Robust Tracking of Vessels in Oceanographic Airborne Images

Jorge Matos
ISR/IST, Universidade de Lisboa
Av. Rovisco Pais 1, 1049-001 Lisboa
Email: jorge.p.s.matos@tecnico.ulisboa.pt

Alexandre Bernardino
ISR/IST, Universidade de Lisboa
Av. Rovisco Pais 1, 1049-001 Lisboa
Email: alex@isr.tecnico.ulisboa.pt

Ricardo Ribeiro
ISR/IST, Universidade de Lisboa
Av. Rovisco Pais 1, 1049-001 Lisboa
Email: ribeiro@isr.tecnico.ulisboa.pt

***Abstract*—In this paper we present and evaluate an algorithm for tracking vessels in oceanographic airborne image sequences on the visible spectrum. Such sequences are challenging due to sun reflections, wakes, wave crests and fast motions, which significantly degrade the performance of general purpose tracking algorithms. The proposed method is based on state-of-the-art correlation filter tracking complemented with an image segmentation and blob analysis stage. The purpose of this later stage is to re-center the target in the tracking window to compensate for drifts in the correlation filter. We evaluate our proposal using a known benchmark in the field and compare it with general purpose tracking algorithms. Results show that our method beats the general purpose state-of-the-art tracking algorithms in the airborne maritime scenario both in performance and in computation time.**

I. Introduction

The SEAGULL project [1] developed an intelligent maritime surveillance system using unmanned autonomous vehicles (UAVs) equipped with various types of optical sensors (visible, infrared, multi- and hyper- spectral). This system is particularly interesting because it is affordable, easy to deploy and with few infrastructure requirements, in contrast to other systems used today. A fleet of fixed wing UAV's are equipped with computers running vision algorithms for the automatic detection of maritime vessels, whose coordinates are communicated to a coastal ground station via a radio link. The developed algorithms work in real time on the embedded hardware and present a low rate of false detections [2]. Nevertheless, the developed methods have difficulties mainly in the presence of sun reflections, breaking waves and boat wakes. The focus of our work is the development of a tracking algorithm more robust to these effects and to perform comparisons with other state-of-the-art tracking algorithms in the maritime scenario.

The main contribution of the present paper is the development of a new approach to maritime vessel tracking, combining an adaptive correlation filter [3] with a local re-detection step consisting of blob analysis for the correction of tracking offsets. The main idea consists in applying a detection step on the region of interest (ROI) to correct the target center when the conditions are favorable, i.e. when the boat is outside sun glare and boat wake regions (low background clutter). These conditions are detected at the blob analysis phase, using a few heuristics applied to the number, area and location of segmented blobs. Using this approach, it is possible to maintain the robustness of the correlation filter tracking, allowing it to keep the track during longer time spans. We compare the proposed approach with several state-of-the-art object tracking algorithms, using a data set of video sequences acquired during the SEAGULL project [1]. This data set is composed by thousands of annotated frames and a public release is under preparation to allow further research in this area.

The paper is organized as follows. In the next section, Sec. II, we present the current state-of-the-art in general purpose tracking algorithms and a few existing applications to maritime scenarios. Then, in Sec. III we present the proposed approach and its main components: the correlation filter, the features used, and the blob analysis step. In Sec. IV we describe the used datasets, the evaluation methods, and the experiments done to assess the performance of our methods in comparison to the state-of-the-art. Results are presented in Sec. V, illustrating the advantages of our methods both in tracking performance and computation time. Finally, Sec. VI summarizes the main conclusions of this work and refers to directions for future research.

II. State-of-the-art

Object tracking is one the most important and difficult tasks in computer vision. According to the Object Tracking Benchmark (OTB) [4], several aspects make visual tracking a very challenging task: illumination variation, scale variation, occlusion, deformation, motion blur, fast motion, in-plane rotation, out-of-plane rotation, out-of-view, background clutters, and low resolution. Annually, the state-of-the-art tracking algorithms are presented at the Visual Object Tracking (VOT) challenge [5]. In the 2015 competition, some of the top performers were based on correlation filter tracking using different kinds of approaches. Usually, correlation filters are designed to produce correlation peaks in the estimated target location while having low responses in the background. Although effective, the training of these methods was impractical for online tracking until the proposal of the Minimum Output Sum of Squared Error (MOSSE) filter [6]. Later developments extended MOSSE in a number of ways, such as the introduction of kernel methods by the Kernelized Correlation Filter (KCF) [3] tracker, and the Discriminative Scale Space

Tracker (DSST) [7]. Both of these methods used multiple channel HoG features [8] instead of the raw pixels used by MOSSE. Later, MUlti-Store Tracker (MUSTer) [9] proposed an approach using short-term and long-term tracking methods, working in a complementary manner. The correlation filter tracker (CFT) works as the short-term tracker while the long-term part is based on key-points and is used to locate the target when the CFT fails.

In the latest VOT competition, the top correlation filter trackers were the Spatially Regularized Correlation Filter (SRDCF) [10] and the DeepSRDCF [11], both from the same authors. They introduced a spatial regularization component in the optimization problem to penalize the correlation filter coefficients farther from the target. Recently, features based on convolutional neural networks (CNNs) [11] [12] have shown state-of-the-art results in various visual recognition tasks including visual tracking. The DeepSRDCF introduces the use of convolutional neural network multi-channel features to discriminate the target with the VGG-2048 [13] neural network used for image classification. More recently, [12] used multiple layers of a CNN to train multiple correlation filters. The idea behind this approach is that early layers of CNNs have higher spatial resolution allowing for precise localization while the features from deeper layers capture more semantic information and are robust to significant appearance changes.

In [14], it is proposed the DLT tracker. A stacked denoising autoencoder is trained offline to learn robust generic image features. Online tracking is made using the encoder trained from the previous autoencoder as a feature extractor and an additional classification layer. One interesting finding is that the filters in the first layer of the trained feature extractor resemble the Gabor filters for edge detection.

In [15] the authors propose the Multi-Domain Convolutional Neural Network (MDNet) tracker, winner of the 2015 VOT challenge. In this method, a CNN is pre-trained using a training set of tracking videos. The main aspect of this CNN is that the last layer is reinitialized at the beginning of each new video (domain-specific) while the deeper layers are updated online.

Other methods also present good results in many benchmarking criteria and are worth mentioning. The STRUCK [16] method uses a kernelized structured output support vector machine (SVM) with Haar features and histogram features for tracking. The MEEM tracker [17] proposes a multi-expert restoration scheme to address the problem of model drift in online tracking using a linear SVM. The ASMS tracker [18] is based on the popular mean-shift object tracking method. The authors propose various changes to address the problem of scale estimation while processing frames at a high frequency.

Despite all work on visual tracking, very few methods address detection and tracking on airborne images of maritime scenarios. The maritime scenario presents several specific challenges, the most significant ones due to sun reflections (illumination variations), waves and wakes originated by the vessel motion (background clutter), and the fast motion of the camera due to UAV maneuvers. Given the top-down perspective of the camera attached to the UAV, other problems exist, such as the in-plane and out-of-plane rotations. Usually, these types of rotations are not present in a motionless camera. The use of long-wave infrared cameras has been proposed to reduce the effects of sun reflections, waves and wakes. In [19] the author proposes a method to detect and classify maritime objects in infrared videos recorded from an autonomous platform. A fusion of three detection methods is made to generate hypotheses of possible boat locations in the image. The first is based on a track-before-detect algorithm using spatio-temporal integrated blob strength, the second exploits stable image regions and the third is based on tracking salient points of the image. Next, a two-stage-classification step with support vector machines (SVMs) is performed to classify the vessels. In [20] it is proposed an image saliency method and entropy analysis to detect vessels on long wave infrared images.

When the algorithms have to run on limited computational resources on board the UAV, additional concerns must be taken in their development. A recent algorithm for boat detection [2] uses simple blob analysis, based on spatial and temporal constraints and is capable of operation in real-time on board an UAV. In [21] a binary classifier using simple features is used to classify a target as vessel or background from optical satellite images. The classifier has a cascade structure which rejects background clutter in the earlier stages to improve the computational performance. In [22] a complete system for maritime coastal surveillance is proposed. The boat detection is performed using a Haar-like classifier and a temporal filter. The tracking module uses a nearest neighbor policy with the Bhattacharyya distance between the HSV value histograms, allowing for multiple target tracking.

Upon the analysis of the current state-of-the-art, the main concerns are the computational cost of the approaches that are usually the top contenders in the visual object tracking challenge. Furthermore, from our experience in the application of general purpose tracking methods to maritime scenarios, we have noticed frequent failures in tracking regions with sun-reflections, waves and wakes. On longer sequences the tracking tends to drift, and approaches based on key-points fail due to the typical low resolution and lack of texture of the target. In this paper we propose a method that is able to mitigate many of these problems.

III. Methodology

The architecture of this system is shown in Fig. 1. The main components are a correlation filter module and a blob analysis module.

In this work we use a Kernelized Correlation Filter (KCF), following the work of [3]. The correlation filter is initialized by training it with an image patch cropped from the first frame around the target bounding box (BB), either using the raw image (gray pixel or RGB) or using different kinds of features. In the present work we use HoG features [8] and CNN features [23]. The area around this patch is then used as the region of interest (ROI) for the detection in the next frame. The computation of the correlations is performed

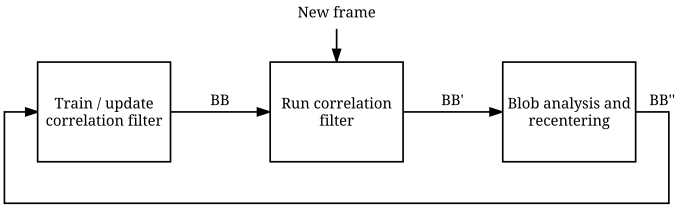

Fig. 1. System architecture. The system receives a new frame and it runs the correlation filter around the previous bounding box (BB) location returning a updated location of the target BB'. After, the error of this estimation (BB') is decreased using blob analysis to detect the vessel and recenter the bounding box (BB"). The correlation filter is updated with the information at the new target location.

in the frequency domain where convolutions are replaced with element-wise multiplications using the Discrete Fourier Transform (DFT) of the images. The response map, where the maximum corresponds to the estimated target location, is obtained using the inverse DFT. This information is used to update the location of the target in the new image and define a new bounding box around it (BB' in Fig. 1). However, due to clutter and noise, the new target estimated position may still be offset with respect to the true position. To improve the target location a blob analysis step is performed. The image is segmented in its most representative blobs and if a dominant large blob is present, its centroid and area are used to update the location and scale of the target, thus defining a new bounding box (BB" in Fig. 1). Finally, the correlation filter is updated with the information from a new patch around the new detection.

In the next sections we will detail the theoretical and practical aspects of the methods used in our implementation.

A. Correlation Filters

Correlations filters are specially tuned to a particular image pattern. They are reminiscent to Template Matching techniques in image processing, where a cropped patch around the target h is used to represent the filter. The correlation of this template with the new image x produces a response map y:

$$y = x * h^{-} \tag{1}$$

where h^{-} is the reflection in both coordinates of the patch and $*$ is the convolution operator. The location corresponding to the maximum value of the response map will indicate the new position of the target. To improve the computational efficiency of the filter, the convolution is computed in the Fourier domain. Because we are in a discrete and finite domain, the convolution must be replaced by the circular convolution ($\otimes$). This produces some boundary artifacts that can be mitigated by pre-weighting the patches with a windowing function. The computations become:

$$y = x \otimes h^{-} = \mathcal{F}^{-1}(\hat{x} \odot \hat{h}^{*}), \tag{2}$$

The hat symbol represents the discrete Fourier transform of a vector and the $\mathcal{F}^{-1}$ represents the inverse Discrete Fourier Transform. The $\odot$ represents the element-wise multiplication and superscript $*$ indicates the complex conjugate.

Using a simple cropped patch to represent the filter produces strong peaks to the target but also responds falsely to the background. To address this problem, methods have been proposed to learn filter coefficients h that produce a more desirable response convolution map y to a given input target x. Typically y is defined as an isotropic Gaussian function with a small standard deviation. A simple way to compute and exact filter is proposed in [24], using the Fourier domain:

$$\hat{h}^{*} = \frac{\hat{y}}{\hat{x}} \tag{3}$$

where the division is element-wise. Still, this approach is not very robust to noise and transformations other than pure translations, so [24] proposed the ASEF algorithm (Average of Synthetic Exact Filters) that, as the name suggests, averages multiple exact filters trained for different transformations of the target:

$$\hat{h}^{*} = \sum_{i} \frac{\hat{y}_i}{\hat{x}_i} \tag{4}$$

where the x_i are transformed image patches of the target, and y_i are the corresponding desired response maps (Gaussian centered in the location of the target).

One year later, the same author proposed the MOSSE filter (Minimum Output sum of Squared Error) allows better performance that ASEF with a smaller number of training patterns [6]. The MOSSE correlation filter is the solution of:

$$\min_{\hat{h}^{*}} \sum_{i} ||\hat{x}_i \odot \hat{h}^{*} - \hat{y}_i||^2, \tag{5}$$

where i indexes each training sample (image). The solution of this optimization problem is given by:

$$\hat{h}^{*} = \frac{\sum_i \hat{y}_i \odot \hat{x}_i^{*}}{\sum_i \hat{x}_i \odot \hat{x}_i^{*}}. \tag{6}$$

This method allowed correlation filters to be trained more efficiently and more robust to variations in lighting, scale, pose and non-rigid deformations.

B. Kernelized Correlation Filters

Despite the success of the MOSSE filter, it is only composed of linear operations on the input signals. Non-linear geometric or brightness transformations of the target can only be represented by increasing the training set of images, which make the method slower. In [3], it was proposed that the correlation filters can be extended to take advantage of the kernel trick to allow for non-linear operations. The correlation filter is formulated as a linear classifier such that the input x is mapped to the output label as $f(x) = \langle \vec{h}, \vec{x} \rangle$, where $\langle \cdot, \cdot \rangle$ means the dot product and $\vec{(\cdot)}$ represents the vectorization operator (reshape a matrix into a vector). Using the Regularized Least Squares (also known as Ridge Regression) the optimization problem becomes:

$$\min_{\vec{h}} \sum_{j} (y(j) - f(x_j))^2 + \lambda ||\vec{h}||^2, \tag{7}$$

where x_j is a transformation of the input and $y(j)$ the desired output (scalar), corresponding to the j^{th} entry of $\vec{y}$, and λ is a regularization term. The solution is given by:

$$\vec{h} = (X^H X + \lambda I)^{-1} X^H \vec{y}, \tag{8}$$

where X is a matrix with one sample $\vec{x}_j$ per row, X^H means the Hermitian transpose $X^H = (X^*)^T$ and I is the identity matrix. The Hermitian transpose is used because the computations will be performed in the Fourier domain where values are usually complex.

By using the kernel trick, the performance can be improved by making the classification on a high-dimensional feature space. The input data is mapped implicitly to a non-linear feature space with the function $\vec{\varphi}(x)$, defined by the kernel as $\kappa(x, x') = \langle \vec{\varphi}(x), \vec{\varphi}(x') \rangle$. Then, the Representer Theorem [25] states that the solution $\vec{h}$ can be expressed by a linear combination of the inputs:

$$\vec{h} = \sum_j \alpha_j \vec{\varphi}(x_j) \tag{9}$$

and the solution to Eq. (7) using kernel functions is:

$$\vec{\alpha} = (K + \lambda I)^{-1} \vec{y}, \tag{10}$$

where K is the kernel matrix with elements $K_{mn} = \kappa(x_m, x_n)$ and I is the identity matrix.

If the transformations in x_j are circular shifts, the computation of the inverse matrix in Eq. (10) can be avoided by the introduction of circulant matrices. Let us consider a one-dimensional vector with a single feature (e.g. pixel value) $u = [u_0, u_1, ..., u_{n-1}]$ where n is the vector length. The equations can then be generalized for 2D images with multiple channels and other multiple channel features as shown in [3]. This approach allows all the translated samples around the target to be used for training without losing computational efficiency. A $n \times n$ circulant matrix $C(u)$ is obtained from a $n \times 1$ vector u by concatenating all cyclic shifts of the vector u:

$$U = C(u) = \begin{bmatrix} u_0 & u_1 & u_2 & ... & u_{n-1} \\ u_{n-1} & u_0 & u_1 & ... & u_{n-2} \\ u_{n-2} & u_{n-1} & u_0 & ... & u_{n-3} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ u_1 & u_2 & u_3 & ... & u_0 \end{bmatrix} \tag{11}$$

Circulant matrices have various interesting properties that will allow the necessary correlation filter computations to be computationally less expensive. For example, the sum, products and inverses of circulant matrices are also circulant. Also, a circulant matrix has the following property:

$$U = F \text{diag}(\hat{u}) F^H \tag{12}$$

i.e. the circulant matrix can be made diagonal with the DFT matrix F, used to compute the DFT of a vector ($\mathcal{F}(u) = \sqrt{n} F u$) and the DFT $\hat{u}$ of the base vector u.

By using this property on Eq. (8) the solution of h can be expressed in the following form in the Fourier domain

$$\hat{h} = \frac{\hat{x}^* \odot \hat{y}}{\hat{x}^* \odot \hat{x} + \lambda}, \tag{13}$$

where the division is element-wise. The complete derivation is available in [3]. Eq. (13) allows for better computational efficiency than the solutions on Eq. (8), being bounded by the DFT operations that have a cost of $\mathcal{O}(n \log n)$ while the Ridge Regression solution has a cost of $\mathcal{O}(n^3)$ bounded by the matrix inversion and products. This is an important property of the system proposed because it allows for real-time computations with great robustness.

The solution above does not yet include the kernel trick. To use the circulant matrix method the kernel matrix K has to be circulant. For that to be true one condition has to be imposed – given a circulant data $C(x)$, the Kernel matrix K is circulant if the kernel function satisfies $\kappa(x, x') = \kappa(Mx, Mx')$ for any permutation matrix M [3]. Some kernels that satisfy this condition are the Gaussian kernel and the linear kernel. The Gaussian kernel is defined for some generic variables x_a and x_b as

$$k_g^{x_a x_b} = e^{-\frac{1}{\sigma^2}(||x_a||^2 + ||x_b||^2 - 2\mathcal{F}^{-1}(\hat{x_a}^* \odot \hat{x_b}))} \tag{14}$$

and the linear kernel as

$$k_l^{x_a x_b} = \mathcal{F}^{-1}(\hat{x_a}^* \odot \hat{x_b}). \tag{15}$$

In this way, Eq. (10) can be written in the Fourier domain as

$$\hat{\alpha} = \frac{\hat{y}}{\hat{k}^{xx} + \lambda}, \tag{16}$$

where k^{xx} is obtained either using the gaussian kernel (14) or the linear kernel (15) using the training sample x. Once again the detailed derivation is presented in [3]. Finally, with Eqs. 14, 15 and 16 it is possible to train the correlation filter.

Another important aspect of correlation filters is how to update the filter with the new information acquired with the new detection. Usually, the running average is the chosen method. In the Kernelized Correlation Filter the coefficients $\hat{\alpha}$ are updated as

$$\hat{\alpha}_t = \eta \frac{\hat{y}}{\hat{k}^{xx} + \lambda} + (1 - \eta) \hat{\alpha}_{t-1}, \tag{17}$$

where t defines the current step, η is the learning rate, and the generic variables x_a and x_b are replaced by x. The target model is updated as

$$\hat{x}_t = \eta \hat{x} + (1 - \eta) \hat{x}_{t-1}, \tag{18}$$

where x is the new sample extracted from the current estimated position of the target.

Given the trained parameter α, the base training samples x_t and the candidate patches for detection z, the detection can be made with the following confidence response map $\bar{y}$:

$$\bar{y} = \mathcal{F}^{-1}(\hat{k}^{x_t z} \odot \hat{\alpha}) \tag{19}$$

by finding the location with maximum value in $\bar{y}$, where $\hat{k}^{x_t z}$ is the DFT of the kernel defined on Eq. 14 or Eq. 15 with the generic variables x_a and x_b replaced by x_t and z respectively.

Fig. 2. From left to right, the region of interest x, the desired output as a 2D Gaussian y and the corresponding correlation filter of the first frame, assuming the use of raw image pixels as features, obtained using Eq. (16) and a Gaussian kernel.

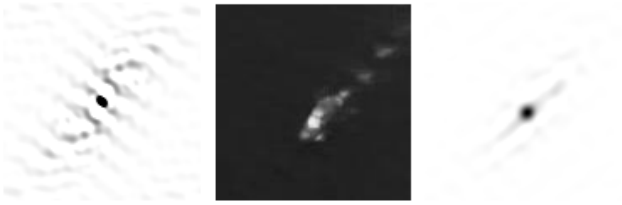

Fig. 3. From left to right, the correlation filter α trained and updated throughout 226 frames, the candidate patch for detection z after 227 frames since the initialization, and the response map $\bar{y}$. Using the trained correlation filter it is possible to estimate the translation of the target relative to the previous position.

In Fig. 2 and Fig. 3 it is possible to get a better understanding of these mathematical concepts during training and detection respectively, by analyzing the image patch and the color maps of the desired output and the obtained correlation filter.

C. Features

Correlation filters work with any kind of dense features that maintain the spatial information. Examples of these features that have shown good performance are raw image pixels, Histogram of Gradients (HoG), Color Names (CN) [26], and more recently Convolutional Neural Network (CNN) features. In this work HoG and CNN are used. When using raw gray pixels the computations are the ones given in the previous section, but for multiple channel features such as RGB, HoG or CNN the kernel computation has a slight difference. For both the linear and the Gaussian kernel the vectors from different channels can be simply added together:

$$k_l^{x_a x_b} = \mathcal{F}^{-1}\left(\sum_c \hat{x}_{a,c}^* \odot \hat{x}_{b,c}\right) \tag{20}$$

$$k_g^{x_a x_b} = e^{-\frac{1}{\sigma^2}(||x_a||^2+||x_b||^2-2\mathcal{F}^{-1}(\sum_c \hat{x}_{a,c}^* \odot \hat{x}_{b,c}))}, \tag{21}$$

where c identifies the channels.

The HoG features used are the ones proposed by Felzenszwalb et al. [8] and are sometimes called Felzenszwalb HoG (FHoG) features. These features are composed of 31 channels where 9 are contrast insensitive, 18 are contrast sensitive and 4 capture the overall gradient energy in different areas.

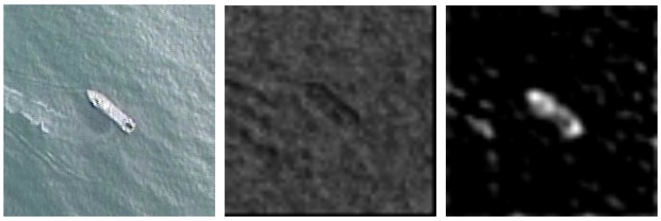

Fig. 4. From left to right, original image, one channel of the FHoG features from a total of 31 and one channel from the CNN features from a total of 256 channels.

For a deeper understanding of the computations required to obtain this features refer to [8]. Fig. 4 shows an example of one channel of the resulting 31 channels after processing a maritime vessel.

Convolutional neural networks can be used as classifiers or regressors. As usual in machine learning, CNN's are trained from a labeled training set in order to produce a desired output. One example is the classification of images such as the images from ImageNet [27]. The VGG-Net [23] won the first and second place in the ImageNet ILSVRC-2014 localization and classification tasks respectively. In the VGG-Net a preprocessed image is given as input and this image will go through 19 layers that are composed of convolutional layers with the rectification (ReLu) non-linearity, max-pooling and fully connected layers. The last layer gives the confidence of an image being one of the 1000 different classes. The appearance of the images is encoded in the neural network weights and the activations of the convolutional hidden layers can be used as multiple channel features for the correlation filter. To extract these features we feed the image to the CNN and use the activation of selected convolutional hidden layers as features.

Note that different layers or even different convolutional neural networks can be used to extract features and the results can vary with the choice. We use the 8th convolutional layer of the VGG-Net [23] with 19 layers that outputs 256 two-dimensional channels (architecture E on the original paper). In Fig. 4, one of the 256 used channels can be examined.

In all cases, each feature channel has to be multiplied by a Hanning window to alleviate the boundary effects when computing the DFT.

The training part of the correlation filter has the following steps when using the FHoG features:

1: Compute features x of the new training sample
2: Multiply x by a Hanning window
3: Compute $\hat{k}_g^{xx} = \mathcal{F}\left(e^{-\frac{1}{\sigma^2}(2||x||^2-2\mathcal{F}^{-1}(\sum_c \hat{x}_c^* \odot \hat{x}_c))}\right)$
4: Update $\hat{\alpha}_t = \eta\frac{\hat{y}}{\hat{k}^{xx}+\lambda} + (1-\eta)\hat{\alpha}_{t-1}$,
5: Update target model $\hat{x}_t = \eta\hat{x} + (1-\eta\hat{x}_{t-1})$

and the detection part of the correlation filter using FHoG features has the following steps:

1: Compute features z of the test sample
2: Multiply z by a Hanning window

3: Compute $\hat{k}_g^{x_t z} = \mathcal{F}\left(e^{-\frac{1}{\sigma^2}(||x_t||^2+||z||^2-2\mathcal{F}^{-1}(\sum_c \hat{x}^*_{t,c}\odot\hat{z}_c))}\right)$
4: Compute $\bar{y} = \mathcal{F}^{-1}(\hat{k}_g^{x_t z} \odot \hat{\alpha})$
5: Determine the arguments of the maxima of $\bar{y}$

When using FHoG, [3] states that the gaussian kernel has better results, as we confirmed experimentally. When testing the CNN features, our results show that the linear kernel achieves a better performance than the gaussian kernel. Consequently, when using CNN features, we changed the algorithms to use the linear kernels k_l^{xx} and $k_l^{x_t z}$.

D. Blob Analysis

The results in [2] show interesting properties of a blob analysis framework in tracking boats in maritime images. Because the appearance of a boat on a maritime background has a blob-like appearance, we propose the use of image segmentation and blob analysis to detect the maritime vessel in the region of interest (instead of the whole image) and to correct the correlation filter tracker.

Image Segmentation. Before the detection of the connected components, usually referred to as blobs, the image patch of interest has to be segmented. In this setting the images are usually composed of two main components: i) the target vessel and ii) the ocean. Thus we adopt a binarization approach via the Otsu's method [28] to separate bright and dark parts of the image that, in principle, will correspond to the boats and the ocean surface, respectively.

The Otsu's method finds a threshold that separates the two peaks of the image histogram from a bimodal image. In mathematical terms, the algorithm exhaustively searches for the threshold t that minimizes the intra-class variance (the variance within the class), defined as a weighted sum of variances of the two classes:

$$\sigma^2_\omega(t) = \omega_0(t)\sigma^2_0(t) + \omega_1(t)\sigma^2_1(t), \tag{22}$$

where $\omega_{0,1}$ are the class probabilities separated by the threshold t and $\sigma^2_{0,1}(t)$ are the class variances. For further details refer to [28].

The segmented image is then eroded (Fig. 5) to isolate the vessel from nearby distractors like wakes, life rafts or other vessels, and also to remove some of the noise originated from waves and sun reflections.

Blob Detection. The next step is to detect the group of pixels (blob) that corresponds to the maritime vessel. We use a method to detect contours in binary images proposed by [29].

Sometimes, the background clutter (waves and wakes) and the sun reflections can interfere with the segmentation. In these cases, the blob detection cannot be used to correct the estimation of the correlation filter because there will be a high number of different blobs and, when going through sun reflections, the vessel will not appear as a blob, as seen in Fig. 5.

To circumvent this problem, a set of heuristics were defined to determine if the conditions allow for the track correction. If a) no blob touches the ROI border; b) the number of blobs is smaller or equal to a defined threshold T_n and c) the blob has an area larger than T_s, then the conditions are favorable to correct the track. The first and second conditions are used to detect the presence of sun reflections or background clutter as boat wakes or waves and the third is used to filter some noise and boat wakes that might go through the two first filters. Afterwards, the blob is chosen using a nearest-neighbor approach with the correlation filter estimation and the track is set to the center of that blob.

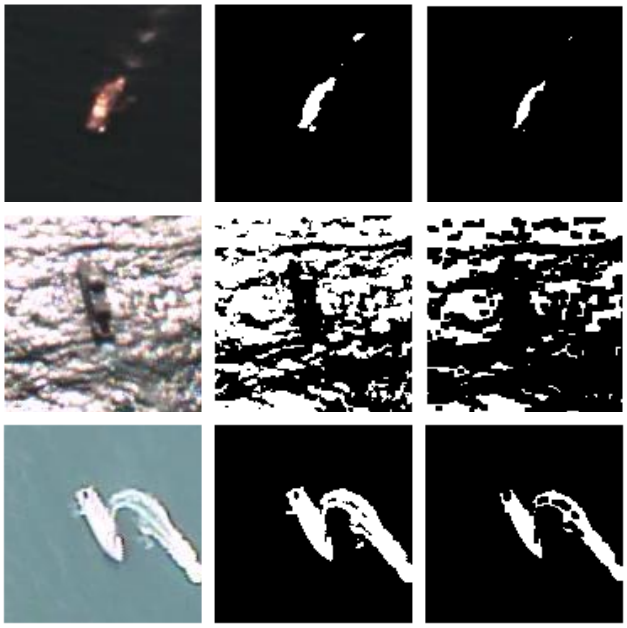

Fig. 5. Original image, segmented image and eroded image (left to right). Three examples shown: a regular case, a case with sun reflections and a case with irregular wake (top to bottom).

To adapt the scale of the target, we set the width and height of the bounding box to a value S_t, proportional to the size of the detected blob unless the size is two times bigger o two times smaller than the previous value ($0.5S_{t-1} < S_t < 2S_{t-1}$). The complete set of steps of the proposed method for one iteration (one frame) is as follows:

if first frame **then**
  Train correlation filter
else
  Estimate translation with the correlation filter
  Perform binary image segmentation
  if no blobs touch the border **and** the number of blobs is smaller then the threshold T_n **and** the blobs are bigger then a threshold T_s **then**
    Determine the blob that is closest to the correlation filter estimation and set the track to the center of that blob
    if $0.5S_{t-1} < S_t < 2S_{t-1}$ **then**
      Update the bounding box width and height to $1.5S_t$
    end if
  end if
  Update the correlation filter
end if

IV. Experiments

A. Evaluation Metrics

To perform the evaluation of the proposed method and compare it to state-of-the-art tracking algorithms, the Object Tracking Benchmark (OTB) [4] framework was used. This methodology evaluates the tracking methods by computing Precision and Success plots of the tracking under two different initialization strategies denoted Temporal Robustness Evaluation (TRE) and Spatial Robustness Evaluation (SRE).

The Precision plot shows the percentage of frames whose Euclidean distance between the centers of the detection and the manually labeled ground truths is lower than a given threshold. The Precision threshold varies from 0 to 50 pixels with a step of 1 pixel. The score chosen to rank the trackers is Precision value for a threshold of 20 pixels as suggested by [4].

The Success plot evaluates the bounding box overlap of the detection with the ground truth. Mathematically, given a detected bounding box r_d and the ground truth bounding box r_{gt} the overlap ratio is given by:

$$S = \frac{|r_d \cap r_{gt}|}{|r_d \cup r_{gt}|}, \tag{23}$$

where $\cap$ and $\cup$ represent the intersection and union, respectively, and $|.|$ represents the number of pixels in that region. Once again, the Success plot shows the percentage of frames whose bounding box overlap ratio is higher than a given threshold from ratio values of 0 to 1, where 1 means perfect match of the detection and ground truth and 0 meaning lost target. To rank the different algorithms, the area under the curve (AUC) of each Success plot is used.

The Temporal Robustness Evaluation consists in initializing the trackers at different frames, not just the first, and running them until the end of the sequence. Each sequence is evaluated by initializing in 20 different frames. The initial frames are chosen by starting with the first frame of the sequence and stepping through them at a regular interval. The step is approximately the number of frames of the sequence divided by 20.

The Spacial Robustness Evaluation consists in introducing error in the initialization by shifting the bounding box by 10% of the target size in 8 different directions, and scaling it by 0.8, 0.9, 1.1 and 1.2 of the ground truth size. This results in 12 different initialization.

These evaluations are pertinent because in a real world scenario the trackers would be initialized with a vessel detector that is likely to introduce error in the initialization.

B. Data set

The data set used for the evaluation is composed of two different videos of the visible spectrum, one acquired with a JAI's AD-080GE camera and the other with a TASE150 camera. Given the limitations of the framework on evaluating videos with out-of-view targets, these videos were cut into smaller sequences. In total, the tracking algorithms run on 8747 manually annotated frames. These smaller sequences include cases where a vessel goes through regions of sun-reflections, cases of big and highly irregular wakes, and cases where the target deploys smaller vessels and life rafts. All frames have the target in the field-of-view.

These and more annotations can be accessed online[1] for further research and validation of the results presented below. See Fig. 6 for some examples of frames from the data set.

C. Tested Algorithms

Some of the state-of-the art methods mentioned above that were available online for testing purposes were evaluated using the OTB framework. The tested trackers were ASMS [18], DSST [7], KCF [3], CF2 [30], SRDCF [10], MUSTer [9], MEEM [17], MDNet [15] and the proposed method either using HoG features or using CNN features.

V. Results

The implementation of the proposed method was made in Python and can be accessed online[2]. To evaluate this method, the parameters used for the correlation filter and blob analysis were as follows. The desired output y for the training step is a M by N gaussian with its peak at the target center:

$$y(m,n) = e^{-\frac{(m-M/2)^2+(n-N/2)^2}{2\sigma^2}}, \tag{24}$$

where $m = 0, 1, 2, ..., M$, $n = 0, 1, 2, ..., N$, and $\sigma = 0.1 \cdot \sqrt{a}$ is the peak width, proportional to target area a. The region of interest is defined as a square 2.5 times larger than the target bounding box. The regularization weight λ used was 10^{-4} and the learning rate η was set to 0.02. When using the HoG features, the kernel used was the Gaussian kernel, with a standard deviation of 0.5, while the linear kernel was used for the CNN features.

The FHoG features were extracted using the Dlib [31] implementation while the CNN features were extracted using the Caffe [32] framework.

For the image segmentation method [28] and the blob detector [29], we use the OpenCV Library [33] implementations. To adapt the bounding box to the target, we use a value given by the OpenCV blob detector that is proportional to the blob size. This variable, called size S, is part of the OpenCV Keypoint class and is defined as "diameter of the meaningful keypoint neighborhood". To allow for some context to be taken into account during the correlation filter step, the width and the height is set to $1.5S_t$. The threshold used for the number of blobs was $T_n = 2$ and the minimum size threshold was $T_s = 10$.

As can be seen in Fig. 7 and Fig. 8, our method, either using FHoG or CNN features, shows better results that the other methods in the airborne maritime scenario. The key is the blob detection and target re-centering steps, which allow greater precision, especially with the version using CNN features, either in the Spatial Robustness Evaluation (SRE) or in the Temporal Robustness Evaluation (TRE). The FHoG

[1] http://vislab.isr.ist.utl.pt/seagull-dataset/

[2] https://github.com/Magnesiam/CFT_OTB/

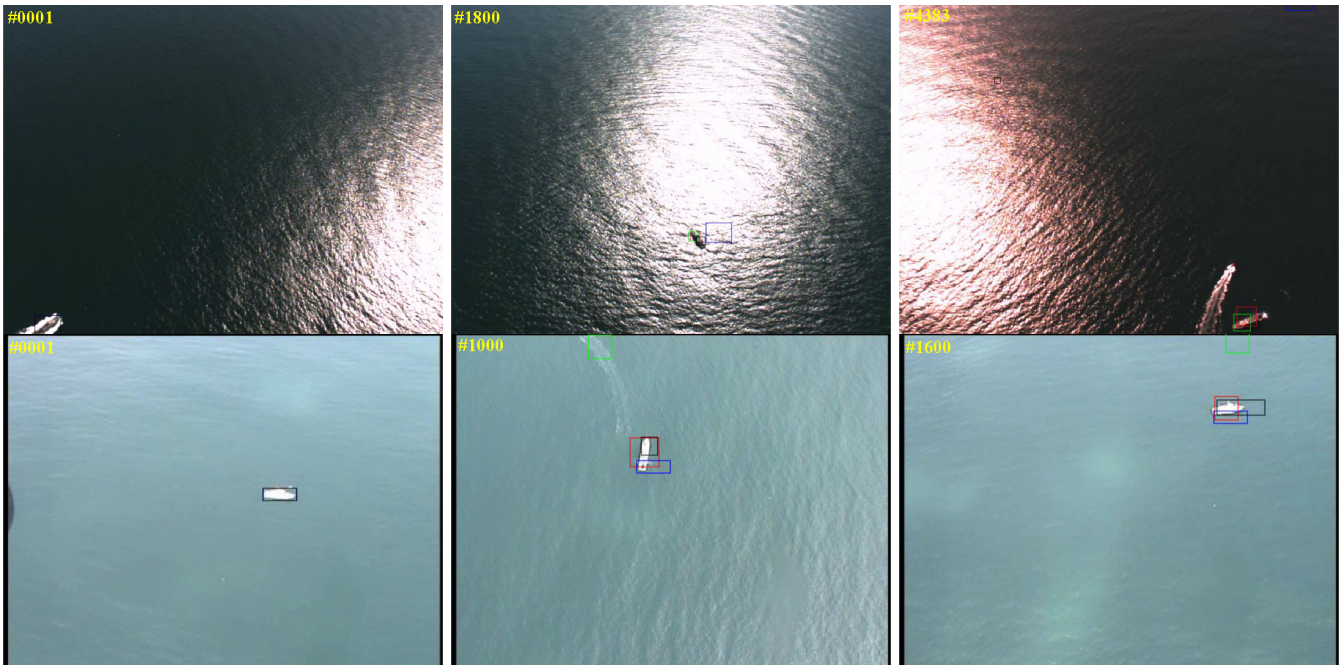

Fig. 6. Frames from the data set with the results, as bounding boxes, of some of the best performing methods: both our methods (CNN as red and HoG as green), MDNet (black) and CF2 (blue). Two examples are shown. The example on top is a case of success for both our methods given that the track is still on the target after 4383 frames, when the target leaves the camera's field of view, even after going through a region with intense sun-reflections multiple times. The other top performing algorithms lose the target much earlier. The example on the bottom is a case of failure for our method using FHoG features, where the track gets lost following the wake of another target. Nevertheless, our method using CNN features can overcome this problem given its capability to better discriminate the target from the wakes.

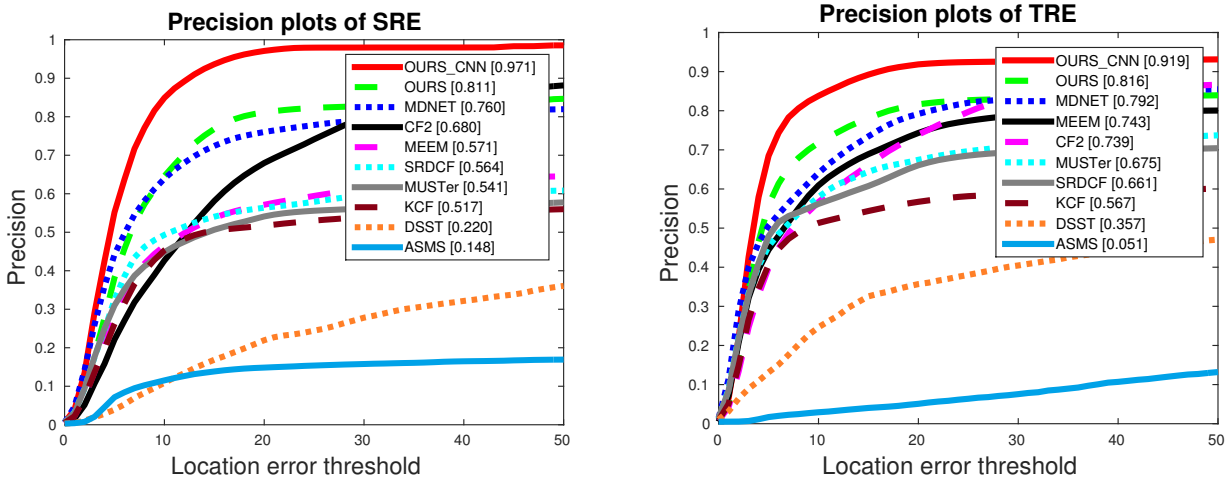

Fig. 7. Precision plots for the SRE and TRE from the OTB framework [4]. The values on the legend correspond to the precision for a location error threshold of 20. Note that the colors represent the ranking and not the different trackers. The same tracker can have different colors in different evaluations depending on its ranking (e.g. red corresponds to the first place, green to the second place and blue to the third place).

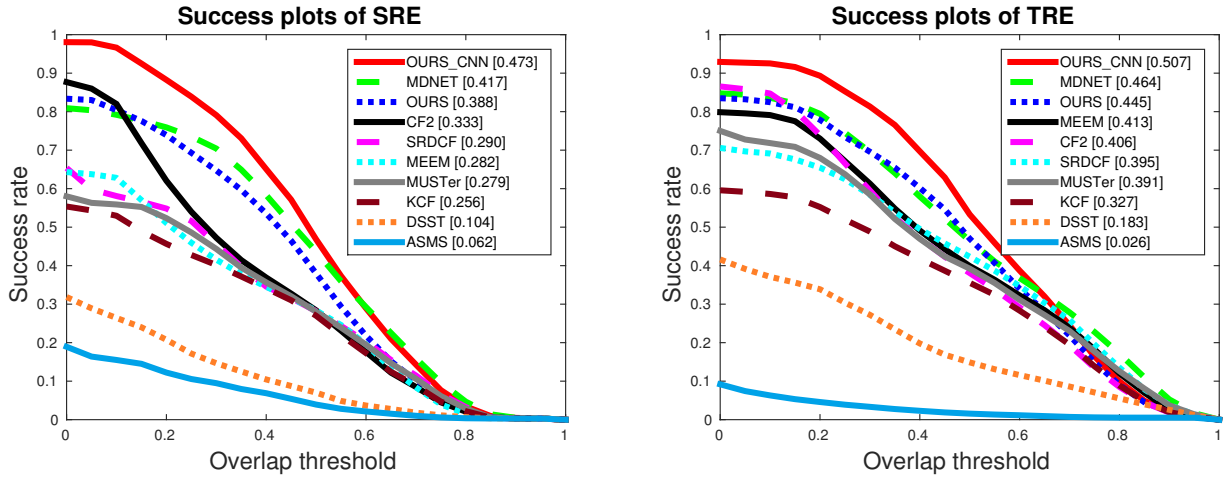

Fig. 8. Success plots for the SRE and TRE from the OTB framework [4]. The values on the success plots legend correspond to the areas under the curves (AUC). Note that the colors represent the ranking and not the different trackers. The same tracker can have different colors in different evaluations depending on its ranking (e.g. red corresponds to the first place, green to the second place and blue to the third place).

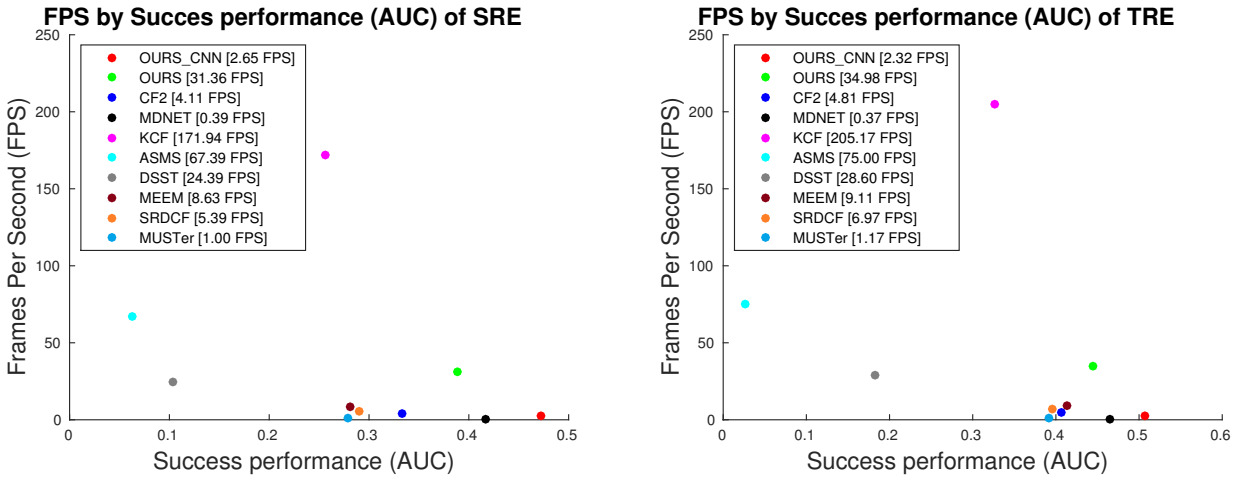

Fig. 9. Performance by computational complexity measured in frames per second (FPS) for the SRE and TRE. Success performance is the area under the curve obtained from the plots shown in Fig. 8.

version also presents competitive results when compared to the MDNen (winner of the VOT Challenge 2015) but running with a frame-rate that is two orders of magnitude faster (Fig. 9). Furthermore, it has better results around the 10 pixels location error threshold (Fig. 7). Some trackers that have great results in general purpose tracking challenges and benchmarks seem not to generalize well enough for the particular case of airborne maritime imagery, such as the ASMS and the DSST. The DSST, even tough being a part of the correlation filter tracking family, it does not use the kernel trick and the scale space search does not work well for the rotations present in this scenario.

In the success rate evaluation there are clearly three top ranking algorithms, particularly on the spacial robustness evaluation, as shown in Fig. 8. These are both of our methods along with MDNet. These methods seem to work well even in the presence of an erroneous initialization. In our case, it is due to the blob analysis that corrects the error in the initialization as soon as the conditions are favorable. In the case of the MDNet, it is safe to assume that this happens because the neural network trained offline has encoded the nuances of a moving target and can converge to its location.

Another important factor to take into account is the computation complexity. The evaluations of Fig. 9 were made using an Intel Xeon CPU W3503 at 2.40GHz and a GeForce GTX 750 graphics card. The CNN computations of several methods were run in parallel in the graphics card. Unfortunately, the top two methods, ours using CNN features and MDNet, run at a low frame-rate. Ours, with CNN features, has a mean frame-rate in the SRE of 2.65 FPS and 2.32 FPS in the TRE and MDNet has 0.39 FPS and 0.37 FPS respectively. To achieve these frame-rate they require parts of the computations to run in a graphics card, otherwise the results would be even slower. The best performing approach that is suitable to be used in a

real-time system is the method proposed in this work with the use of FHoG features. In this case, the algorithm shows a frame-rate of 31.36 FPS in the SRE and 34.98 in the TRE and does not require the use of a graphics card.

VI. Conclusions

In this work we have presented and benchmarked a new algorithm for tracking that performs beyond the state-of-the-art in maritime scenarios. Some of the problems concerning the detection and tracking of vessels in airborne oceanographic imagery, namely sun-reflection, waves, wakes and long-term tracking, are overcome thanks to a combination of correlation filters and blob analysis. We present two versions of the algorithms. The best performing is based on CNN features and significantly outperforms current state-of-the-art general purpose trackers. The second version uses HoG features and attains performances competitive with the current state-of-the-art, but at a much faster frame-rate, suitable for real-time operation on airborne systems.

For future work it is proposed the fusion of a detection module with the tracking step proposed here to allow for a completely autonomous system. Also, improvements on the bounding box or target segmentation could increase the precision and success rate of the algorithm. A multiple target tracking evaluation is also important but, for a meaningful evaluation, requires more videos with multiple vessels.

Acknowledgments

This work was partially supported by project PTDC/EEIPRO/0426/2014 and the Portuguese Government through FCT project [UID/EEA/50009/2013].

References

[1] M. M. Marques, P. Dias, N. P. Santos, V. Lobo, R. Batista, D. Salgueiro, A. Aguiar, M. Costa, J. E. da Silva, A. S. Ferreira, J. Sousa, M. de Fátima Nunes, E. Pereira, J. Morgado, R. Ribeiro, J. S. Marques, A. Bernardino, M. Griné, and M. Taiana, "Unmanned aircraft systems in maritime operations: Challenges addressed in the scope of the seagull project," in *OCEANS 2015 - Genova*, May 2015, pp. 1–6.

[2] J. S. Marques, A. Bernardino, G. Cruz, and M. Bento, "An algorithm for the detection of vessels in aerial images," in *Advanced Video and Signal Based Surveillance (AVSS), 2014 11th IEEE International Conference on*. IEEE, 2014, pp. 295–300.

[3] J. F. Henriques, R. Caseiro, P. Martins, and J. Batista, "High-speed tracking with kernelized correlation filters," *Pattern Analysis and Machine Intelligence, IEEE Trans.*, vol. 37, no. 3, pp. 583–596, 2015.

[4] Y. Wu, J. Lim, and M.-H. Yang, "Online object tracking: A benchmark," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2013.

[5] "The visual object tracking vot2015 challenge results," Dec 2015. [Online]. Available: http://www.votchallenge.net/vot2015/

[6] D. S. Bolme, J. R. Beveridge, B. A. Draper, and Y. M. Lui, "Visual object tracking using adaptive correlation filters," in *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*. IEEE, 2010, pp. 2544–2550.

[7] M. Danelljan, G. Häger, F. Khan, and M. Felsberg, "Accurate scale estimation for robust visual tracking," in *British Machine Vision Conference, Nottingham, September 1-5, 2014*. BMVA Press, 2014.

[8] P. F. Felzenszwalb, R. B. Girshick, D. McAllester, and D. Ramanan, "Object detection with discriminatively trained part-based models," *IEEE transactions on pattern analysis and machine intelligence*, vol. 32, no. 9, pp. 1627–1645, 2010.

[9] Z. Hong, Z. Chen, C. Wang, X. Mei, D. Prokhorov, and D. Tao, "Multi-store tracker (muster): A cognitive psychology inspired approach to object tracking," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 749–758.

[10] M. Danelljan, G. Hager, F. Shahbaz Khan, and M. Felsberg, "Learning spatially regularized correlation filters for visual tracking," in *Proceedings of the IEEE International Conference on Computer Vision*, 2015, pp. 4310–4318.

[11] M. Danelljan, G. Hager, F. Khan, and M. Felsberg, "Convolutional features for correlation filter based visual tracking," in *Proceedings of the IEEE Int. Conf. on Computer Vision Workshops*, 2015, pp. 58–66.

[12] C. Ma, J.-B. Huang, X. Yang, and M.-H. Yang, "Hierarchical convolutional features for visual tracking," in *Proceedings of the IEEE International Conference on Computer Vision*, 2015.

[13] K. Chatfield, K. Simonyan, A. Vedaldi, and A. Zisserman, "Return of the devil in the details: Delving deep into convolutional nets," *arXiv preprint arXiv:1405.3531*, 2014.

[14] N. Wang and D.-Y. Yeung, "Learning a deep compact image representation for visual tracking," in *Advances in Neural Information Processing Systems 26*, C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, Eds. Curran Associates, Inc., 2013, pp. 809–817.

[15] H. Nam and B. Han, "Learning multi-domain convolutional neural networks for visual tracking," *arXiv preprint arXiv:1510.07945*, 2015.

[16] S. Hare, S. Golodetz, A. Saffari, V. Vineet, M.-M. Cheng, S. Hicks, and P. Torr, "Struck: Structured output tracking with kernels," 2014.

[17] J. Zhang, S. Ma, and S. Sclaroff, "MEEM: robust tracking via multiple experts using entropy minimization," in *Proc. of the European Conference on Computer Vision (ECCV)*, 2014.

[18] T. Vojir, J. Noskova, and J. Matas, "Robust scale-adaptive mean-shift for tracking," *Pattern Recognition Letters*, vol. 49, pp. 250–258, 2014.

[19] M. Teutsch and W. Krüger, "Classification of small boats in infrared images for maritime surveillance," in *2010 International WaterSide Security Conference*. IEEE, 2010, pp. 1–7.

[20] G. Cruz and A. Bernardino, "Image saliency applied to infrared images for unmanned maritime monitoring," in *International Conference on Computer Vision Systems*. Springer, 2015, pp. 511–522.

[21] G. Mattyus, "Near real-time automatic vessel detection on optical satellite images," in *ISPRS Hannover Workshop*. ISPRS Archives, 2013, pp. 233–237.

[22] D. Bloisi, L. Iocchi, M. Fiorini, and G. Graziano, "Automatic maritime surveillance with visual target detection," in *Proc. of the International Defense and Homeland Security Simulation Workshop (DHSS)*, 2011, pp. 141–145.

[23] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *CoRR*, vol. abs/1409.1556, 2014.

[24] D. S. Bolme, , B. A. Draper, and J. R. Beveridge, "Average of synthetic exact filters," in *Computer Vision and Pattern Recognition (CVPR), 2009 IEEE Conference on*. IEEE, 2009, pp. 2105–2112.

[25] B. Schölkopf and A. J. Smola, *Learning with kernels: support vector machines, regularization, optimization, and beyond*. MIT press, 2002.

[26] M. Danelljan, F. Shahbaz Khan, M. Felsberg, and J. Van de Weijer, "Adaptive color attributes for real-time visual tracking," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2014, pp. 1090–1097.

[27] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "ImageNet: A Large-Scale Hierarchical Image Database," in *CVPR09*, 2009.

[28] N. Otsu, "A threshold selection method from gray-level histograms," *Automatica*, vol. 11, no. 285-296, pp. 23–27, 1975.

[29] S. Suzuki *et al.*, "Topological structural analysis of digitized binary images by border following," *Computer Vision, Graphics, and Image Processing*, vol. 30, no. 1, pp. 32–46, 1985.

[30] C. Ma, J.-B. Huang, X. Yang, and M.-H. Yang, "Hierarchical convolutional features for visual tracking," in *Proceedings of the IEEE International Conference on Computer Vision*, 2015, pp. 3074–3082.

[31] D. E. King, "Dlib-ml: A machine learning toolkit," *Journal of Machine Learning Research*, vol. 10, pp. 1755–1758, 2009.

[32] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell, "Caffe: Convolutional architecture for fast feature embedding," *arXiv preprint arXiv:1408.5093*, 2014.

[33] G. Bradski, "The OpenCV Library," *Dr. Dobb's Journal of Software Tools*, 2000.