A review of Discriminative Correlation Filter methods for object detection from low-altitude UAVs

DRAI Gabriel
Centrale Supélec
3 rue Joliot Curie

SIDBON Shawn
Centrale Supélec
3 rue Joliot Curie

Abstract

The task of object tracking through the use of correlation filters has recently gained a lot of traction as it is the fastest option and therefore is the perfect fit for online tasks. In this paper, we have re-implemented 3 of the core algorithms, namely Minimum Output Sum of Squared Error (MOSSE) [2], Kernelized Correlation Filters (KCF) and Dual Correlation Filters [8] using raw features (grayscale or RGB) and HOG features on the task of tracking from the point of view of a UAV using the UAV123 dataset. While the KCF and DCF have proven much more performant than the MOSSE algorithm with MIOU scores nearly 50% higher, the use of HOG features seemed to be detrimental in the case of the KCF which goes against what was demonstrated in the paper from Henriques et al. (2015) [8]. These 3 algorithms form the basis of their more advanced counterparts and suffer from drawbacks from variations in the frames of the videos used. The most prominent difficulty is that of tracking in the context of Full Occlusion. The task was all the more difficult since we used a 10fps version of the dataset making variations between frames all the more important.

1. Introduction

1.1. Object Tracking

Object tracking is an application of computer vision where the goal is to determine the position of an object at a particular frame, it is different from object detection in the sense that a Region Of Interest (ROI) is given for the first frame of the video. It has been extensively used in surveillance, security, traffic monitoring, anomaly detection, robot vision, and visual tracking. The two main subsets of object tracking are: Single Object Tracking (SOT) and Multiple Object Tracking (MOT) which is far more difficult and for which SOT techniques are not sufficient. In this project we will focus primarily on the former, namely Single Object Tracking.

The task of object tracking can be further divided into subcategories, Discriminative Filters and tracking by detection models. The former focuses on the target and its surrounding environment while the second works in a sliding window manner to detect whether or not an object is in a given window. This is usually much more time consuming than its Discriminative counterparts who have proven to reach speeds of 600 fps [2] on ordinary machines. Discriminative models are therefore extremely convenient for online applications and still provide state-of-the-art performance (SOTA). Milestone algorithms include Minimum Output Sum of Squared Error (MOSSE) [2], Kernelized Correlation Filters (KCF) [8], Spatially Regularized Discriminative Correlation Filters (SRDCF) [6], Discriminative Correlation Filter with Channel and Spatial Reliability (CSRDCF) [10] and more.

1.2. Key Challenges in Object Tracking

As mentioned, a main point of concern in object tracking is that of speed. Tracking algorithms are expected to detect and localize the object in a video in a fraction of a second and with high accuracy. The detection speed highly depends on the size of the ROI and hence, although a bigger search area might lead to better performance it does come at an increase computational expense.

The second main concern is that of the accuracy of the tracking which can be greatly affected by numerous variations that occur throughout a video recording. Indeed, for instance, an object can be present in an image (or a video) and in various sizes and orientations. this spatial difference can greatly hinder the performance when tracking. Another major issue is that of occlusion. Occlusion is when multiple objects come so close together that they appear to be merged. This can confuse the algorithm into thinking the merged object is a single object or simply wrongly identifying the object.

Aside from these, numerous other variations make tracking a complicated task, such as switching of identity after crossing (e.g. one bicycle passes in-front of the bicycle we want to track), motion blur, variation in the viewpoint, cluttering of similar objects in the background, low resolution, and variation in the illumination.

1.3. The Problem

The general problem that we try to solve with correlation filter is two-fold : the first part is that of training during which we have the ground truth for the first frame and hence through iterations or circulant matrices [8] we attempt to learn an optimal set of parameters by minimizing an objective function. For the purpose of this paper we will mainly be using the loss associated to a Ridge Regression [9].

$$\min_{\mathbf{W}} \sum_{i} (f(\mathbf{X_i}) - Y_i)^2 + \lambda |\mathbf{W}|_2 \tag{1}$$

Where f is a function that approximates the ground truth and that maps an input from $\mathbb{R}^n xm$ to $\mathbb{R}^n xm$, $f(\mathbf{x_i}) = \mathbf{W}^\intercal X_i$ and Y_i is the ground truth. This formulation is very general but is the main idea behind all of the algorithms mentioned. There are some variations and additional concepts to take into account when for instance, we attempt to adapt the scale - which will not be studied in this paper.

As the name suggests, Discriminative Correlation Filters attempt to find the optimal filter to determine the changes in position of an object. An optimal correlation filter when multiplied with the current frame will assign high values to what it considers to be the most probable position for the center of the tracked object.

Additionally, we should note that although the ridge regression admits a closed form solution, it is quite difficult to calculate. Similarly correlations are computationally expensive. Hence, in the algorithms that we will present, the mathematical tool used to overcome these issues is that of moving to the Fourier domain through a Discrete Fourier Transform as a correlation in the spatial domain is equivalent to a element-wise product in the frequency domain.

Hence the objective in correlation filter-based tracking is that of finding the optimal filter such that:

$$\hat{\mathbf{y}} = \hat{\mathbf{x}} \odot \hat{\mathbf{w}}^* \tag{2}$$

where '$\hat{}$' symbolizes a Discrete Fourier Transform (DFT) and '$*$' refers to the complex conjugate.

The second problem is that of updating the position of the ROI at each frame as well as updating our learned parameters with the latest observations. This can be done through a weighing of past observations and will be discussed in more detail in following sections.

2. Literature Review

Research into the topic of correlation filters started in the early 1980's through Synthetic Discriminant Functions [13], Minimum Average Correlation Energy [11] and many others. It has nonetheless only gained popularity since 2010 with Bolme et al's (2010) MOSSE tracker [2], which used a grayscale variant of the image to achieve state of the art performance at an incredible processing speed. Numerous efforts were made to take advantage of non-linearity's and the use of more expressive features to improve the performance while still keeping the great advantages of speed. the DCF and KCF proposed by Henriques et al. (2015) [8] relied no longer on grayscale images but HOG features as well as circulant kernels which greatly simplified the computational cost while allowing for more expressive features and functions. HOG features [4] proved to be a great addition and did not come at an increased computational cost although it greatly increased the number of features.

At the same time some effort was spent on finding solutions to objects that varied in scale. The idea of Discriminative Scale Space Tracking (DSST) [5] was hence put forward and achieved state of the art perfor-

mance and led to the development of more complex algorithms. Following these efforts, much of the work was focused on introducing the concepts of spatial and temporal dependence. which culminated with Channel and Spatial Reliability Discriminative Correlation Filter (CSR-DCF) [10] and reached state of the art performance. The aforementioned algorithm through the introduction of spatial reliability is able to adjust itself to the part of the object that is most suitable for tracking which enables larger regions to be searched as well as the tracking of non rectangular objects and achieved state of the art performance on VOT 2015, VOT 2016 and OTB100. [14]

3. Methodology

3 of the algorithms, namely MOSSE [2], DCF and KCF [8], were re-implemented in python, hence, it is important to note that some performance issues, notably speed, might come up as the original versions were implemented in matlab or C. The algorithms will be measured in terms of Intersection over Union and according to the various challenges that each video recording depicts, e.g. occlusion, illumination variation, blur, cluttered background etc. This will give us a better understanding on the nature of their absolute and relative performance.

$$IoU = \frac{A \cap B}{A \cup B} \tag{3}$$

where A represent the ground truth of the region of interest and B represents the predicted region of interest. As we will have no variation in scale in our prediction i.e. our last prediction will have the same area as our first, we will not be looking at precision.

Furthermore the performance will be tested on the UAV123 dataset [12] which comprises various videos taken from drones, the first frame of the image has a bounding box made available for us to use and the remaining frames must be predicted.

3.1. MOSSE

Using the notations defined above - which are different from the original paper [2] for simplicity and readability purposes - let's note our ground truth for frame i, $\mathbf{Y_i}$ and its Fourier transform $\hat{\mathbf{y}}_\mathbf{i}$, similarly let's note the $\mathbf{x_i}$ the preprocessed frame cropped around the ROI and $\hat{\mathbf{x}}_\mathbf{i}$ its Fourier transform.

Moving to the Fourier domain, the minimization objective becomes:

$$\min_{\hat{\mathbf{W}}^*} \sum_i |\hat{\mathbf{X}}_\mathbf{i} \odot \hat{\mathbf{W}}^* - \hat{\mathbf{Y}}_\mathbf{i}|^2 \tag{4}$$

As mentioned previously, passing to the Fourier domain greatly simplifies our calculations as the filter can be calculated as the element wise division of the frame by the ground truth which is in fact a Gaussian with a peak centered on the target. Through a partial derivation with respect to every element in $\hat{\mathbf{W}}^*$ it is easy to find a closed form solution to the optimization task.

$$\hat{\mathbf{W}}^* = \frac{\sum_i \hat{\mathbf{Y}}_\mathbf{i} \odot \hat{\mathbf{X}}_\mathbf{i}^*}{\sum_i \hat{\mathbf{X}}_\mathbf{i} \odot \hat{\mathbf{X}}_\mathbf{i}^*} \tag{5}$$

For practical purposes, Y_i is a Gaussian response map with the highest values centered and lower values as you get closer to the borders. Furthermore, some preprocessing is done on $\mathbf{X_i}$ such as a log transform, a standardization as well as the application of a cosine window to put more emphasise on the center of the frame.

Following the initial training phase of the algorithm mentioned above, the filter needs to be updated after each prediction and this is done through the addition of a weight decay on earlier observations, hence giving more importance to recent frames. The update process of the filter after frame i goes as follows:

$$\hat{\mathbf{W}}_\mathbf{i}^* = \frac{\mathbf{A_i}}{\mathbf{B_i}} \tag{6}$$

where

$$\mathbf{A_i} = \eta \hat{\mathbf{Y}}_\mathbf{i} \odot \hat{\mathbf{X}}_\mathbf{i}^* + (1-\eta)\mathbf{A_{i-1}} \tag{7}$$

$$\mathbf{B_i} = \eta \hat{\mathbf{X}}_\mathbf{i} \odot \hat{\mathbf{X}}_\mathbf{i}^* + (1-\eta)\mathbf{B_{i-1}} \tag{8}$$

and η is the interpolation rate (or adaptation rate) that the algorithm uses for the weighing during the tracking phase to update the filter according to the new observations.

A quick note on the parameters used for the MOSSE algorithm:

MOSSE parameters	
Parameter	value
Gaussian response map σ	2
Adaptation rate η	1e-2
Affine Transformations	100

3.2. DCF and KCF

DCF and KCF [8], come with the additional notion of cyclic shifts and circulant matrices which enables the training of a model taking into account the base case i.e. positive sample and various translations of it i.e. negative samples. A circulant matrix can be seen as follows for the case of a 1 dimensional vector and is extendable to the multi-dimensional case.

$$C(\mathbf{x}) = \begin{pmatrix} x_1 & x_2 & ... & x_n \\ x_n & x_1 & ... & x_{n-1} \\ . & . & ... & . \\ . & . & ... & . \\ . & . & ... & . \\ x_2 & x_3 & ... & x_1 \end{pmatrix} \tag{9}$$

Among the many interesting properties that circulant matrices have, the most important is perhaps the notion that all circulant matrices are made diagonal by the discrete Forurier transform which greatly simplifies calculations and makes the computational complexity of tracking much more manageable.

For a give vector $\mathbf{x}$, we can hence write the circulant matrix of its DFT as:

$$\mathbf{X} = \mathbf{F} diag(\hat{\mathbf{x}}) \mathbf{F^H} \tag{10}$$

where $\mathbf{F^H}$ is the hermitian transpose of $\mathbf{F}$. and hence

$$\mathbf{X^H X} = \mathbf{F} diag(\hat{\mathbf{x}}^* \odot \hat{\mathbf{x}}) \mathbf{F^H} \tag{11}$$

Once again, in this particular case we are trying to minimize the ridge regression loss function. however, the main difference is that the above equation was viewed as a linear relationship between the filter and the frame in the Fourier domain, in our particular case we are using kernel functions as approximators which allow for non-linearities without the added computational cost.

$$\hat{\mathbf{w}} = \frac{\hat{\mathbf{y}} \odot \hat{\mathbf{x}}^*}{\hat{\mathbf{x}} \odot \hat{\mathbf{x}}^* + \lambda} \tag{12}$$

To note that this does not change if we are working with matrices and the equation becomes similar to what we have in $\mathbf{eq.5}$.

In this particular case we will focus on the linear (DCF) and Gaussian (KCF) kernels as both allow the circulant matrix to be diagonalizable in the fourier domain and allow for a fast solution to the ridge regression problem. Moreover other kernels (e.g. the polynomial kernel) have proven to be more complex and providing similar results.

$$\hat{\alpha} = \frac{\hat{\mathbf{y}}}{k^{\hat{\mathbf{xx}}} + \lambda} \tag{13}$$

To note that the alternative would be to solve the equation below which calls for an matrix inversion and is hence polynomial in complexity

$$\alpha = (K\lambda I)^{-1} y \tag{14}$$

The linear kernel admits a simple solution and we can note that the added expense of having additional channels (e.g. 37 for HOG features with 11 orientations) is low as we simply sum over the channels.

$$k^{xx'} = \mathfrak{F}^{-1}(\sum_c \hat{x_c^*} \odot \hat{x'_c}) \tag{15}$$

The Gaussian (or Radial Basis) kernel admits a slightly more complicated solution but shares the same benefits of the linear kernel, that is to say, the term on the right sums over all channels.

$$k^{xx'} = \exp \frac{-1}{\sigma^2}(|x|^2 + |x'|^2 + \mathfrak{F}^{-1}(\sum_c \hat{x_c^*} \odot \hat{x'_c})) \tag{16}$$

In their paper, Henriques et al, prove that although using HOG features comes at the cost of feature extraction from the ROI, this is compensated by the reduced dimensions (height and width) driven by the cell size chosen for the hog feature extraction.

The parameters that were used for both these implementations are available below.

Raw parameters	
Parameter	value
Gaussian response map σ_1	6e-2 $\sqrt{h * w}$
Adaptation rate η	1.5e-1
Padding	3
Gaussian Kernel sigma σ_2	0.2
HOG parameters	
Gaussian response map σ_1	$6e-2\sqrt{h/4 * w/4}$
Adaptation rate η	8e-2
Padding	3
Gaussian Kernel sigma σ_2	0.5
cell size	4
orientations	11

3.3. UAV-123

The UAV 123 dataset [12] is composed of drone footage of either aerial, land-based or water-based objects. It contains a total of 123 videos with varying characteristics to test the performance of trackers in various scenarios. It is furthermore important to note that we are working with the UAV123 10fps dataset which presents the added difficulty of only having 10 frames per second and hence a bigger delta between each frame. The performance is therefore obviously lower than the original dataset with 30fps.

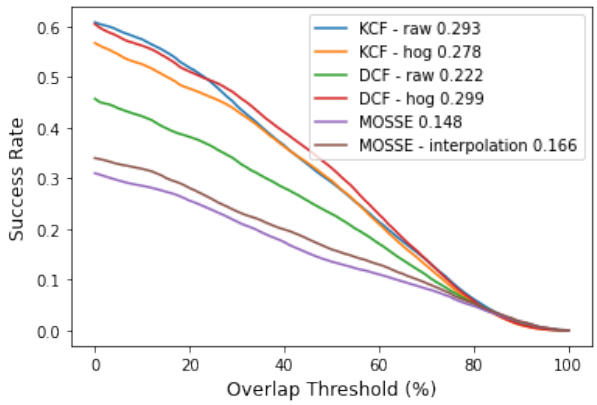
UAV123 dataset attributes	
Attributes	Sequences
Aspect Ratio Change	65
Background Clutter	21
Camera Motion	64
Fast Motion	22
Full Occlusion	33
Illumination Variation	25
Low Resolution	48
Out of View	28
Partial Occlusion	68
Similar Object	39
Scale Variation	103
Viewpoint Change	55

4. Results

Before diving into the various results it is important to note that the algorithms were implemented on python and hence do not have the benefits of C in terms of processing speed, it is therefore normal that our implementations do not share the speed that is depicted in the papers. Some improvement can certainly be made, however, our main objective was the understanding and re-implementation of key concepts in the space of visual tracking and not the improvement of already existing implementations.

As mentioned earlier we have re-implemented MOSSE, KCF and DCF, they do not perform at SOTA as currently the best methods combine the idea of Kernelized correlation filters and deep learning concepts. to which we have added the notion of both interpolation (i.e. merging frames) and the idea of scale adaptation. The results obtained are OPE meaning that only the first bbox was made available to the algorithm and the rest had to be predicted without peaking at previous ground truths. Furthermore we have separated the results into the attributes mentioned earlier.

Figure 1. OPE - success plot



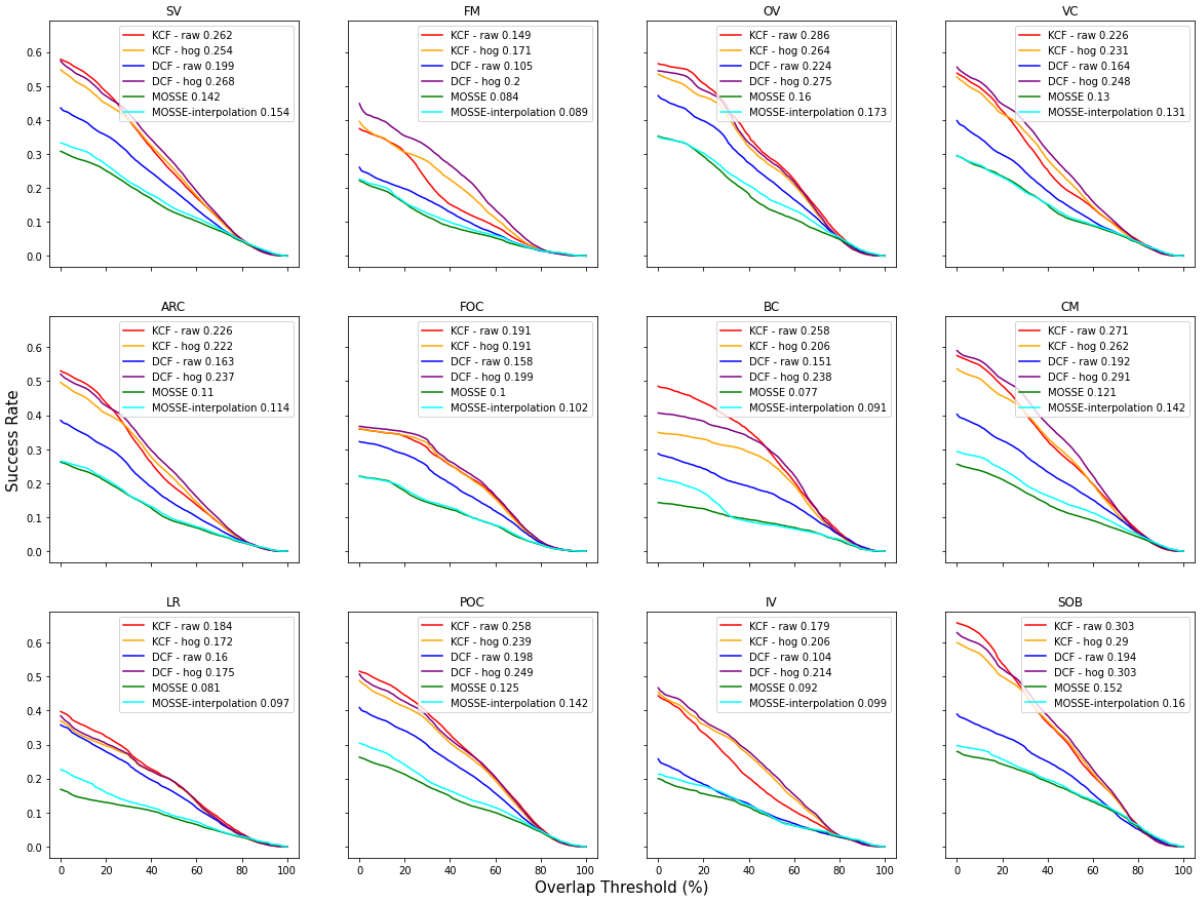
Success plot of MOSS (purple, brown), KCF (blue, orange) and DCF (green, red) and some variations on the features used and the use of merged frames

Fig.1, DCF with hog features and KCF without hog features greatly outperform the other algorithms with a Mean Intersection Over Union close to 0.3. It is important to mention that our results do not corroborate with the original paper by Henrique et al [8] as they obtain a much better performance when using HOG features as opposed to the raw (i.e. RGB) features. It also seems that merging frames together improves the results at the expense of high computational cost. If the merging process uses 4 intermediate steps, then the computational complexity is at least 4 times higher as we have to update the algorithm for each intermediate step.

When it comes to the speed of the trackers, MOSSE is obviously the fastest and the DCF with raw features seem to be even faster. The hog feature version might be slower due to the implementation used to extract the hog features. The hardware used to run the following was a first generation mac book air with M1 chip.

Tracking algorithm - Speed	
Algorithm	FPS
MOSSE	102
MOSSE - interpolation	22
DCF - raw	55
DCF - hog	41
KCF - raw	54
KCF - hog	40

Figure 2. OPE - success plot per attribute



Success plot of MOSS (green, cyan), KCF (red, orange) and DCF (blue, purple) and some variations on the features used and the use of merged frames for various attributes

Fig.2, Depicts the areas of weakness and strength of the 3 implemented algorithms. The obvious weakness - across the board- is that of full occlusion. The algorithms implemented are not robust to these types of variations in the environment. Nonetheless, although HOG features do not lead to as big a jump in performance as expected, they still enhance greatly the performances in areas where HOG features are well known to provide robustness namely Illumination variation (IV) and Fast Motion (FM). Similar Objects (SOB) is the easiest variation for all of the trackers. Using different color scales might help in certain aspects such as illumination variation (using HSV for instance) but were not implemented in this study.

5. Discussion

As mentioned previously, a few remarks must be made, (1) as the implementation was done in python there is a clear discrepancy when it comes to the performance of such models. (2) Furthermore, the overall performance of our models is questionable especially in the case of HOG features where the models are not performing as they should according to the original paper [8].

(3) Newer implementations such as CSRDCF and SRDCF which are not based on deep learning approaches outperform the models implemented, however the heavily build upon the same notions and to understand them it is necessary to understand the ones we have implemented. Hence, re-scaling the bounding box, as well as introducing the concepts of spatial and temporal reliability map would provide significant improvements on our current results. (4) Finally, the use of Deep Learning models was not considered in our work, however, it is important to note that models SOTA is performed my models that are built on some concepts such as Siamese Networks [3] [1], Recurrent Networks [7] and Reinforcement Learning.

References

[1] Luca Bertinetto, Jack Valmadre, João F. Henriques, Andrea Vedaldi, and Philip H. S. Torr. Fully-convolutional siamese networks for object tracking. *CoRR*, abs/1606.09549, 2016. 8

[2] David S. Bolme, J. Ross Beveridge, Bruce A. Draper, and Yui Man Lui. Visual object tracking using adaptive correlation filters. pages 2544–2550, 2010. 1, 2, 3

[3] Miaobin Cen and Cheolkon Jung. Fully convolutional siamese fusion networks for object tracking. In *2018 25th IEEE International Conference on Image Processing (ICIP)*, pages 3718–3722, 2018. 8

[4] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, volume 1, pages 886–893 vol. 1, 2005. 2

[5] Martin Danelljan, Gustav Häger andFahad Shahbaz Khan, and Michael Felsberg. Discriminative scale space tracking. *IEEE Trans. Pattern Anal. Mach. Intell.*, 39(8):1561–1575, 2017. 2

[6] Martin Danelljan, Gustav Häger, Fahad Shahbaz Khan, and Michael Felsberg. Learning spatially regularized correlation filters for visual tracking. In *2015 IEEE International Conference on Computer Vision (ICCV)*, pages 4310–4318, 2015. 1

[7] Daniel Gordon, Ali Farhadi, and Dieter Fox. Re^3: Real-time recurrent regression networks for visual tracking of generic objects. *IEEE Robotics and Automation Letters*, 3(2):788–795, 2018. 8

[8] João F. Henriques, Rui Caseiro, Pedro Martins, and Jorge Batista. High-speed tracking with kernelized correlation filters. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 37(3):583–596, 2015. 1, 2, 3, 4, 5, 8

[9] Arthur E. Hoerl and Robert W. Kennard. Ridge regression: Biased estimation for nonorthogonal problems. *Technometrics*, 12(1):55–67, 1970. 2

[10] Alan Lukežič, Tomáš Vojíř, Luka Čehovin Zajc, Jiří Matas, and Matej Kristan. Discriminative correlation filter tracker with channel and spatial reliability. *International Journal of Computer Vision*, 126(7):671–688, 2018. 1, 3

[11] Abhijit Mahalanobis, B. V. K. Vijaya Kumar, and David Casasent. Minimum average correlation energy filters. *Appl. Opt.*, 26(17):3633–3640, Sep 1987. 2

[12] Matthias Mueller, Neil Smith, and Bernard Ghanem. *A Benchmark and Simulator for UAV Tracking*, pages 445–461. Springer International Publishing, Cham, 2016. 3, 5

[13] B.V.K. Vijaya Kumar, J.D. Brasher, C.F. Hester, G. Srinivasan, and S. Bollapragada. Synthetic discriminant functions for recognition of images on the boundary of the convex hull of the training set. *Pattern Recognition*, 27(4):543–548, 1994. 2

[14] Yi Wu, Jongwoo Lim, and Ming-Hsuan Yang. Object tracking benchmark. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 37(9):1834–1848, 2015. 3