Proc. IEEE Aerospace Conf. (AC)*, 2017, pp. 1–10.

[11] H. Cheng, L. Lin, Z. Zheng, Y. Guan, and Z. Liu, “An autonomous vision-based target tracking system for rotorcraft unmanned aerial vehicles,” in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, 2017, pp. 1732–1738.

[12] N. Yao, E. Anaya, Q. Tao, S. Cho, H. Zheng, and F. Zhang, “Monocular vision-based human following on miniature robotic blimp,” in *Proc. IEEE Int. Conf. Robot. Automat. (ICRA)*, 2017, pp. 3244–3249.

[13] C. Yuan, Z. Liu, and Y. Zhang, “Aerial images-based forest fire detection for firefighting using optical remote sensing techniques and unmanned aerial vehicles,” *J. Intell. Robot. Syst.*, vol. 88, nos. 2–4, pp. 635–654, 2017. doi: 10.1007/s10846-016-0464-7.

[14] C. Yuan, Z. Liu, and Y. Zhang, “UAV-based forest fire detection and tracking using image processing techniques,” in *Proc. Int. Conf. Unmanned Aircraft Syst. (ICUAS)*, 2015, pp. 639–643.

[15] R. Bonatti, C. Ho, W. Wang, S. Choudhury, and S. Scherer, “Towards a robust aerial cinematography platform: localizing and tracking moving targets in unstructured environments,” in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, 2019, pp. 229–236.

[16] W. M. Martins, A. J. D. Filho, L. D. de Jesus, A. D. de Souza, A. C. Ramos, and T. C. Pimenta, “Tracking for inspection in energy transmission power lines using unmanned aerial vehicles (UAV): A systematic review of current and specific literature,” *Int. J. Robot. Automat. (IJRA)*, vol. 9, no. 4, pp. 233–243, 2020. doi: 10.11591/ijra.v9i4.pp.233-243.

[17] P. Hamelin et al., “Discrete-time control of LineDrone: An assisted tracking and landing UAV for live power line inspection and maintenance,” in *Proc. Int. Conf. Unmanned Aircraft Syst. (ICUAS)*, 2019, pp. 292–298.

[18] S. Li and D.-Y. Yeung, “Visual object tracking for unmanned aerial vehicles: Benchmark and new motion models,” in *Proc. AAAI Conf. Artif. Intell. (AAAI)*, 2017, pp. 1–7.

[19] M. Mueller, N. Smith, and B. Ghanem, “A benchmark and simulator for UAV tracking,” in *Proc. European Conf. Comput. Vision (ECCV)*, 2016, pp. 445–461.

[20] D. Du et al., “The unmanned aerial vehicle benchmark: Object detection and tracking,” in *Proc. European Conf. Comput. Vision (ECCV)*, 2018, pp. 370–386.

[21] D. Du et al., “VisDrone-SOT2019: The vision meets drone single object tracking challenge results,” in *Proc. Int. Conf. Comput. Vision Workshops (ICCVW)*, 2019, pp. 199–212.

[22] Y. Li, C. Fu, F. Ding, Z. Huang, and G. Lu, “AutoTrack: Towards high-performance visual tracking for UAV with automatic spatio-temporal regularization,” in *Proc. IEEE Conf. Comput. Vision Pattern Recognit. (CVPR)*, 2020, pp. 11,920–11,929.

[23] F. Lin, C. Fu, Y. He, F. Guo, and Q. Tang, “BiCF: Learning bi-directional incongruity-aware correlation filter for efficient UAV object tracking,” in *Proc. Int. Conf. Robot. Automat. (ICRA)*, 2020, pp. 2365–2371.

[24] Z. Huang, C. Fu, Y. Li, F. Lin, and P. Lu, “Learning aberrance repressed correlation filters for real-time UAV tracking,” in *Proc. IEEE Int. Conf. Comput. Vision (ICCV)*, 2019, pp. 2891–2900.

[25] C. Fu, Y. Zhang, R. Duan, and Z. Xie, “Robust scalable part-based visual tracking for UAV with background-aware correlation filter,” in *Proc. IEEE Int. Conf. Robot. Biomimetics (ROBIO)*, 2018, pp. 2245–2252.

[26] Y. Li, C. Fu, F. Ding, Z. Huang, and J. Pan, “Augmented memory for correlation filters in real-time UAV tracking,” in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, 2020, pp. 1559–1566.

[27] F. Ding, C. Fu, Y. Li, J. Jin, and C. Feng, “Automatic failure recovery and re-initialization for online UAV tracking with joint scale and aspect ratio optimization,” in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, 2020, pp. 5970–5977.

[28] Y. Wu, J. Lim, and M.-H. Yang, “Object tracking benchmark,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 37, no. 9, pp. 1834–1848, 2015. doi: 10.1109/TPAMI.2014.2388226.

[29] S. Liu, D. Liu, G. Srivastava, D. Polap, and M. Woźniak, “Overview and methods of correlation filter algorithms in object tracking,” *Complex Intell. Syst.*, June 2020, pp. 1–23.

[30] F. Nex and F. Remondino, “UAV for 3D mapping applications: A review,” *Appl. Geomatics*, vol. 6, no. 1, pp. 1–15, 2014. doi: 10.1007/s12518-013-0120-x.

[31] T. Adão et al., “Hyperspectral imaging: A review on UAV-based sensors, data processing and applications for agriculture and forestry,” *Remote Sens.*, vol. 9, no. 11, pp. 1–30, 2017. doi: 10.3390/rs9111110.

[32] B. N. Chand, P. Mahalakshmi, and V. P. S. Naidu, “Sense and avoid technology in unmanned aerial vehicles: A review,” in *Proc. Int. Conf. Electr., Electron., Commun., Comput. Optimizat. Techn.*, 2017, pp. 512–517.

[33] J. Hu et al., “A brief review on the positioning technologies for unmanned aerial vehicles,” in *Proc. IEEE Int. Conf. Unmanned Syst. (ICUS)*, 2017, pp. 527–532.

[34] B. Yang, “Projection approximation subspace tracking,” *IEEE Trans. Signal Process.*, vol. 43, no. 1, pp. 95–107, 1995. doi: 10.1109/7.8635290.

[35] X. Mei and H. Ling, “Robust visual tracking and vehicle classification via sparse representation,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 33, no. 11, pp. 2259–2272, 2011. doi: 10.1109/TPAMI.2011.66.

[36] N. Wang, J. Wang, and D. Yeung, “Online robust non-negative dictionary learning for visual tracking,” in *Proc. IEEE Int. Conf. Comput. Vision (ICCV)*, 2013, pp. 657–664.
[37] S. Avidan, "Support vector tracking," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 26, no. 8, pp. 1064–1072, 2004. doi: 10.1109/TPAMI.2004.53.

[38] D. Held, S. Thrun, and S. Sebastian, "Learning to track at 100 FPS with deep regression networks," in *Proc. European Conf. Comput. Vision (ECCV)*, 2016, pp. 749–765.

[39] J. F.Henriques, R. Caseiro, P. Martins, and J. Batista, "High-speed tracking with kernelized correlation filters," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 37, no. 3, pp. 583–596, 2015. doi: 10.1109/TPAMI.2014.2345390.

[40] B. D. Lucas and T. Kanade, "An iterative image registration technique with an application to stereo vision," in *Proc. Int. Joint Conf. Artif. Intell.* (IJCAI), 1981, pp. 1–9.

[41] R. D. DeGroat, "Noniterative subspace tracking," *IEEE Trans. Signal Process.*, vol. 40, no. 3, pp. 571–577, 1992. doi: 10.1109/78.120800.

[42] R. Badeau, B. David, and G. Richard, "Fast approximated power iteration subspace tracking," *IEEE Trans. Signal Process.*, vol. 53, no. 8, pp. 2931–2941, 2005. doi: 10.1109/TSP.2005.850378.

[43] X. Lan, A. J. Ma, P. C. Yuen, and R. Chellappa, "Joint sparse representation and robust feature-level fusion for multi-cue visual tracking," *IEEE Trans. Image Process.*, vol. 24, no. 12, pp. 5826–5841, 2015. doi: 10.1109/TIP.2015.2481325.

[44] T. Bai and Y. F. Li, "Robust visual tracking with structured sparse representation appearance model," *Pattern Recogn.*, vol. 45, no. 6, pp. 2390–2404, 2012. doi: 10.1016/j.patcog.2011.12.004.

[45] X. Jia, H. Lu, and M. Yang, "Visual tracking via adaptive structural local sparse appearance model," in *Proc. IEEE Conf. Comput. Vision Pattern Recogn.* (CVPR), 2012, pp. 1822–1829.

[46] J. Ning, J. Yang, S. Jiang, L. Zhang, and M.-H. Yang, "Object tracking via dual linear structured SVM and explicit feature map," in *Proc. IEEE Conf. Comput. Vision Pattern Recogn.* (CVPR), June 2016, pp. 4266–4274.

[47] S. Hare et al., "STRUCK: Structured output tracking with kernels," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 38, no. 10, pp. 2096–2109, 2016. doi: 10.1109/TPAMI.2015.2509974.

[48] Y. Bai and M. Tang, "Robust tracking via weakly supervised ranking SVM," in *Proc. IEEE Conf. Comput. Vision Pattern Recogn.* (CVPR), 2012, pp. 1854–1861.

[49] B. Babenko, M. Yang, and S. Belongie, "Visual tracking with online multiple instance learning," in *Proc. IEEE Conf. Comput. Vision Pattern Recogn.* (CVPR), 2009, pp. 983–990.

[50] B. Babenko, M.-H. Yang, and S. Belongie, "Robust object tracking with online multiple instance learning," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 33, no. 8, pp. 1619–1632, 2011. doi: 10.1109/TPAMI.2010.226.

[51] K. Zhang and H. Song, "Real-time visual tracking via online weighted multiple instance learning," *Pattern Recogn.*, vol. 46, no. 1, pp. 397–411, 2013. doi: 10.1016/j.patcog.2012.07.013.

[52] R. D. Fu, Changhong and E. Kayacan, "Visual tracking with online structural similarity-based weighted multiple instance learning," *Inf. Sci.*, vol. 481, pp. 292–310, May 2019. doi: 10.1016/j.ins.2018.12.080.

[53] K. Zhang, L. Zhang, and M.-H. Yang, "Real-time compressive tracking," in *Proc. European Conf. Comput. Vision (ECCV)*, 2012, pp. 864–877.

[54] P. Li, D. Wang, L. Wang, and H. Lu, "Deep visual tracking: Review and experimental comparison," *Pattern Recogn.*, vol. 76, pp. 323–338, Apr. 2018. doi: 10.1016/j.patcog.2017.11.007.

[55] B. Li, J. Yan, W. Wu, Z. Zhu, and X. Hu, "High performance visual tracking with Siamese region proposal network," in *Proc. IEEE Conf. Comput. Vision Pattern Recogn.* (CVPR), 2018, pp. 8971–8980.

[56] M. Danelljan, C. Bhat, F. S. Khan, and M. Felsberg, "ATOM: Accurate tracking by overlap maximization," in *Proc. IEEE Conf. Comput. Vision Pattern Recogn.* (CVPR), 2019, pp. 4660–4669.

[57] L. Bertinetto, J. Valmadre, J. F. Henriques, A. Vedaldi, and P. H. S. Torr, "Fully-convolutional Siamese networks for object tracking," in *Proc. European Conf. Comput. Vision (ECCV)*, 2016, pp. 850–865.

[58] Q. Guo, W. Feng, C. Zhou, R. Huang, L. Wan, and S. Wang, "Learning dynamic Siamese network for visual object tracking," in *Proc. IEEE Int. Conf. Comput. Vision (ICCV)*, 2017, pp. 1781–1789.

[59] B. Yan, H. Zhao, D. Wang, H. Lu, and X. Yang, "Skimming-perusal tracking: A framework for real-time and robust long-term tracking," in *Proc. IEEE Int. Conf. Comput. Vision (ICCV)*, 2019, pp. 2385–2393.

[60] Z. Teng, J. Xing, Q. Wang, B. Zhang, and J. Fan, "Deep spatial and temporal network for robust visual object tracking," *IEEE Trans. Image Process.*, vol. 29, pp. 1762–1775, 2020. doi: 10.1109/TIP.2019.2942502.

[61] R. Girshick, "Fast R-CNN," in *Proc. IEEE Conf. Comput. Vision Pattern Recogn.* (CVPR), 2015, pp. 1440–1448.

[62] B. Jiang, R. Luo, J. Mao, T. Xiao, and Y. Jiang, "Acquisition of localization confidence for accurate object detection," in *Proc. European Conf. Comput. Vision (ECCV)*, 2018, pp. 784–799.

[63] K. Dai, Y. Zhang, D. Wang, J. Li, H. Lu, and X. Yang, "High-performance long-term tracking with meta-updater," in *Proc. IEEE Conf. Comput. Vision Pattern Recogn.* (CVPR), 2020, pp. 6298–6307.

[64] Z. He, Y. Fan, J. Zhuang, Y. Dong, and H. Bai, "Correlation filters with weighted convolution responses," in *Proc. IEEE Conf. Comput. Vision Pattern Recogn.* (CVPRW), 2017, pp. 1992–2000.

[65] T. Zhang, C. Xu, and M.-H. Yang, "Multi-task correlation particle filter for robust object tracking," in *Proc. IEEE Conf. Comput. Vision Pattern Recogn.* (CVPR), 2017, pp. 4335–4343.

[66] F. Li, C. Tian, W. Zuo, L. Zhang, and M. Yang, "Learning spatial-temporal regularized correlation filters for visual tracking," in *Proc. IEEE Conf. Comput. Vision Pattern Recogn.* (CVPR), 2018, pp. 4904–4913.

[67] C. Ma, J. Huang, X. Yang, and M. Yang, "Hierarchical convolutional features for visual tracking," in *Proc. Int. Conf. Comput. Vision (ICCV)*, 2015, pp. 3074–3082.

[68] Y. Sun, C. Sun, D. Wang, Y. He, and H. Lu, "ROI pooled correlation filters for visual tracking," in *Proc. IEEE Conf. Comput. Vision Pattern Recogn.* (CVPR), June 2019, pp. 5783–5791.

[69] B. Yan, D. Wang, H. Lu, and X. Yang, "Cooling-shrinking attack: Blinding the tracker with imperceptible noises," in *Proc. IEEE Conf. Comput. Vision Pattern Recogn.* (CVPR), 2020, pp. 990–999.

[70] D. Zhao, L. Xiao, H. Fu, T. Wu, X. Xu, and B. Dai, "Augmenting cascaded correlation filters with spatial–temporal saliency for..."
visual tracking,” *Inf. Sci.*, vol. 470, pp. 78–93, Jan. 2019. doi: 10.1016/j.ins.2018.08.053.

[71] M. Danelljan, G. Häger, F. S. Khan, and M. Felsberg, “Learning spatially regularized correlation filters for visual tracking,” in *Proc. IEEE Int. Conf. Comput. Vision (ICCV)*, Dec. 2015, pp. 4310–4318.

[72] M. Danelljan, A. Robinson, F. S. Khan, and M. Felsberg, “Beyond correlation filters: Learning continuous convolution operators for visual tracking,” in *Proc. European Conf. Comput. Vision (ECCV)*, 2016, pp. 472–488.

[73] K. Dai, D. Wang, H. Lu, C. Sun, and J. Li, “Visual tracking via adaptive spatially-regularized correlation filters,” in *Proc. IEEE Conf. Comput. Vision Pattern Recogn. (CVPR)*, 2019, pp. 4665–4674.

[74] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *Proc. IEEE Conf. Comput. Vision Pattern Recogn. (CVPR)*, 2005, vol. 1, pp. 886–893.

[75] P. F. Felzenszwalb, R. B. Girshick, D. McAllester, and D. Ramanan, “Object detection with discriminatively trained part-based models,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 32, no. 9, pp. 1627–1645, 2010. doi: 10.1109/TPAMI.2009.167.

[76] J. van de Weijer and C. Schmid, “Coloring local feature extraction,” in *Proc. European Conf. Comput. Vision (ECCV)*, 2006, pp. 334–348.

[77] K. Simonyan and A. Zisserman, “Very Deep Convolutional Networks for Large-Scale Image Recognition,” in *Proc. Int. Conf. Learning Representations (ICLR)*, 2014, pp. 1–21.

[78] D. S. Bolme, J. R. Beveridge, B. A. Draper, and Y. M. Lui, “Visual object tracking using adaptive correlation filters,” in *Proc. IEEE Conf. Comput. Vision Pattern Recogn. (CVPR)*, 2010, pp. 2544–2550.

[79] M. Danelljan, G. Häger, F. S. Khan, and M. Felsberg, “Discriminative scale space tracking,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 39, no. 8, pp. 1561–1575, 2017. doi: 10.1109/TPAMI.2016.2609928.

[80] H. K. Galoogahi, A. Fagg, and S. Lucey, “Learning background-aware correlation filters for visual tracking,” in *Proc. IEEE Int. Conf. Comput. Vision (ICCV)*, 2017, pp. 1144–1152.

[81] C. Fu, Y. He, F. Lin, and W. Xiong, “Robust multi-kernelized correlators for UAV tracking with adaptive context analysis and dynamic weighted filters,” *Neural Comput. Appl.*, vol. 32, no. 16, pp. 12,591–12,607, 2020. doi: 10.1007/s00521-020-04716-x.

[82] F. Li, C. Fu, F. Lin, Y. Li, and P. Lu, “Training-set distillation for real-time UAV object tracking,” in *Proc. Int. Conf. Robot. Automat.* (ICRA), 2020, pp. 9715–9721.

[83] Y. Li and J. Zhu, “A scale adaptive kernel correlation filter tracker with feature integration,” in *Proc. European Conf. Comput. Vision (ECCV)*, 2014, pp. 254–265.

[84] M. Danelljan, G. Häger, F. Khan, and M. Felsberg, “Accurate scale estimation for robust visual tracking,” in *Proc. British Machine Vision Conf. (BMVC)*, 2014, pp. 1–14.

[85] C. Wang, L. Zhang, L. Xie, and J. Yuan, “Kernel cross-correlator,” in *Proc. AAAI Conf. Artif. Intell.* (AAAI), 2018, pp. 4179–4186.

[86] J. F. Henriques, R. Caseiro, P. Martins, and J. Batista, “Exploiting the circulant structure of tracking-by-detection with kernels,” in *Proc. European Conf. Comput. Vision (ECCV)*, 2012, pp. 702–715.

[87] M. Danelljan, F. Shahbaz Khan, M. Felsberg, and J. Van de Weijer, “Adaptive color attributes for real-time visual tracking,” in *Proc. IEEE Conf. Comput. Vision Pattern Recogn. (CVPR)*, 2014, pp. 1090–1097.

[88] L. Bertinetto, J. Valmadre, S. Golodetz, O. Miksik, and P. H. Torr, “Staple: Complementary learners for real-time tracking,” in *Proc. IEEE Conf. Comput. Vision Pattern Recogn. (CVPR)*, 2016, pp. 1401–1409.

[89] N. Wang, W. Zhou, Q. Tian, R. Hong, M. Wang, and H. Li, “Multi-cue correlation filters for robust visual tracking,” in *Proc. IEEE Conf. Comput. Vision Pattern Recogn. (CVPR)*, 2018, pp. 4844–4853.

[90] M. Danelljan, G. Häger, F. Shahbaz Khan, and M. Felsberg, “Adaptive decontamination of the training set: A unified formulation for discriminative visual tracking,” in *Proc. IEEE Conf. Comput. Vision Pattern Recogn. (CVPR)*, 2016, pp. 1430–1438.

[91] A. Lukezic, T. Vojir, L. Cehovin Zajc, J. Matas, and M. Kristan, “Discriminative correlation filter with channel and spatial reliability,” in *Proc. IEEE Conf. Comput. Vision Pattern Recogn. (CVPR)*, 2017, pp. 6309–6318.

[92] H. Kiani Galoogahi, A. Fagg, and S. Lucey, “Learning background-aware correlation filters for visual tracking,” in *Proc. Int. Conf. Comput. Vision (ICCV)*, 2017, pp. 1135–1143.

[93] C. Ma, X. Yang, C. Zhang, and M.-H. Yang, “Long-term correlation tracking,” in *Proc. IEEE Conf. Comput. Vision Pattern Recogn. (CVPR)*, 2015, pp. 5388–5396.

[94] C. Ma, J.-B. Huang, X. Yang, and M.-H. Yang, “Adaptive correlation filters with long-term and short-term memory for object tracking,” *Int. J. Comput. Vis.*, vol. 126, no. 8, pp. 771–796, 2018. doi: 10.1007/s11263-018-1076-4.

[95] M. Danelljan, G. Bhat, F. Shahbaz Khan, and M. Felsberg, “ECO: Efficient convolution operators for tracking,” in *Proc. IEEE Int. Conf. Comput. Vision and Pattern Recogn. (CVPR)*, 2017, pp. 6638–6646.

[96] M. Mueller, N. Smith, and B. Ghanem, “Context-aware correlation filter tracking,” in *Proc. IEEE Conf. Comput. Vision Pattern Recogn. (CVPR)*, 2017, pp. 1396–1404.

[97] Z. Kalal, K. Mikolajczyk, and J. Matas, “Tracking-learning-detection,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 34, no. 7, pp. 1409–1422, 2012. doi: 10.1109/TPAMI.2011.239.

[98] N. Wang, Y. Song, C. Ma, W. Zhou, W. Liu, and H. Li, “Unsupervised deep tracking,” in *Proc. IEEE Conf. Comput. Vision Pattern Recogn. (CVPR)*, 2019, pp. 1308–1317.

[99] J. Valmadre, L. Bertinetto, J. Henriques, A. Vedaldi, and P. H. Torr, “End-to-end representation learning for correlation filter-based tracking,” in *Proc. IEEE Conf. Comput. Vision Pattern Recogn. (CVPR)*, 2017, pp. 2805–2813.

[100] X. Li, C. Ma, B. Wu, Z. He, and M.-H. Yang, “Target-aware deep tracking,” in *Proc. IEEE Conf. Comput. Vision Pattern Recogn. (CVPR)*, 2019, pp. 1369–1378.

[101] L. Zhang and P. N. Suganthan, “Robust visual tracking via co-trained kernelized correlation filters,” *Pattern Recogn.*, vol. 69, pp. 82–93, Sept. 2017. doi: 10.1016/j.patcog.2017.04.004.

[102] F. Li, Y. Yao, P. Li, D. Zhang, W. Zuo, and M.-H. Yang, “Integrating boundary and center correlation filters for visual tracking with aspect ratio variation,” in *Proc. Int. Conf. Comput. Vision Workshops (ICCVW)*, 2017, pp. 2001–2009.