TÉCNICO LISBOA

Vessel Detection in Oceanographic Airborne Imagery

Miguel António Azevedo Griné

Thesis to obtain the Master of Science Degree in

Electrical and Computer Engineering

Supervisor: Professor Alexandre José Malheiro Bernardino

Examination Committee

Chairperson: Professor João Fernando Cardoso Silva Sequeira
Supervisor: Professor Alexandre José Malheiro Bernardino
Member: Professor Jorge dos Santos Salvador Marques

November 2014

Contents

Agradecimentos

É com bastante orgulho que escrevo esta secção, pois significa o culminar de uma etapa extremamente importante na minha vida académica.

Gostaria de aproveitar este espaço para deixar um agradecimento especial ao Matteo Taiana, Ricardo Ribeiro e Andreia Moço pelas inúmeras discussões frutíferas que fomos tendo ao longo deste semestre bem como todo o apoio providenciado.

Quero dedicar este trabalho à minha família, em especial aos meus pais e irmão, pelo tempo e paciência disponibilizados.

Aos meus amigos António Fitas, Cláudio Martins, Duarte Cardão, Filipe Guapo, Henrique Carvalho, Miguel Paixão, Rafael Gamas, Rui Martins, Stephano Pugliese e Vasco Pinho, agradeço e recordo todos os momentos que passámos juntos ao longo destes meses.

Resumo

Este trabalho foca-se no processamento de imagem para detectar embarcações num ambiente oceanográfico, através de imagens captadas por meios aéreos não tripulados, em tempo real. A falta de largura de banda nas comunicações entre o veículo aéreo e a estação de controlo limita a capacidade para enviar todos os dados adquiridos pelo drone (veículo aéreo não tripulado) para a estação base, pelo que as imagens têm que ser avaliadas em tempo-real de modo a decidir se um evento específico precisa de atenção especial. O problema é atacado a partir da detecção inicial de eventos anómalos, i.e. regiões de imagem que são diferentes da aparência normal da superfície marítima, as quais serão posteriormente processadas por detectores de eventos mais específicos. Uma vez detectada uma embarcação, uma imagem contendo apenas esse evento é enviada para a estação de terra para posterior validação humana. Ao analisar a situação, o utilizador humano poderá decidir qual o procedimento a seguir, nomeadamente, seguimento do veículo náutico, zoom na câmara para verificação de matrícula, entre outros. Assim sendo, a ideia principal é usar um detector de barcos eficiente, permitindo decidir quais serão os *patches* de imagens que necessitam de ser analisados com mais detalhe.

O problema proposto terá três requisitos principais: custo computacional baixo devido aos limitados recursos computacionais a bordo do veículo, invariância à rotação dado que a embarcação pode ser observada de várias perspectivas, e tolerância à escala, uma vez que o veículo aéreo pode voar a diversas altitudes. Pretende-se ainda baixa taxa de falsos alarmes devido ao custo elevado das comunicações, e uma taxa de detecção que permita detectar um evento pelo menos uma vez enquanto o barco se encontrar dentro do campo de visão da câmara. Para detectar regiões de interesse, implementámos vários métodos de detecção de anomalias por saliência. Posteriormente, nestas regiões relevantes, serão então aplicados detectores mais específicos, dependentes dos objectos que se procuram. Construímos, então, um detector de padrões de embarcações baseado em descritores HOG invariantes a rotações, fazendo uso da análise de Fourier em coordenadas polares. Foram etiquetadas milhares de imagens para a criação de uma base de imagens de treino e *benchmarking* dos algoritmos. Um modelo de treino supervisionado, correndo uma Support Vector Machine (SVM) linear, foi usado para classificação.

Palavras chave: Análise de Fourier, Aprendizagem Supervisionada, Descritores de Imagem, Histogramas de Gradientes Orientados, Invariância à Rotação, Saliência

Abstract

This work focuses on the use of image processing techniques to detect vessels in an oceanographic environment through aerial images captured by Unmanned Aerial Vehicles (UAVs). The lack of bandwidth in communications between the aerial vehicle and the base station limits the capacity to send all the data acquired by the drone, thus the images have to be assessed in real-time in order to decide if a specific event needs special attention. This problem is initially addressed by the detection of anomalous events, i.e. regions which are different from the usual appearance of the maritime background. Then, those regions will be deeply processed by specific event detectors. Once a vessel is detected, the image patch containing that event is sent to the base station in order to be validated by a human operator. Analyzing the situation, the user is able to decide which is the best procedure, namely tracking of a target or zoom in to check the license plate of a boat, among others. Therefore, the main idea is to use an efficient vessels detector allowing to decide which image patches need to be duly analyzed.

The presented problem has three main requirements: computational cost due to the limited computational resources, the need of rotation invariance since the vessel can be observed by different perspectives, and scale tolerance due to the different flight altitudes and different sizes of vessels. It is also important to ensure a low false positive rate since the communications are expensive, and a detection rate allowing to detect an event, at least, while the vessel is within the field of view of the camera. In order to detect conspicuous regions, some saliency methods were implemented. Then, in those regions, a powerful detector will be applied. A vessel detector was constructed based on rotation-invariant HOG descriptors using Fourier analysis in polar coordinates. Thousands of images were labelled in order to create an image database and algorithm benchmarking. A supervised model was trained by running a SVM.

Keywords: Fourier Analysis, Histogram of Oriented Gradients, Image Descriptor, Rotation-Invariance, Saliency, Supervised Learning

List of Acronyms

ACF Aggregated Channel Features

BB Bounding Box

CC Connected Component

DTD Dominant Texture Descriptor

FN False Negative

FP False Positive

FFT Fast Fourier Transform

GBVS Graph-Based Visual Saliency

GT Ground Truth

HOG Histograms of Oriented Gradients

HSV Hue-Saturation-Value

IFFT Inverse Fast Fourier Transform

MSE Mean Squared Error

k-NN k-Nearest-Neighbors

NMS Non-Maximum Suppression

PR Precision-Recall

RF Random Forest

RGB Red-Green-Blue

SIFT Scale Invariant Feature Transform

SVM Support Vector Machine

TN True Negative

TP True Positive

UAV Unmanned Aerial Vehicle

List of Figures

List of Tables

Chapter 1

Introduction

This thesis is part of the SEAGULL project which aims at patrolling the coast line using drones, since they are a practical and inexpensive solution (approximately 5 hour autonomy with a cost of 50cent per hour) besides requiring few communication and human resources[1]. One of the objectives of the project is the detection of ships involved in illegal activities at sea (fishing, traffic, pollution). Hence, this work aims at developing a semi autonomous system capable of detecting vessels in oceanographic environments. In this work we researched for the best methodologies to be used, their implementation, benefits and drawbacks. The main goal is to build an algorithm which is able to process, in real-time, the images captured by the drone. Due to the lack of bandwidth in communications between the aerial vehicle and the base station, it is impossible to send all images captured for inspection, therefore it is required to filter those images and send only the relevant ones. Once an event is detected, we just send the corresponding image patch to the base station in order to be validated by a human operator, deciding what is the best way to deal with that. Fig. 1.1 shows the workflow of our system. The solution which is proposed in this work should be able to generalize in order to detect other kind of events such as different types of vessels, oil spills, shipwrecked and lifeboats. By solving the proposed problem, the Portuguese Air Force and Navy will be able to patrol in an efficient way the portuguese coast line, spending few resources. It will help to track targets, check their license plate and prevent illegal activities such as drug smuggling. On the other hand, it will be important to control environmental disasters, by delimiting the areas of oil spills. The identification of shipwrecked and lifeboats will also be fundamental to act immediately in order to help people in danger.

In order to patrol efficiently the coast line, it is important to develop a good descriptor of what we are looking for, so that it will be robust to adverse conditions such as low resolution, varying illumination conditions and sun reflections. Aditionally, we need to deal with flight manoeuvres, different flight altitudes and a huge variety of boat shapes. Even for a known boat, we need to deal with its observation from different perspectives and rotations. From the stated problems, the need of rotation-invariance reveals as one of the major challenges to overcome. Therefore, we adopted rotation-invariant HOG descriptors (Liu et al. [22]) to obtain the desired features which will compose the training examples to be used in a supervised learning model, as a SVM (Boser et al., [3]; Cortes & Vapnik, [6]). Regarding scale-invariance, we approach the problem by the selection of samples with sufficient scale diversity to compose the training set of the classifier. The processing time is, as usual, a big concern, since we need to build a solution which will be able to respond in real time. The computer which is embeded in the

[1]For further information about the project, please refer to http://www.criticalsoftware.com/pt/seagull.

Figure 1.1: Workflow: the developed prototype should detect anomalous events in the images captured by the aerial vehicle; when one of those events is detect, the image patch containing it is sent to the base station in order to be assessed by a human operator.

drone has a small processor and hard power constraints, thus it is important to reduce the computational cost of the algorithms which compose our prototype.

We alert to the fact that, despite the current solution is completely oriented to detect a specific target, it could be used to detect other types of objects, provided a labelled database of samples of those objects. In other words, in the process of gathering features to train the model, it is just needed to use a different dataset that characterizes as well as possible the target object we are tracking. This is very important since, for now, our model is just able to detect vessels. To detect other objects, we need to train a new model with different positive samples.

We also would like to highlight that we are focusing in analysis of isolated images instead of sequences, therefore we are not concerning about temporal events. However, establishing temporal coherence between frames would allow to achieve more reliable results.

As main contributions of this work, a new saliency method was developed which reveals more efficient in terms of computational cost and presents very satisfactory results in our problem. The development of an algorithm to detect regions of sun reflections revealed very useful, avoiding to waste time by processing those regions of an image. Beyond that, some strategies were developed in order to improve the computational cost of the method presented by Liu et. al ([22]), by using FFTs (refer to section 4.2). The combination of our own preprocessing methods with the detection method proposed by Liu et al. was well succeeded, since we achieve better results by taking considerably less time. The use of the state-of-the-art algorithms in this dataset (maritime environment) is also new.

The rest of the report is organized as follows: in chapter 2, we review some related work; chapter 3 presents the proposed approach to solve the problem, as well as a detailed explanation of the algorithms; then, chapter 4 describes the proposed algorithmic improvements which optimize the computational cost of the solution provided; in chapter 5, the dataset is introduced, some implementation details are given as well as the evaluation methods

and results attained are presented; finally, chapter 6 summarizes the work performed and highlights the main achievements and contributions in this work as well as provides some ideas of what can be done in the future.

Chapter 2

Related Work

In order to know what is already developed in this area, we studied a survey of object recognition ([2]) containing techniques from the past 50 years, giving special attention to the local feature-based recognition methods. From this survey, we were able to get some ideas to create a solution for our problem. The general approach designed consists in two main stages. Initially, we select the most conspicuous regions from a given image, using saliency methods. In other words, we are selecting relevant regions relatively to the ocean background, which can be interesting or not. Since this step highlights many regions containing sun reflections, this preprocessing stage has another step which is the filtering of those events in order to avoid processing them. The goal of this stage is to reduce the areas where to apply a more powerful algorithm, reducing the overall computational cost. Afterwards, it is applied a window based classifier to search these regions for a specific type of object. In this step, we should take into account the need of rotation-invariance, since the drone will change its direction frequently, meaning that we will see the targets we are tracking in different perspectives.

2.1 Saliency Methods

Saliency methods are an useful tool to filter the relevant regions of a given image. These kind of methods are really important in recognition tasks, since they allow to detect salient objects. They help as a preprocessing step to a classification task because they highlight salient regions, restricting areas where we can look for a specific target. The Itti & Koch approaches ([19], [18], [17]) are well known for their biological inspiration, although their precision is considerably lower than the state of the art algorithms. Graph-Based Visual Saliency (GBVS), proposed by Harel et al. ([15]), is a new bottom-up visual saliency model, which exploits the distributed nature of graph algorithms. This algorithm differs from the others due to its alternative activation and normalization steps. Therefore, the proposed graph-based solution uses local computation to obtain a saliency map which is everywhere dependent on global information. Interestingly, it matches the human attentional system, being able to predict human fixations with a satisfactory precision. Kadir & Brady ([20]) present a multiscale algorithm for the selection of salient regions of an image. In their work, saliency, scale selection and content description are intrinsically related aspects which are dealt with in conjunction. Hou et al. ([16]) introduced a simple image descriptor referred to as the *image signature*, which spatially approximates the foreground of an image, allowing to build a saliency algorithm based on this concept. This algorithm predicts human fixation points best among competitors, doing so

in much shorter running time. Although the results obtained with the methods proposed by Harel et al. ([15]) and Hou et al. ([16]) were very satisfactory, the computational cost revealed as a problem. Thus, it was developed a simple algorithm to highlight interesting regions based on contrast between pixels, making use of the potentialities of the gradient of an image. This solution, which was integrated in the final system, will be described in 3.1.1.

As an alternative way to solve the presented problem, we considered using image segmentation techniques. Malik et al. ([25]) presented an algorithm for partioning grayscale images into disjoint regions of coherent brightness and texture. They explored, simultaneously, contours and texture, introducing the concept of texturedness of the neighborhood at a pixel in order to know how likely two nearby pixels are to belong to the same region.

2.2 Classification Methods

After salient regions have been detected, they have to be further analysed for search of relevant items. Typical methods compute discriminant features on those regions that are then matched to specific object models. Scale Invariant Feature Transform (SIFT) is a popular solution for extracting distinctive invariant features from images which can be used to perform reliable matching between different views of an object pose (Lowe, [24]). SIFT aligns a local coordinate system to the dominant gradient direction at each detected interest point. This pose normalization is based on the assumption that the referred dominant gradient orientation is available. However, this method does not work well for arbitrary positions or dense feature computation. As verified by Lin et al. ([21]), the orientation ambiguity is the main source of error in dense image alignment using SIFT features.

Dalal and Triggs ([7]) introduced the HOG which nowadays are widely used in image processing. The problem of their concept is the lack of rotation-invariance, as it will be shown in 3.2.1. Standard 2D object recognition approaches use the non-invariant dense HOG features complemented with a sliding window classifier (Felzenszwalb et al., [13]).

Fadeev & Frigui ([11]) proposed a generic approach for computing image texture features, referred as Dominant Texture Descriptor (DTD). The main idea is to cluster the local texture features and identify the dominant components and their spatial distribution. Then, the classification task is done using a k-Nearest-Neighbors (k-NN) classifier. Since this technique is based on discrete HOGs, which are computed according to a fixed coordinate system, it is not rotation-invariant. As it was referred, the rotation-invariance property is crucial to solve the proposed problem. Therefore, we focused on work developed by Liu et al. ([22]), a method that uses rotation-invariant HOG descriptors using Fourier analysis in polar coordinates. The main difference to the standard techniques is that Liu et al. consider a gradient histogram as a continuous angular signal which can be well represented by the Fourier basis. Hence, they avoid discretization artifacts (keeping the substantial information about the gradient patterns) and create a continuous and smooth mapping from the raw image to the feature space (final descriptors), besides being rotation-invariant. This method will be carefully explained in 3.2.2, which allows to perform the classification task achieving very satisfactory results, as it will be shown in chapter 5. The reason why this technique called our attention was the fact that this method outperforms state-of-the-art algorithms on a challenging aerial car detection task, which also address the rotation-invariance problem, but in different ways. Vedaldi et al. ([30]) uses the standard HOG feature with a nonlinear structured SVM classifier. Schmidt & Roth ([28]) focuses on features

and descriptors. They propose rotation-aware feature learning, i.e., it yields features that have a notion of which specific image transformation they are used with. The results cited by Liu et al. ([22]) are significantly better than the ones shown in [28] and [30], mainly when they use a nonlinear classifier (Random Forest, [4]) instead of a linear SVM ([6]). In chapter 5, it is possible to check the PR curves as well as the Miss Rate/False Positives per Image curves drawn for several performed tests, in order to understand how our solution is behaving using this method.

Besides the sliding window techniques we are implementing, it is possible to perform detections in another way. There is a fast rotation-invariant detection framework published by Reisert & Burkhardt ([27]), which proposes a new class of rotation-equivariant nonlinear filters based on Group Integration and Fourier analysis. The referred framework is related to Hough voting methods, being definitely better and faster than the alternative approaches. In Skibbe & Reisert ([29]), equivariant filters using the HOG represented in Fourier space (circular features) are used for rotation invariant 2D object detection in biological applications. Liu et al. ([23]) used an improved equivariant filter using the Fourier HOG based regional descriptors for motorbike detection in images from freestyle motocross.

2.3 Combined Methods

Dawkins et al. [8] developed an approach which automatically detects, tracks and classifies different objects within aerial images of a maritime environment. First, they remove regions of the image belonging to background (such as calm water and sky) by applying a multi-layered saliency detector. Then, they classify the salient foreground into wake and non-wake detections. This detector extracts several features generated at the level of each individual pixel, which are highly correlated with the target objects, grouping them into Connected Components (CCs). Afterwards, a track initialization is performed by comparing the CCs across multiple frames, assuming a linear motion. Kalman filtering, descriptor and template matching techniques are used to update the tracks. Lastly, bag-of-words and HOG descriptors are computed around active tracks in order to be used to classify them using machine learning techniques.

Chapter 3

Approach and Methodologies

Human beings own amazing capacities to process different scenarios, correctly deciding which are the objects that need more attention, without spending a lot of resources. Hence, scientific community is researching on how to provide human perception in an automatic way. Core applications of machine vision, including scene understanding, content-based image retrieval, object recognition and tracking, are revealing challenging for engineers, yet they are managing to overcome the obstacles found.

To solve the presented problem, we outlined a strategy which will be composed by two main steps taking into consideration the requirements stated in chapter 1, when formulating the problem, namely the lower computational cost, the need of rotation invariance and scale variation tolerance. The first one can be understood as a preprocessing stage which aims to reduce the computational cost of our final solution, since the available computational power is restricted. Therefore, saliency maps ([15], [16], [17], [18], [19]) are initially computed in order to highlight the most relevant regions of a given image. However, those saliency maps take approximately $1s$ to compute which is not acceptable since we are dealing with a real-time problem. Thus, we alternatively developed a faster method to compute regions of interest by searching for areas with high contrast relative to surrounding areas (3.1.1). Since we noticed that the saliency methods were highlighting regions of vessels and sun reflections, we developed a filter to explicitly detect sun reflections (3.1.2). On the second stage, only regions not filtered by preprocessing steps will be further analyzed. Then, we use rotation-invariant HOG descriptors using Fourier analysis in polar coordinates ([22]) to learn classifiers for boats (3.2.2). The way the training samples given to the classifier are chosen allows one to achieve scale tolerance within a certain range (3.3). This range was defined according to some studies performed, covering sizes of boats which we can aspire to detect.

In Fig. 3.1, it is possible to visualize a diagram comprising the different steps of the outlined approach. Those are the main stages of our approach which were integrated into a final solution. The following sections provide a detailed explanation of each step and its contribution to the solution outlined.

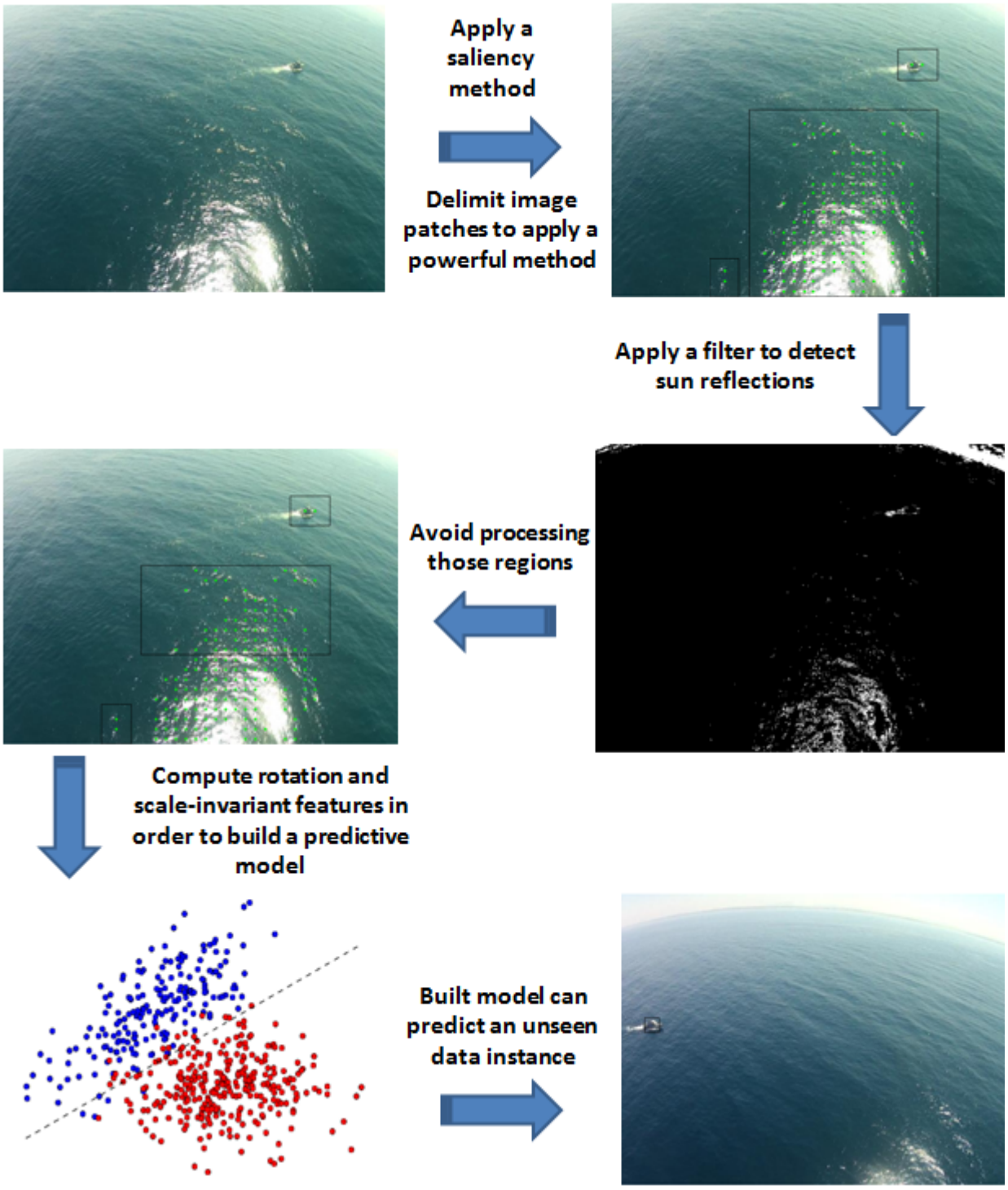

Figure 3.1: Workflow of the implemented approach. *Top* Original image; Saliency method applied to the original image in order to highlight the most conspicuous regions. *Middle* Method to filter sun reflections. *Bottom* Implementation of rotation-invariant HOG descriptors in order to get the desired features to train a classifier for vessel detection; using the output model, we are now able to detect an unseen data instance.

3.1 Preprocessing Stages

3.1.1 Saliency by Contrast

This section is dedicated to saliency methods which were applied in order to highlight the most conspicuous regions of a given image. As we stated previously, the reason of using such methods its due to the computational cost of our final solution. To deal with the problem formulated, we will need to develop complex algorithms in order to overcome several identified problems. Therefore, using saliency methods, we can restrict the areas of an image where we will run a more powerful algorithm.

After the implementation of the methods studied (as it was referred on chapter 2), we found they were taking approximately $1s$ to compute the most relevant regions of a 720x405 image (note that the resolution of the images captured by the camera is full HD, but we agreed to treat the images using the resolution which is more convenient, provided that we keep the aspect ratio). Since our goal is to process, at least, one frame per second, we needed to reduce significantly the time consumed by this step.

Focusing on our problem, first we compute the magnitude of the gradient for each pixel of a given image, $||d(r,c)||$ (where r and c represent the row and column of that pixel, respectively), using the Sobel operator. Then, we compute the sum of the gradient magnitude for the pixels belonging to a specific rectangular region, which we call $window_{contrast}$, as

$$window_{contrast} = \sum_{sR}^{eR} \sum_{sC}^{eC} ||d(r,c)||, \tag{3.1}$$

where sR, eR, sC and eC are the starting/ending rows/columns which define the rectangular region, respectively. In our approach, we split the image into square blocks (16x16 pixels), assigning to each one the result of the window contrast belonging to that block. In Fig. 3.2, we are marking the centroids of the blocks whose value is greater than a threshold, defined empirically. In order to decide the regions to be further processed, we compute a binary image, of size equal to the number of blocks, in which we assign the value 1 in the position corresponding to the block whose centroid was marked. Then, we perform a mask operation, an image dilation (with a square mask of 2 pixels radius), followed by the computation of the CCs based on an 8-Connected Neighborhood. Afterwards, some properties of those CCs are computed such as the area, bounding boxes and centroids. As it is possible to check in the figure below, CCs that have an area above a certain threshold are skipped. Obviously, this can be a problem considering a situation in which a vessel is too small. However, this threshold can be a parameter which can be adjusted by the user taking into account the current situation. In Algorithm 1, it is possible to visualize the workflow of this method.

This technique allows to process each frame at a rate of $100ms$, on average, which is considerably faster than the method proposed by Harel et al. ([15]). Additionally, the results achieved were really similar in this problem.

In SEAGULL project, other algorithms were developed to face this problem (a comparison against them is presented in section 5.5). Since this preprocessing step was considered useful as a first stage for all the algorithms, we had to think about how to perform it quickly for all methods. Hence, we propose using an integral image (popularized by Viola & Jones algorithm, [31]) in order to simplify the computation over the pixels, since it allows to

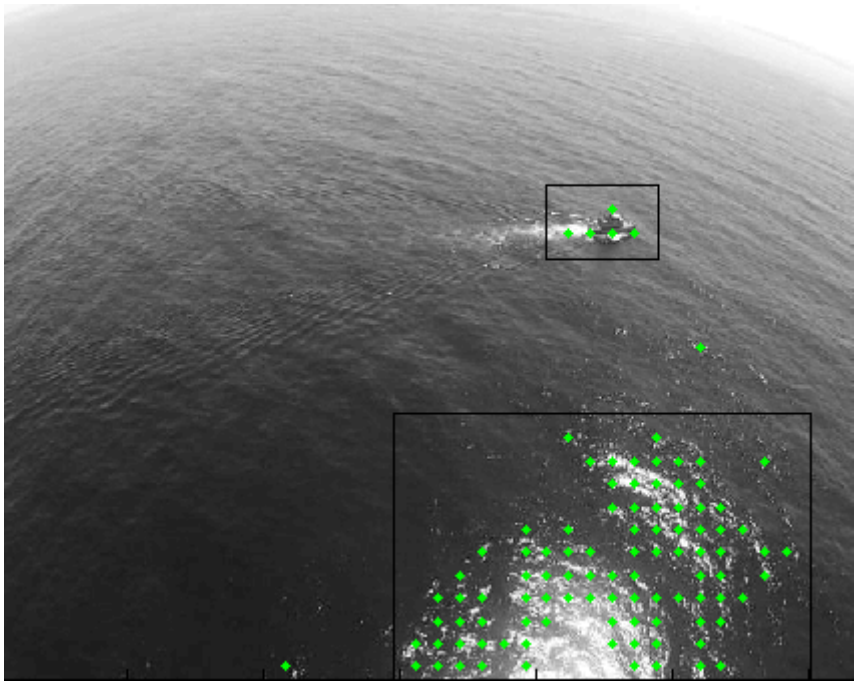

Figure 3.2: Preprocessing step to delimit areas of the image where it will be applied a more powerful algorithm.

Algorithm 1 Saliency by Contrast

Input: image I, $thresh_{sal}$, $block_{size}$
Output: $binary_{matrix}$

//Step 1: compute the image gradient magnitude
//Step 2: split I into square blocks of size $block_{size}$x$block_{size}$ *and initialize a* $binary_{matrix}$*, with zeros, of size:* $(\frac{I_{rows}}{block_{size}}, \frac{I_{columns}}{block_{size}})$
for each block **do**
    compute the $window_{contrast}$ belonging to that block
    if $window_{contrast} > thresh_{sal}$ **then**
        update to one the position of $binary_{matrix}$ corresponding to that block
    end if
end for
//Step 3: perform an image dilation on $binary_{matrix}$
//Step 4: compute the CCs of $binary_{matrix}$ *based on a* 8*-Connected Neighborhood*

calculate summations over image subregions. Integral images facilitate summation of pixels and can be performed in constant time, regardless of the neighborhood size. The idea is to compute a single time the integral image of the gradient magnitude for every pixels of an image. Therefore, we could call Algorithm 1 for each method, varying the $block_{size}$ taking into account the needs of each one.

The integral image, J, is a padded version of the cummulative sum along the columns and lines of an intensity image I (eq. 3.2). Please refer to appendix A for further details about the computation of an integral image and a

simple example.

$$J(r,c) = \sum_{\substack{0<r'<r \\ 0<c'<c}} I(r',c') \tag{3.2}$$

This technique can be helpful since we can use the integral image J to quickly retrieve the sum of pixels over a rectangular region of an image I (in our case, I would be a matrix with the gradient magnitudes for each pixel).

3.1.2 Sun Reflections Detector

The saliency method explained above (section 3.1.1), when applied to the available images, highlight regions of boats and sun reflections, filtering the sky and ocean. As we stated previously, its goal is to delimit some areas of an image where it will be applied a more powerful algorithm. Since the computational cost is presented as one of the major limitations of our project, we thought that we could build an additional preprocessing step which could filter the regions of sun reflections given by the saliency method. Therefore, it was developed a fast algorithm to detect sun reflections, avoiding to waste time in those areas.

Hence, we performed some studies in order to understand how we could classify those events using the available images. First, we converted the RGB images to an Hue-Saturation-Value (HSV) color map. Looking to the images, we verified that the sun reflections have a particular level of saturation and brightness. Thus, we studied how the second and third component of an HSV image behave along several images, particularly the values belonging to the pixels of sun reflections. Using the image presented at left of Fig. 3.4, we plot a 2D histogram of the Saturation and Value components of that image. As it is possible to check in Fig. 3.3, there is a thin peak on the right side which represents the pixels belonging to sun reflections. This peak encapsulates pixels with low saturation values (between 0 and 0.1) and high values of brightness (between 0.9 and 1). In Fig. 3.4, we show the same image, but the pixels whose saturation and brightness belong to that range were assigned to zero. Therefore, by performing this thresholding operation, we can easily identify regions of sun reflections, avoiding its processing. Basically, we split the image into square blocks (16x16pixels) and check if there is a number of pixels, considered as belonging to sun reflections, which is greater than a specific percentage of the area of that block (in our case, we avoid to process a block whose number of sun reflections pixels was superior to 60% of the block area). It is obvious that this range of values is directly related with the atmospheric conditions, whereby it can also be assigned a parameter to be sent to the drone in order to adjust the threshold. Algorithm 2 shows the main steps of this preprocessing step.

3.2 Feature Extraction

This section focuses on HOG descriptors which allow one to describe particular features of the targets. First, it will be introduced the concept proposed by Dalal and Triggs ([7]) which is widely used on image description. Then, the rotation-invariant HOG descriptors ([22]) will be presented and carefully explained.

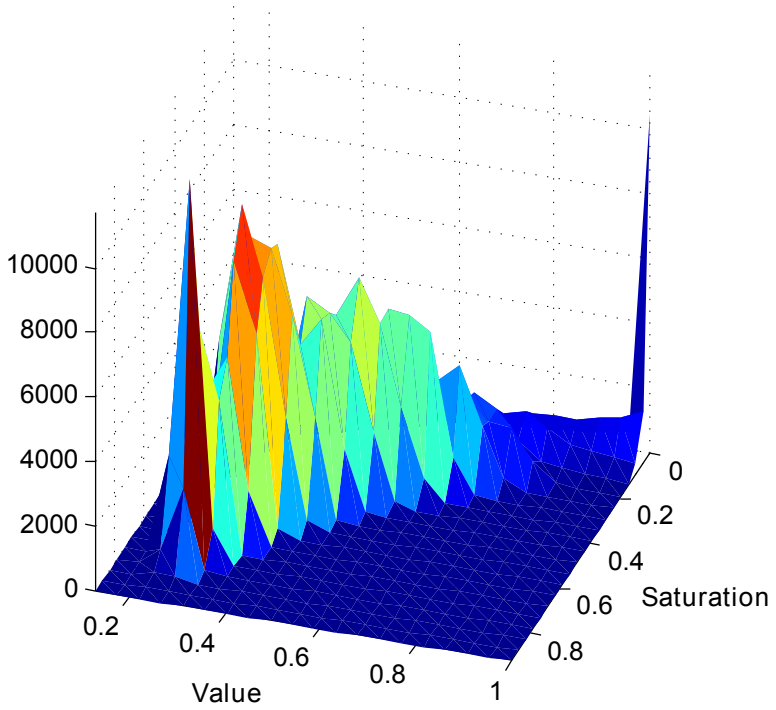

Figure 3.3: 2D histogram representing Saturation and Value components of the left image presented in Fig. 3.4.

Figure 3.4: *Left* Example of an image available in which it is possible to visualize a large sun reflection. *Middle* Thresholding of Saturation and Value components; it was assigned zero values (black) to the RGB components of pixels whose Saturation was lower than 0.05 and Value greater than 0.90. *Right* Thresholding of Saturation and Value components; it was assigned zero values (black) to the RGB components of pixels whose Saturation was lower than 0.10 and Value greater than 0.70 (in this case, we consider lower values for brightness than the ones we state after analyzing the histogram, since it is possible to make a cut until 0.70 as well as we could consider saturation values up to 0.30).

Algorithm 2 Sun Reflections Detector

Input: image I, $thresh_{SAT}$, $thresh_{VAL}$, $thresh_{area}$
Output: $binary_{matrix}$
Local Variables: $sunrefl_{count}$

//Step 1: covert I from RGB to HSV color space
//Step 2: split I into square blocks of size $block_{size}$x$block_{size}$ *and initialize a* $binary_{matrix}$*, with zeros, of size:* $(\frac{I_{rows}}{block_{size}}, \frac{I_{columns}}{block_{size}})$
for each block **do**
  for each pixel **do**
    $sunrefl_{count} = 0$
    if $pixel_{saturation} < thresh_{SAT}$ && $pixel_{brightness} > thresh_{VAL}$ **then**
      $sunrefl_{count}++$
    end if
  end for
  if $sunrefl_{count} < thresh_{area}$x$block_{area}$ **then**
    update to one the position of $binary_{matrix}$ corresponding to that block
  end if
end for
//Step 4: perform an image dilation on $binary_{matrix}$
//Step 5: compute the CCs of $binary_{matrix}$ *based on a* 8*-Connected Neighborhood*

3.2.1 Histograms of Oriented Gradients - Standard Approach

The standard approach proposed by Dalal and Triggs ([7]) have been widely used to solve a huge variety of problems in image processing. The main idea is to split a given image into equal cells. For each cell, we compute a discrete gradient histogram of the local image patch. Each cell is denoted a HOG cell. A quantity that describes certain image content is generally called a feature. An assembled feature vector that describes a region of multiple HOG cells is referred to as a HOG descriptor.

In Fig. 3.5, it is possible to visualize the workflow of this method using one image of the available sequences as an example. In this case, each bin has a range of $45°$, covering all possible gradient orientations of the image pixels (from $-180°$ to $180°$). Imagine the image is rotated, for instance, by $90°$. If a new discrete gradient histogram is computed for the corresponding HOG cell (as shown in Fig. 3.6), it is possible to observe an interesting behavior. When the underlying image rotates by an angle which is multiple of the range of the bins, which is the case, the new HOG feature can be obtained by a cyclic permutation. In this case, we have a shift of two units for each bin comparing with the preceding histogram, since the angle of rotation is two times the range of the bins. However, for rotations which are not multiple of the range of the bins, it is impossible to infer the same conclusion.

The point we want to prove here is that this strategy is clearly not rotation-invariant. To corroborate this fact, we will show other example where it is possible to check the lack of rotation-invariance. Consider now Fig. 3.7, in which the boat is in the center HOG cell. Some tests were performed, just rotating that image patch (the center of rotation is the center of the image), by angles which are not multiple of the range of the bins established, and analyzing the behavior of the gradient. As can be observed, there is no clear relation between the different histograms. The values of each bin are mixed from one histogram to another, so that this is clearly not a rotation-invariant approach. Note that, in this case, we are just analyzing the gradient of the image patch belonging to the center HOG cell after each rotation. Some pixels of the boat which are present in the initial patch will belong to

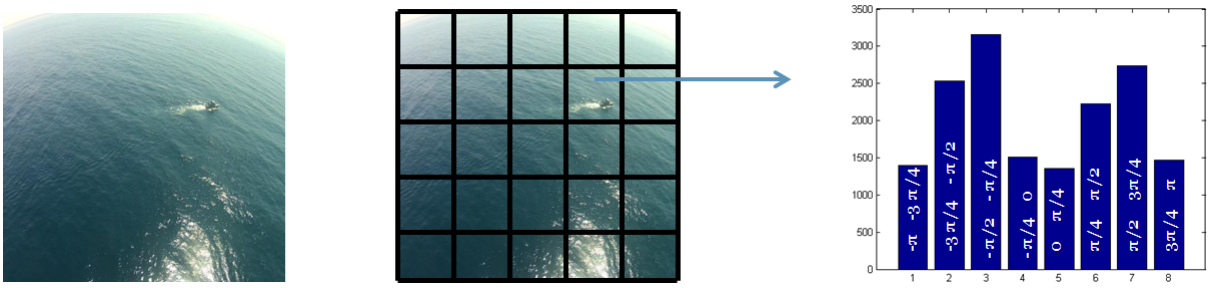

Figure 3.5: Workflow of the method proposed by Dalal and Triggs to study an image gradient: first, we split the image into HOG cells; then we compute a discrete gradient histogram for each image patch.

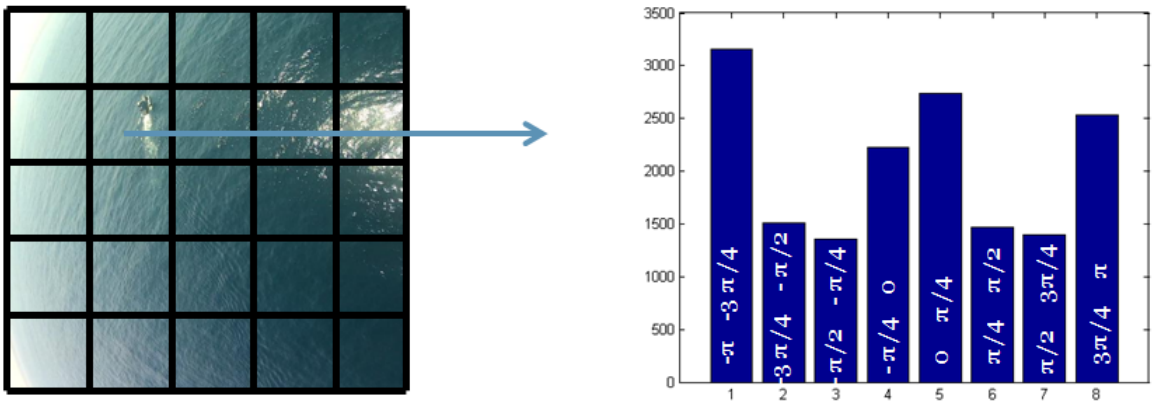

Figure 3.6: Discrete gradient histogram of the same image patch, when the image was rotated by 90°. As can be seen, the bins were shifted by 2 units comparing with the previous histogram. The first two bins were shifted to the end while the remaining ones were shifted to the left.

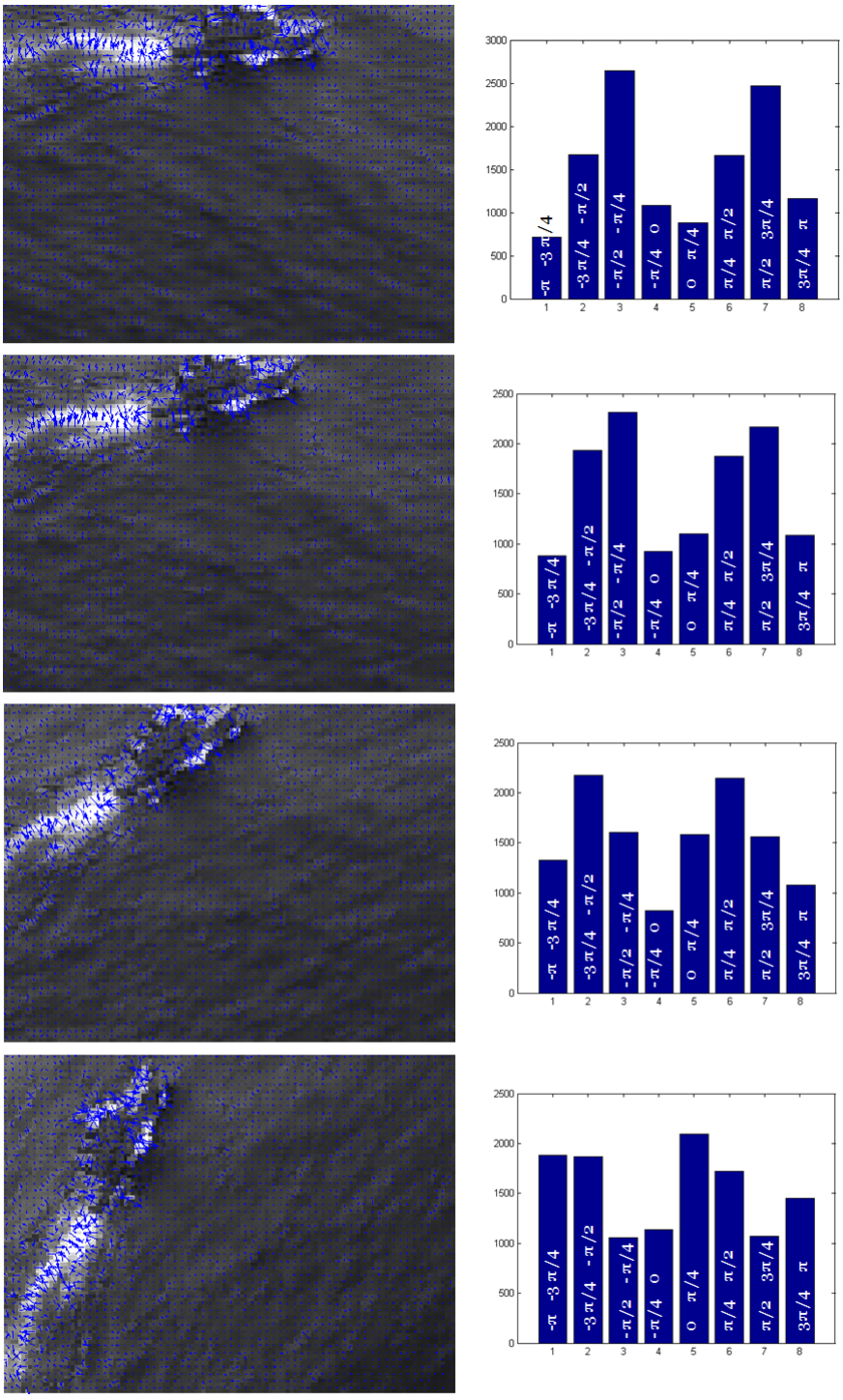

Figure 3.7: Lack of rotation-invariance on the method proposed by Dalal and Triggs ([7]). We applied several rotations to an image patch belonging to the center HOG cell. After each rotation, it is hard to see the relation between the different histograms.

other image patch after a rotation. Hence, on the right subfigures of Fig. 3.7 we are successively representing the orientations of the gradient for some different pixels regarding the ones before the rotation which demonstrates our point. Beyond that, the orientation of the gradient of a pixel which is in both images patches (before and after a rotation) changes, explaining the reason why our histograms simply do not shift between rotations.

As it was shown, discrete HOGs do not help to overcome one of the main requirements of our problem. The lack of rotation invariance is explained by the discretizations of the bins which prevents us from representing all possible orientations unambiguously. Obviously, the solution would be to represent HOGs in a continuous way, however this would be computationally prohibitive. The next section introduces a new concept to compute rotation-invariant HOG descriptors, using Fourier analysis to approximate a continuous representation.

3.2.2 Rotation-Invariant HOG Descriptors using Fourier Analysis in Polar Coordinates

This section focuses on the core of our solution. Liu et al. ([22]) proposed a method which allows one to gather rotation-invariant features which characterize the targets we want to detect. By implementing their method, the challenge of dealing with different perspectives of vessels is overcome. Liu et al. ([22]) built a descriptor composed by only rotation-invariant features which can be compared with each other using a standard classifier. Rotation-invariant features means that they are independent of the orientation of the described objects, so no further steering is required for their comparison.

3.2.2.1 Reasons of using a continuous approach

Most of the state-of-the-art approaches for image description use the standard HOGs. As it was stated in 3.2.1, discrete HOG features neglects some intrinsic properties of rotations, since the orientations in histograms are computed according to a fixed coordinate system. As proposed by Dalal and Triggs (3.2.1), a HOG cell is always represented by a discrete histogram. When the underlying image rotates by an angle which is multiple of the bins range, the new HOG feature can be obtained by a cyclic permutation. For the remaining rotations, the feature can only be approximated. Beyond that, this discrete approach prevent us from representing all possible gradient orientations. Hence, in order to cover the full range of gradient orientations, it is obvious that we need to represent histograms in a continuous way, avoiding bins discretization. However, it would be necessary infinitesimal bins, seriously overloading the final computational cost.

In order to overcome this problem, Liu et al. proposed to treat HOG cells as continuous functions, which could be represented by Fourier basis. The original information encoded by a HOG cell is a gradient density function of orientation, which can be represented by a continuous function, $h(\varphi)$, in the 2D case (φ represents the orientation). Usually, in standard approaches, the HOG features are computed in three steps: **gradient orientation binning**, **spatial aggregation** and **normalization**. Hence, in the following section, it will be explained how Liu et al. ([22]) compute rotation-invariant image descriptors by using the shift property of Fourier analysis.

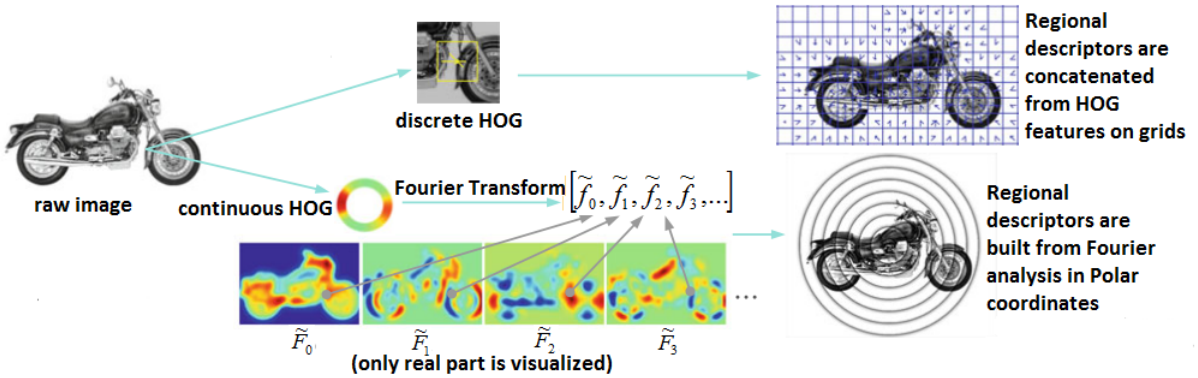

Figure 3.8: High-level comparison between the standard and rotation-invariant HOG descriptors *Top* Standard HOG descriptor computes discrete HOG features on multiple cells and concatenates them into a regional descriptor. *Bottom* The proposed method treats the HOG cells as continuous functions and use the Fourier basis in polar coordinates to represent them, then they can be easily embedded into a Fourier analysis framework to build a rotation-invariant descriptor (Adapted from [22])

3.2.2.2 Computing the Fourier Representation of Histograms of Oriented Gradients

In [22], Liu et al. proposed to create an orientation distribution function $h(\varphi)_{(x,y)}$ at each pixel (x, y). Consider the image gradient computed for one pixel as $d(x, y) \in \mathbb{R}^2$. In Fig. 3.9 it is possible to visualize the magnitude and orientation of the gradient for this pixel. Therefore, the distribution function for the referred pixel, which is oriented to $\varphi(d(x, y))$, should be an impulse function with value equal to its magnitude, $||d(x, y)||$:

$$h(\varphi)_{(x,y)} := ||d(x, y)||\delta(\varphi - \varphi(d(x, y))) \tag{3.3}$$

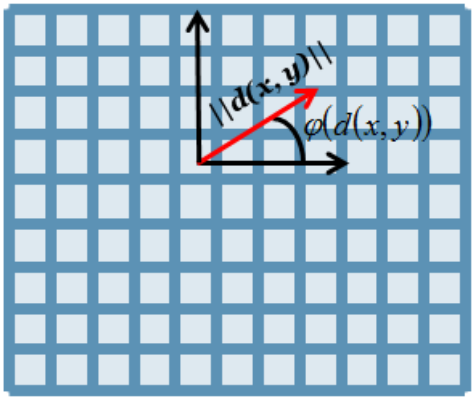

Figure 3.9: Magnitude and orientation of the image gradient represented for a single pixel.

Thus, using a Fourier basis for a rotation-friendly representation, the Fourier coefficients $\hat{f}_{m,(x,y)}$ of the orientation distribution function $h(\varphi)_{(x,y)}$ for each pixel (x, y) are

$$\hat{f}_{m,(x,y)} = \langle h(\varphi)_{(x,y)}, e^{im\varphi} \rangle = ||d(x, y)||e^{-im\varphi(d(x,y))}, m \in \mathbb{Z} \tag{3.4}$$

Since it is possible to compute infinite projected Fourier coefficients (infinite possibilities for m), this computation may look redundant as a representation for only one gradient. Limiting the maximal frequency degree $|m|$ that is considered for representing the density function provides the desired smoothing effect of the classical soft binning[2]. This is equivalent to a low-pass filtering in the orientation domain.

Now, it is important to understand the optimality of the Fourier Transform for rotation analysis. From eq. 3.4, the analysis equation, it is possible to infer the synthesis equation which states that a function on a circle can be expanded linearly in terms of Fourier basis functions as

$$h(\varphi)_{(x,y)} = \sum_{m=-\infty}^{\infty} \hat{f}_{m,(x,y)} e^{im\varphi} \tag{3.5}$$

Eq. 3.5 defines the Fourier series representation of a periodic signal $h(\varphi)_{(x,y)}$. It can be expressed as the sum of the product between the Fourier coefficients and these harmonically related complex exponentials. This signal has an interesting and simple rotation behavior as you can check in eq. 3.6. Considering that the function rotates by an angle α into $h(\varphi)'_{(x,y)}$, the Fourier coefficients transform by a simple multiplication as

$$h'(\varphi)_{(x,y)} = h(\varphi - \alpha)_{(x,y)} = \sum_{m} \hat{f}_{m,(x,y)} e^{im(\varphi-\alpha)} = \sum_{m} (\hat{f}_{m,(x,y)} e^{-im\alpha}) e^{im\varphi}, \tag{3.6}$$

which is the well known Shift Property of the Fourier Transform. From eq. 3.6, we conclude that for these special functions, **the rearrangement of the pixels caused by a rotation can be substituted by a pixel-wise multiplication of the Fourier coefficients with a single complex number**, which depends on the order of Fourier coefficients, m, and the angle of rotation α. Refer to appendix B for a concrete example.

Fig. 3.10 shows the differences between a discrete and a continuous representation. After a rotation applied to the image, the discrete representation changes irregularly. On the other hand, the continuous representation maintains its shape up to a cyclic shift. This means that we can use the analytical rotation $h'(\varphi) = h(\varphi - \alpha_g)$, avoiding quantization errors. Besides that, by projecting h onto the Fourier basis, the obtained Fourier representation (illustrated in the middle of Fig. 3.8) transforms with a simple multiplication under rotations (according to the shift property of Fourier Transform). Hence, we avoid discretization artifacts and we can map, in a continuous way, the image to the feature space.

The second step, the spatial aggregation, can be implemented by spatial convolutions on the Fourier coefficients using, for instance, an isotropic triangle or a Gaussian kernel. Finally, the local contrast normalization can be also implemented applying spatial convolutions. Note that, performing filtering techniques with isotropic kernels is rotation-invariant, in contrast to some grid-block based operations as in standard approaches.

Considering two convolution kernels, $K_1 : \mathbb{R}^2 \to \mathbb{R}$ for the spatial aggregation and $K_2 : \mathbb{R}^2 \to \mathbb{R}$ for the local normalization (based on gradient energy), D as the gradient field for the whole image and $\hat{F}_m : \mathbb{R}^2 \to \mathbb{C}$ as the densely computed Fourier coefficients $\hat{f}_m$ for every pixel belonging to an image, from eq. 3.4, we can compute

[2] In the standard techniques, a triangular interpolation is used to soften the orientation quantization for robustness, performing a convolution with a 1D triangle kernel. Liu et al. found, empirically, that this smoothing is redundant and it did not improve performance.

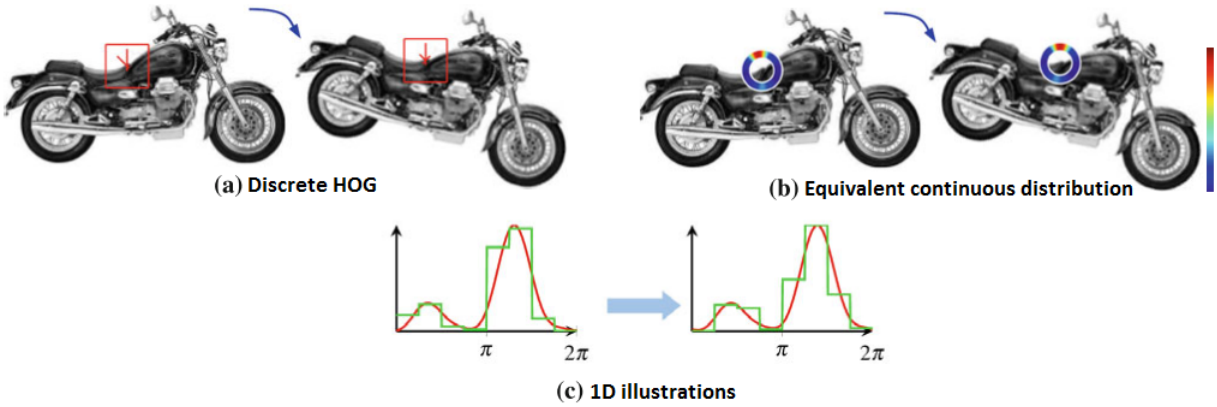

Figure 3.10: Discrete vs Continuous HOG. Representation for a specific patch, before and after a rotation of 15 °. **(a)** Discrete HOG. **(b)** A continuous angular signal representing the same distribution. **(c)** 1D illustration of a discrete histogram (*green*) and its corresponding continuous representation (*red*). (Adapted from [22])

the *Fourier HOG field* (its degree-m component) as

$$\tilde{F}_m = \frac{\hat{F}_m * K_1}{\sqrt{||D||^2 * K_2}}. \tag{3.7}$$

In Fig. 3.8, it is possible to visualize some examples of $\tilde{F}_m$. Fig. 3.11 shows an high-level schematic of what we are doing. Through eq. 3.4, we compute the Fourier coefficients for each pixel of the image. In our case, we are limiting the maximal frequency degree, m, thus we only compute the first five coefficients (from $m = 0$ until $m = 4$). This provides the desired soft-binning effect in orientation. Liu et al. came to the conclusion that limiting $|m| \leq 4$ is sufficient, since these few low-frequency Fourier coefficients encode the useful information.

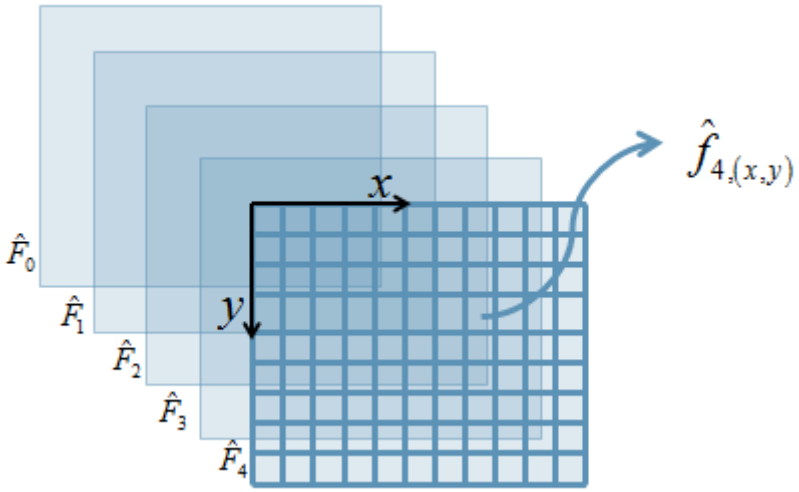

Figure 3.11: High-level schematic: we compute Fourier coefficients of different orders for each pixel of the image (in the illustration, 5). $\hat{F}_m$ represents all the Fourier coefficients of order m of the image. Here, $\hat{f}_{4,(x,y)}$ represents the Fourier coefficient of pixel (x, y) with order 4.

To finalize the explanation of this method, it is important to understand how we can compute rotation-invariant features from the computed Fourier HOG field, $\tilde{F}_m$, taking into account the shift property of the Fourier analysis. In the next section, we will start to recall the referred property in order to understand how it is possible to take advantage of it to create rotation-invariant HOG descriptors.

3.2.2.3 From the HOG Cells Representation to the Computation of Regional Features

By now, we are just representing, in a continuous way, HOG cells (in this approach are represented by the Fourier HOG field) which describe an image only locally. Hence, we need to compute regional descriptors to describe large regions, on the densely computed HOG cells. Basically, we want to compute higher-level features by filtering on the computed Fourier HOG field. This is equivalent to other techniques as SIFT or sliding window techniques, where histograms at neighboring cells, with similar response to an applied filter, are concatenated into a regional descriptor.

There is more than one way to represent a 2D location, such as Cartesian coordinates or polar coordinates. In our case, since we need to develop a rotation-invariant strategy, circular windows, represented by polar coordinates, fit better to this problem instead of the traditional square windows of standard approaches, which are not rotation-invariant (as shown in section 3.2.1). Polar coordinates ($[r, \varphi] : r = ||x||, \varphi = \Phi(x) = atan2(y, x) \in [0, 2\pi)$) are ideal for analyzing 2D functions concerning rotations, since they separate the radial part, which is naturally invariant to rotations, from the angular part, which is involved in rotations and needs to be carefully chosen. Due to the stated reasons, Liu et al. ([22]) argued that an ideal basis should take a separable form as $U(r, \varphi) = P(r)\Psi(\varphi)$. So, while the radial part $P(r)$ can be chosen according to the context, the optimal choice for the angular part is the Fourier basis $\Psi_m(\varphi) = e^{im\varphi}$, where m is an integer. Hence, it is possible to compute infinite basis functions $[\Psi_0, \Psi_1, \Psi_2, ...]$, which will form harmonics on a circle (circular harmonics).

A basis function in the form $U_{j,k} = P(r_j)e^{ik\theta}$ has a nice and simple rotation behavior as it was discussed previously. Now, it is just necessary to choose a proper radial basis, $P(r_j)$, in order to build 2D basis which can describe the Fourier HOG field. Liu et al. chose to sample in the radial direction, which leads to a 2D basis functions of the form

$$U_{j,k}(r, \theta) = \delta(r - r_j)e^{ik\theta} \tag{3.8}$$

where $j \in \mathbb{N}_0$, $k \in \mathbb{Z}$. Since the computation of the HOG field includes a convolution on the Fourier coefficients $\hat{F}_m$ for a smoothed spatial aggregation (eq. 3.7), a proper down-sampling in radial direction, according to the scale of the smoothing kernel K_1, can keep all the information (K_1 is a triangle kernel of half-width equal to 6 pixels while $U_{j,k}$ will have different radial profiles with half-width also equal to 6 pixels and sampled at 4 radii - for further details, refer to section 5.3).

Actually, the basis function we will use to perform the convolutions with the Fourier HOG field in order to compute regional descriptors is not so simple as presented in eq. 3.8. What the authors suggest is to merge this step with the smoothing one (convolution between K_1 and $\hat{F}_m$), allowing to compute just one convolution instead of two. So, we need to combine K_1 and $U_{j,k}$, which leads to a 2D basis function with a triangular radial profile and a Fourier basis as

$$U_{j,k}(r, \varphi) = \Lambda(r - r_j, \sigma)e^{ik\theta} \tag{3.9}$$

where Λ is a triangular function of width 2σ defined as $\Lambda(x, \sigma) = max(\frac{\sigma - |x|}{\sigma}, 0)$. You can check the basis functions used for regional descriptions, by visualizing Fig. 3.12, which include smoothing both in radial and angular direction. As they did when computing the Fourier coefficients, they only compute five different orders for the basis functions (from $k = 0$ to $k = 4$).

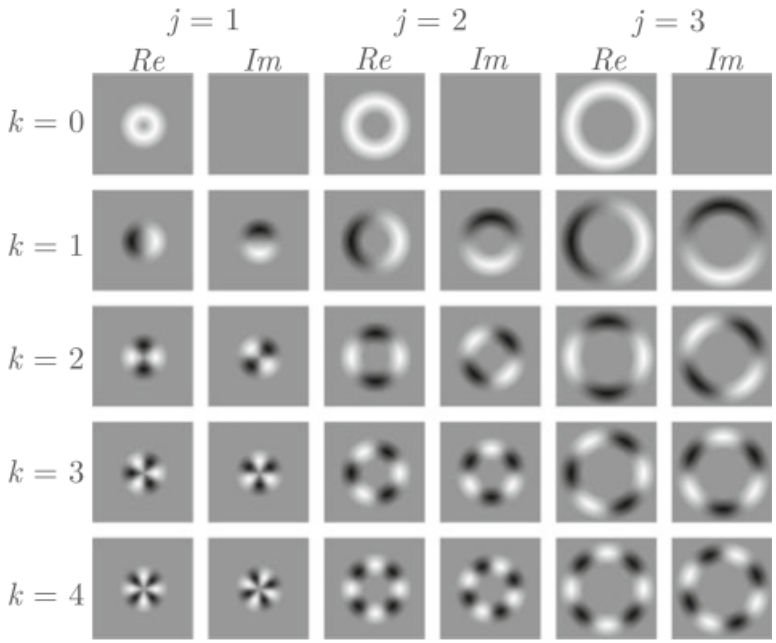

Figure 3.12: Some examples of filters used to generate regional descriptors. These kernels have the form $U_{j,k} = P(r_j)e^{ik\theta}$. Refer to section 5.3, to check the values of the radii used. (Adapted from [22])

Hence, the convolution between such a basis function $U_{j,k}$ and a component of the Fourier HOG field, $\tilde{F}_m$, generates a feature which describes the configuration of HOG features in the region covered by $U_{j,k}$. In the next section, it is explained how we get rotation-invariant features from this regional features we are computing by performing a convolution between the kernels (represented in Fig. 3.12) and the computed Fourier HOG field (refer to eq. 3.7).

3.2.2.4 Generation of Rotation-Invariant HOG Descriptors

Fig. 3.13 shows how we compute regional features, by sliding the 15 different kernels shown in Fig. 3.12 over the image (represented as the degree-m component of Fourier coefficients for each pixel, $\tilde{F}_m$). So, when we convolve a kernel with the computed Fourier coefficients, we are just doing multiplications between the kernel values and the Fourier coefficients of the pixels which are being covered by that kernel (marked red in Fig. 3.13). All the Fourier coefficients will be covered by the several kernels we use, since we are sliding them over the densely computed Fourier representation of $\hat{f}_m$, $\hat{F}_m$. This is also shown in eq. 3.10, where we consider the convolution between those basis functions and the Fourier coefficients of the pixels covered by $U_{j,k}$. Note that since $\tilde{F}_m$ (3.7) is computed from $\hat{F}_m$ just with isotropic filtering, the conclusions we will take for $\hat{F}_m$ are also valid to $\tilde{F}_m$.

$$U_{j,k} * \hat{F}_m = \sum_{r,\theta \in [0,2\pi)} P(r_j) e^{ik\theta} \hat{f}_m \tag{3.10}$$

Under a rotation, we will need to recompute the new Fourier coefficients for the new image. Recall that, the Shift Property of the Fourier transform (3.6) tells that these new Fourier coefficients are the ones computed before the rotation multiplied by a complex factor which depends on the order of the computed Fourier coefficients and the angle α, as we can check in eq. 3.11.

$$U_{j,k} * \hat{F}'_m = \sum_{r,\theta \in [0,2\pi)} P(r_j) e^{ik\theta} \hat{f}'_m = \sum_{r,\theta} P(r_j) e^{ik\theta} \hat{f}_m e^{-im\alpha} \tag{3.11}$$

Therefore, considering the case in which the kernel has the same order of the Fourier coefficients, $k = m$, we can simplify the above equation, as we have here:

$$U_{j,k} * \hat{F}'_m = \sum_{r,\theta} P(r_j) \hat{f}_m e^{ik(\theta-\alpha)} = \sum_{r,\theta'} P(r_j) \hat{f}_m e^{ik\theta'} \tag{3.12}$$

which is just the same we have before the rotation (eq. 3.10) if we apply a change of variable $\theta' = \theta - \alpha$. Since $\theta \in [0, 2\pi)$, as you can see in Fig. 3.13, it does not matter if you start in θ or in θ' because you will sum over the same values, since it is periodic. We are taking the sum of the product between the Fourier coefficients and the kernel values it is covering. Therefore, ensuring that the order of the kernel we are using to slide is the same of the computed Fourier coefficients ($k = m$), we are taking rotation-invariant features.

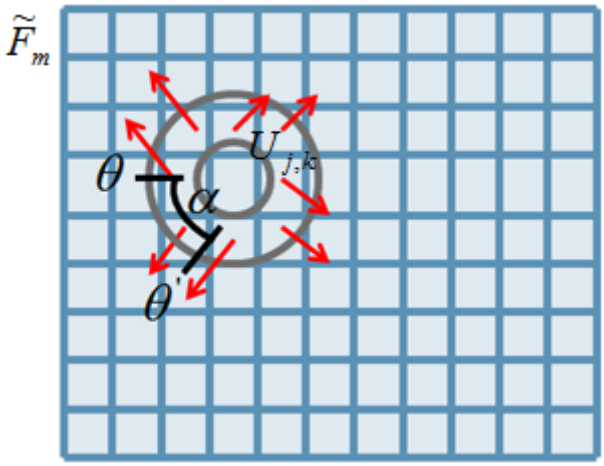

Figure 3.13: To compute higher-level features, we slide the basis functions shown in Fig. 3.12 over the computed Fourier HOG field. The basis functions are fixed in the whole image analysis process (they are independent of the image rotations).

The features computed ensuring the condition defined above are what Liu et al. call as naturally rotation-invariant features. However, it is possible to create many more invariant features from coupling two of the filtering results as in eq. 3.13. These features, just by themselves, are not rotation-invariant but they can be coupled as a rotation-invariant feature as long as their rotation order is the same. Rotation order of a filtering result is the differ-

ence between the order of the kernel used, $U_{j,k}$, and the order of the computed Fourier coefficients, $\tilde{F}_m$ (thus, the rotation order will be $k-m$). Therefore, instead of only taking the magnitude of the expansion coefficients (which are naturally rotation-invariant), this coupling provides the possibility to create many more invariant features.

$$(U_{j_1,k_1} * \tilde{F}_{m_1})\overline{(U_{j_2,k_2} * \tilde{F}_{m_2})}, \forall k_1 - m_1 = k_2 - m_2. \tag{3.13}$$

Now, it is important to prove that if we ensure that the rotation orders of the filtering results are the same, we are also computing rotation-invariant features. Hence, as we did for the previous case, first we present the coupling of two filtering results before a rotation:

$$(U_{j_1,k_1} * \tilde{F}_{m_1})\overline{(U_{j_2,k_2} * \tilde{F}_{m_2})} = (\sum_{r_1,\theta_1} P(r_1)e^{ik_1\theta_1}\hat{f}_{m_1})(\sum_{r_2,\theta_2} P(r_2)e^{-ik_2\theta_2}\overline{\hat{f}_{m_2}}) \tag{3.14}$$

After a rotation of an angle α, taking into account the Shift Property of Fourier Transform, we can write:

$$(U_{j_1,k_1} * \tilde{F}'_{m_1})\overline{(U_{j_2,k_2} * \tilde{F}'_{m_2})} = \sum_{r_1,\theta_1} P(r_1)e^{ik_1\theta_1}\hat{f}_{m_1}e^{-im_1\alpha}\sum_{r_2,\theta_2} P(r_2)e^{-ik_2\theta_2}\overline{\hat{f}_{m_2}}e^{im_2\alpha} \tag{3.15}$$

Thus, enforcing that $k_1 - m_1 = k_2 - m_2 \Leftrightarrow m_1 - m_2 = k_1 - k_2$, we have:

$$\begin{aligned}(U_{j_1,k_1} * \tilde{F}'_{m_1})\overline{(U_{j_2,k_2} * \tilde{F}'_{m_2})} = \sum_{r_1,\theta_1}\sum_{r_2,\theta_2} P(r_1)\hat{f}_{m_1}P(r_2)\overline{\hat{f}_{m_2}}e^{i\alpha(m_2-m_1)}e^{i(k_1\theta_1-k_2\theta_2)} = \\ \sum_{r_1,\theta_1}\sum_{r_2,\theta_2} P(r_1)\hat{f}_{m_1}P(r_2)\overline{\hat{f}_{m_2}}e^{i[k_1(\theta_1-\alpha)-k_2(\theta_2-\alpha)]}\end{aligned} \tag{3.16}$$

Again, applying a change of variable like $\theta'_1 = \theta_1 - \alpha$ and $\theta'_2 = \theta_2 - \alpha$, we can achieve the same result we had before the rotation:

$$(U_{j_1,k_1} * \tilde{F}'_{m_1})\overline{(U_{j_2,k_2} * \tilde{F}'_{m_2})} = \sum_{r_1,\theta'_1} P(r_1)e^{ik_1\theta'_1}\hat{f}_{m_1}\sum_{r_2,\theta'_2} P(r_2)e^{-ik_2\theta'_2}\overline{\hat{f}_{m_2}} \tag{3.17}$$

Doing the same reasoning as we did for the first case, we can conclude that, in this case, we are also taking rotation-invariant features, since it does not matter to start in θ_1 instead of θ'_1 or θ_2 instead of θ'_2, because we will sum over the same values in both cases since $\theta_1, \theta_2 \in [0, 2\pi)$.

In Algorithm 3 (adapted from Liu et al., [22]), it is possible to see the pseudo code of the implemented algorithm. A flow diagram is also provided in Fig. 3.14.

3.3 Classifier Design

This section explains how the classifier was designed. The data labelling, the chosen classifier, the training process and performance evaluation as well as how the features were selected in order to achieve scale tolerance within a reasonable range, will be explained in the following sections.

Algorithm 3 2D Fourier HOG Descriptor

Input: image *I*
Output: rotation-invariant feature vector field **C: x** $\mapsto$ **C(x)**

//Step 1: compute Fourier HOG (eq. 3.7)
$\mathbf{D} = \nabla I$ *//compute gradient*
for $m = 0 : m_{max}$ **do**
  $\hat{F}_m(x) = ||D(x)||e^{-im\varphi(D(x))}$ *//$\varphi(D(x))$ is the gradient orientation*
  $\tilde{F}_m = \hat{F}_m / \sqrt{||D||^2 * K}$ *//normalization, K is a smoothing convolution kernel*
end for

//Step 2: compute regional features
$i = 0$
for all $\tilde{F}_m$ **do**
  for all basis function $U_{j,k}$ (e.g., the basis in Fig. 3.12) **do**
    $B_i = U_{j,k} * \tilde{F}_m$ *//regional feature*
    $m_i = k - m$ *//its rotation order*
    $i = i + 1$
  end for
end for

//Step 3: generate final rotation-invariant features
$\mathbf{C} = \emptyset$
for all B_i **do**
  if $m_i = 0$ **then**
    append B_i to **C** *//when the order of $U_{j,k}$ is equal to $\tilde{F}_m$*
  else
    append $||B_i||$ to **C** *//we just take the magnitude which is naturally rotation-invariant*
  end if
end for
//More features can be generated by coupling (eq. 3.13)
for all pairs of features B_i and $B_{i'}$ **do**
  if $m_i = m_{i'}$ and $m_i \neq 0$ **then**
    append $\bar{B}_i B_{i'}$ to **C**
  end if
end for

3.3.1 Data Labelling

Concerning the classification task, the first step was data labelling. The labels associated with a data instance determine if that instance is a vessel or not ([5]). In our problem, as in the majority of the cases, the labelling was done manually. For each image of the available videos, we stored the top left coordinate, the width and height of a Bounding Box (BB) containing the boat, in a *.txt* file, which will compose our Ground Truth (GT). We would like to highlight that the labelled data is not fully representative of all vessels we want to detect, since we are constrained to the videos which are available. Since the videos provided just contain boats moving or stopped, the data instances labelled are only representative of the available vessel types. Until now, we labelled almost 10000 images. However, to build predictive models to detect other kind of targets as we proposed, such as shipwrecked, oil spills, lifeboats, among other situations which may happen, we will need more sample videos which contain those objects. As negative examples, we consider everything else in the background: features from sea, sky and

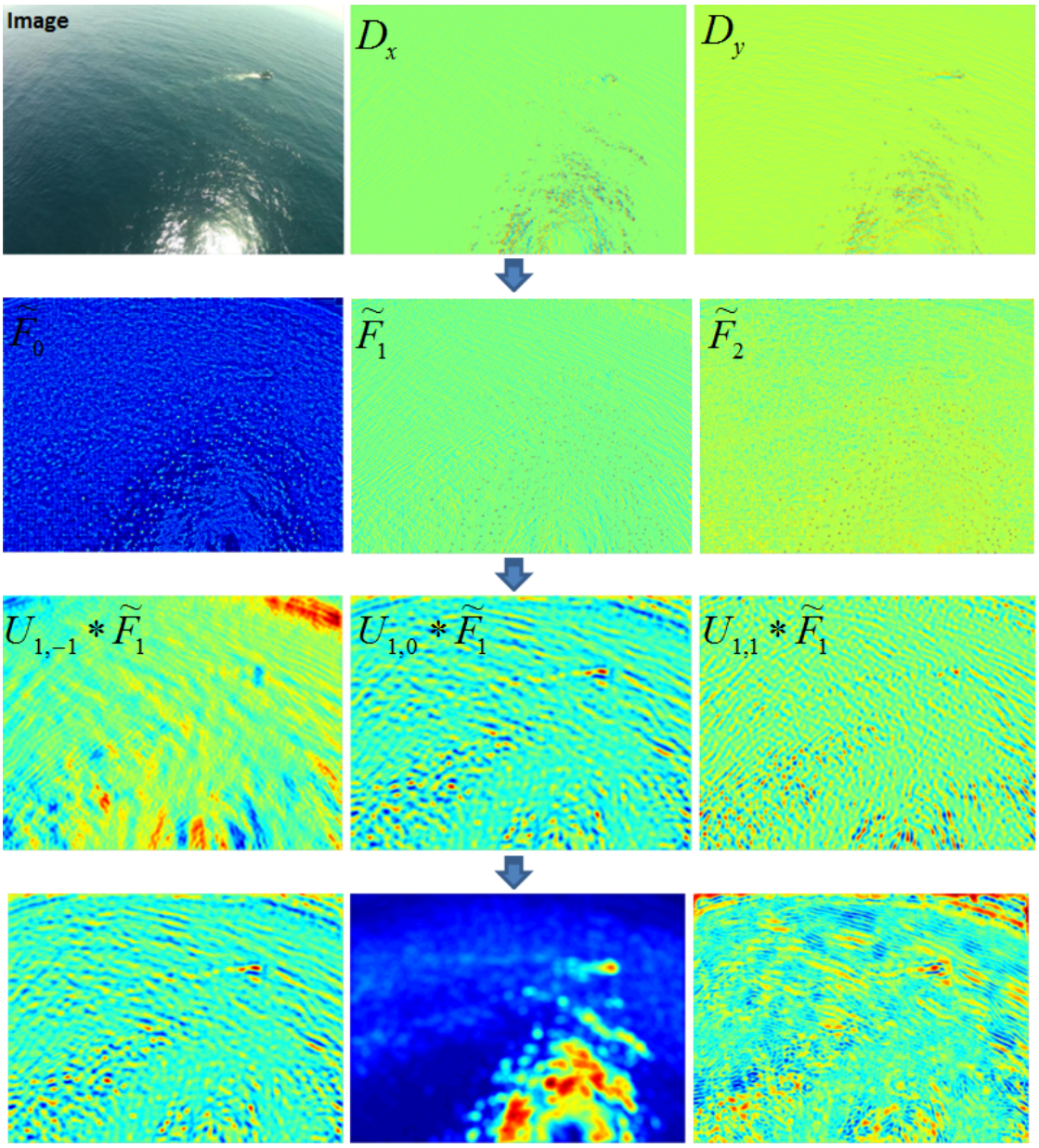

Figure 3.14: Flow diagram showing the main steps of Liu et al. approach. First, we take an image and its gradient is computed. Then, it is computed the Fourier HOG field from the gradient using equations 3.4 and 3.7 (here, for $m = 1, 2$, only the real part is represented). After that, the regional descriptors are computed by convolutions with circular harmonic basis, $U_{j,k}$ (represented in Fig. 3.12). Finally, rotation-invariant features are generated: *Bottom Left* some features are naturally rotation-invariant; *Bottom Middle* for the remaining ones, it is taken the magnitude; *Bottom Right* it is possible to generate many more rotation-invariant features using the couple operation in eq. 3.13. (Adapted from [22])

sun reflections.

3.3.2 Classifier Choice

Classification techniques can operate in one of three modes: *supervised*, *semi-supervised* or *unsupervised*. Since it is available a training data set with several labelled instances for positive and negative classes, we decided to use the supervised mode. Therefore, the idea is to build a predictive model which, for an unseen data instance, determines which class it belongs to. We performed linear classification, running a linear SVM using the LIBLINEAR library (Fan et al., [12]) in order to fairly compare the method proposed by Liu et al. ([22]) against our solution (composed by their method preceded by pre-processing stages) since it was the procedure they used to train their model.

3.3.3 Training and Evaluation Methodology

The workflow of the learning process consisted in three steps. Initially, we selected the positive samples as the features which are inside the BB containing the GT. The BB size was enlarged by 2 pixels so that a small region around the vessel is also used to describe the pattern. Regarding the selection of negative samples, first we randomly picked features which were outside the BB (approximately 30 samples per image). However, we found that this way of taking negative samples is not the best, since we may not sample a balanced number of the different types of negatives (sun reflections, ocean and sky). Therefore, we chose them manually in order to get negative samples which fully represents all events which can be considered as background. Here, we performed bootstrapping along several runs by adding features belonging to images patches marked as false positives to the set of negative samples, slightly improving the final results. We also normalized each feature dimension into the range $[-1, 1]$ in order to work with the linear SVM.

In order to evaluate the predictive model, several tests were performed using the available videos (refer to section 5.1 for further details). We splitted the available sequences in training and testing sets. Some combinations for the training set were tried and, then, tested in other videos, in order to assess the generalization ability of our classifier regarding two aspects: detect the targets in different scales and different types of vessels with which it was trained.

As it was stated, the model was trained using features of the samples belonging to the training set and, then, performed the detection on the test data. Each instance (pixel of the feature matrix - computed by $\tilde{F}_m * U_{j,k}$) of the training set contains one target value (the class label) and several attributes (the features, in our case we compute 233 features per pixel). The goal of a SVM is to produce a model, based on the training data, which predicts the target values of the test set given only the test data attributes. In Fig. 3.15, it is possible to observe a simple example in which the training data are linearly separable (optimal case), thus we can select two hyperplanes in a way that they separate the data and there are no points between them. Therefore, the main goal is to maximize their distance, since the larger the margin the lower the generalization error of the classifier. As output of the linear SVM, we get two variables, w and b, which are the normal vector to the hyperplane and the bias, respectively.

Considering that in this figure the training data is composed by the real vector x_i (corresponding to the pixel i) with an associated class $y_i \in \{-1, 1\}$, the maximum-margin hyperplane can be described as the set of points x satisfying $w \cdot x - b = 0$. However, in our case it is not so straightforward (see a possible example on Fig. 3.16). Hence, instead of having a score greater/lower than $1/-1$ to positive/negative matches and a clear gap between them, we will have a decision threshold which is susceptible to errors. The trick will be how we adjust it.

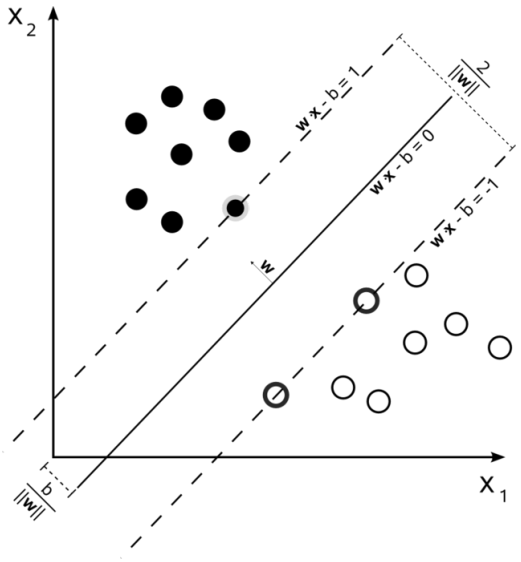

Figure 3.15: Maximum-margin hyperplane and margins for a linear SVM trained with samples from two classes ($[-1, 1]$). Samples on the margin are called the support vectors. (Adapted from [1])

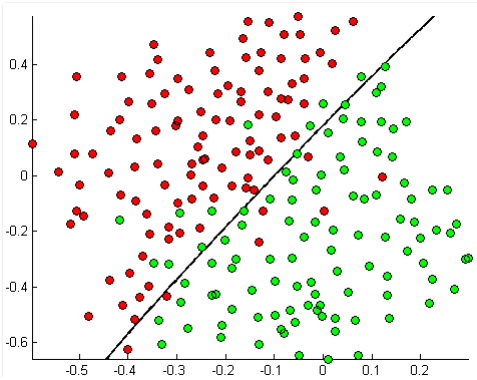

Figure 3.16: An example of data which can not be fully separated linearly. However it is possible to define a reasonable decision threshold, using a linear classifier.

Thus, we apply the trained classifier at every position of the image, computing a matrix of the same size of the image with a score associated to each pixel as:

$$score = w \cdot x - b \tag{3.18}$$

Then, we create a binary image with the same size of the score matrix in which the locations of the regional maxima are identified. In other words, we set to 1 the pixels which are a regional maximum while the others are set to 0. The way we check if some pixel is a regional maximum is based on 8-Connected Neighborhood, as illustrated in Fig. 3.17. After that, an image dilation with a square mask of 2x2 pixels is performed. Then, we compute the CCs as well as some of their properties as the centroid and the BB (basically, the top left and bottom right coordinates are (x_{min}, y_{min}) and (x_{max}, y_{max}) of the pixels belonging to that CC). As in the normal sliding window approaches, we apply Non-Maximum Suppression (NMS). In our case, we look for BBs which have overlap (it is sufficient that they just touch themselves), eliminating the one which has the lower score associated. The output is an array in which we store, for each image, the BBs coordinates of all objects detected, as well as their respective score. Therefore, it is possible to adjust a threshold in order to have many (low threshold) or few (high threshold) detections, which will depend on the user's needs.

```
score_matrix =

    10    10    10    10    10    10    10    10    10    10
    10    22    22    22    10    10    44    10    10    10
    10    22    22    22    10    10    10    45    10    10
    10    22    22    22    10    10    10    10    44    10
    10    10    10    10    10    10    10    10    10    10
    10    10    10    10    10    33    33    33    10    10
    10    10    10    10    10    33    33    33    10    10
    10    10    10    10    10    33    33    33    10    10
    10    10    10    10    10    10    10    10    10    10
    10    10    10    10    10    10    10    10    10    10

regional_max =

     0     0     0     0     0     0     0     0     0     0
     0     1     1     1     0     0     0     0     0     0
     0     1     1     1     0     0     0     1     0     0
     0     1     1     1     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     1     1     1     0     0
     0     0     0     0     0     1     1     1     0     0
     0     0     0     0     0     1     1     1     0     0
     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0
```

Figure 3.17: Regional maximum computation of a score matrix. For each pixel, it analyzes all the 8 neighbors and check if it has a higher or equal score to its neighbors. If so, it marks that pixel as 1, otherwise, it is marked as 0. Note that this is an illustrative example to better understand how it is computed (in our case, the scores are normalized in the range $[-2, 2]$).

3.3.4 Scale Tolerance

The need of scale tolerance was stated as one of the major requirements to fulfil when formulating the problem in chapter 1. The different flight altitudes (although the plane has a limited range of altitude between $150m$ and $300m$) as well as the huge variety of boats we may find in this environment prove to be problematic.

The computation of the features is done performing a filtering technique, by sliding several kernels with different orders and scales (check Fig. 3.12) over the computed Fourier coefficients of a given image, as sketched in Fig. 3.13. We are using three different scales for the kernels (refer to section 5.3): 23x23, 35x35 and 47x47 pixels. In Fig. 3.18, we are representing different models of boats which have different sizes. Hence, we slide several kernels over the Fourier coefficients of the image. For the big boat, obviously the kernel which will gather more information will be the larger. The same reasoning can be done for the remaining cases. However, the smaller kernel can get features of certain parts of the biggest boat which is also useful. Using those sizes for the kernels, we are convering the range between 20 and 50 pixels, approximately.

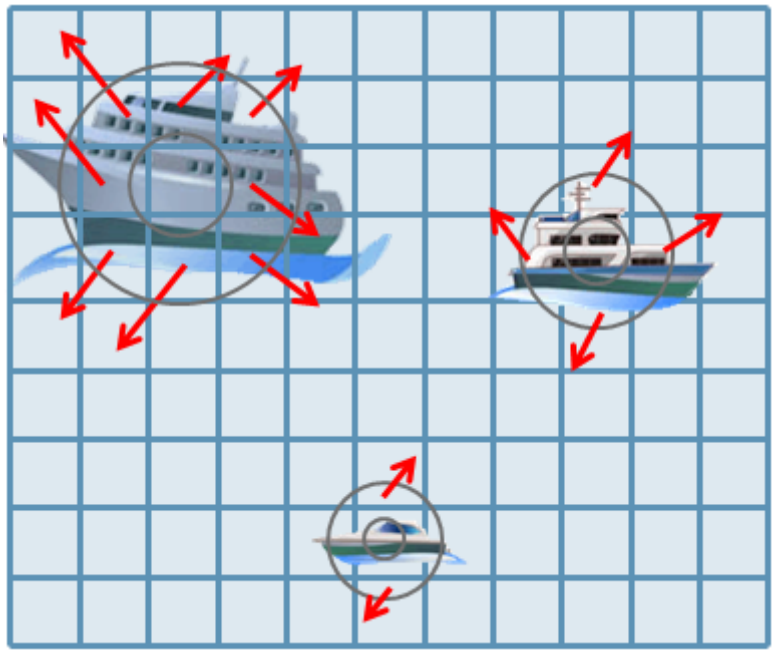

Figure 3.18: To compute features, we slide 15 different kernels (check Fig. 3.12) over the computed Fourier coefficients (marked red). These kernels have 3 different scales which allow to gather the relevant information of boats having a size within the range of the kernels.

Therefore, the idea we want to transmit here is that by sliding kernels which cover a certain range of scales, we can get features which best characterizes boats with different sizes within a certain range. Concluding, we can use several scales (but it will increase the computational cost of the algorithm) to cover the range of sizes we want. Then, we can train a model using features of boats with different sizes and features of the same boat but recorded in different flight altitudes, achieving scale robustness within a certain range.

Chapter 4

Computational Improvements

As it was stated in chapter 1 when formulating the problem, the computational cost is a huge concern. The computer which is embedded in the aerial vehicle has a small computational power, whereby there is the need to reduce the computational cost of the algorithms we selected to compose the outlined approach.

As it was explained in chapter 3, we developed two preprocessing steps (check section 3.1.1 and 3.1.2), which were included in our final prototype, allowing to achieve very satisfactory computational costs. However, in addition, we tried other solutions to reduce the time consumed by the algorithms. Therefore, this chapter shows what we did to overcome one of the three main requirements initially stated. Hence, in the following sections, we will discuss different approaches to face this problem.

4.1 Performing Convolutions in the Frequency Domain

In order to reduce the computational cost, we started to identify, step by step, the bottleneck of the algorithm proposed by Liu et al. ([22]) which are the convolutions performed in step 2 of Algorithm 3. Hence, we tried to optimize the code itself by identifying some code snippets which could be changed in order to reduce the consumed resources. The first aspect we noticed was the way Liu et al. were performing convolutions to compute regional descriptors ($U_{j,k} * \tilde{F}_m$). They were using a built-in function from MATLAB to convolve 2D signals spatially. However, a convolution in the space domain is the same as a multiplication in the frequency domain. Hence, we computed the FFT of the Fourier HOG fields, $\tilde{F}_m$, and the kernels used, $U_{j,k}$, separately (with *zero-padding*). Therefore, it is possible to perform multiplications between the FFTs of both signals instead of doing spatial convolutions, which is considerably faster. Then, we can revert this step by computing the IFFT of the result achieved after the multiplications. Fig. 4.1 shows this workflow which replaces the time-consuming spatial convolutions.

Testing this alternative way, we achieved exactly the same results, saving half the time we were spending. The authors code to compute the rotation-invariant HOG descriptors was taking, for our 720x405 images, approximately $10s$, while using the code with this slight change, we were taking between 4 and $5s$ to compute the rotation-invariant features. Since, we have exactly the same result doing the convolution in the space or in the

frequency domain, we are not jeopardizing the classification precision previously achieved by Liu et al. ([22]). Still our goal was to achieve about 1s per image. The next section presents an additional solution to decrease the computational cost.

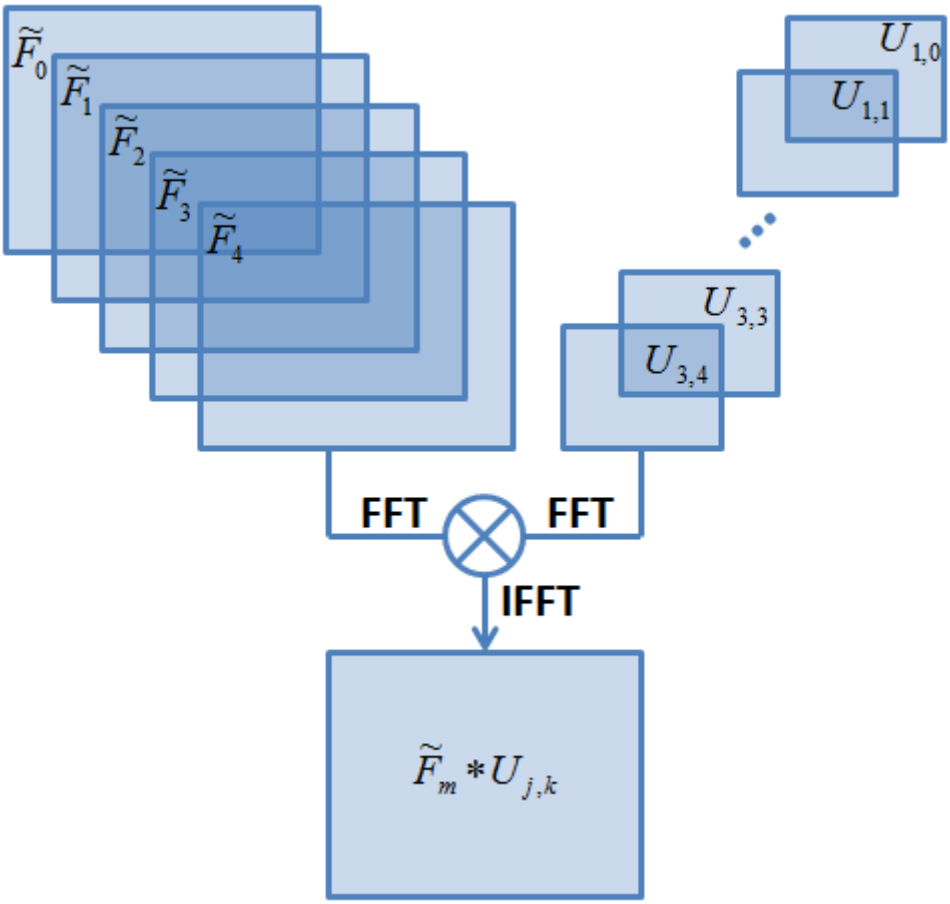

Figure 4.1: To overcome the bottleneck of the algorithm proposed by Liu et al. ([22]) which were the spatial convolutions performed to compute regional descriptors, we change it to the frequency domain. First we compute the FFTs of the Fourier HOG field, $\tilde{F}_m$, and the filters, $U_{j,k}$, separately. Then we multiply them and compute the IFFT of the result in order to go back to the spatial domain.

4.2 Frequency Downsampling

When we perform the multiplication between the FFTs of the image ($\tilde{F}_m$) and the kernel ($U_{j,k}$), representing the absolute value of the result (as in Fig. 4.2), we can conclude that most of the information is concentrated in the lower frequencies. This observation corroborates what Liu et al. say in their paper ([22]) which is the fact these kernels (Fig. 3.12) behave as low-pass filters. Beyond that, using this property, we can outline a strategy to reduce the computational cost. Before we explain that, look to the simple example presented in Fig. 4.3, in which there is a 1D signal with sampling frequency F_s. This periodic signal only has information at low frequencies. Hence, considering that the signal has length N and, in this case, from 0 to $\frac{N}{4} - 1$ and from $\frac{3N}{4}$ to $N - 1$ we are taking all

the interesting information, we can assume that it is possible to do this frequency downsampling (just taking the blue shaded region), without missing the important information.

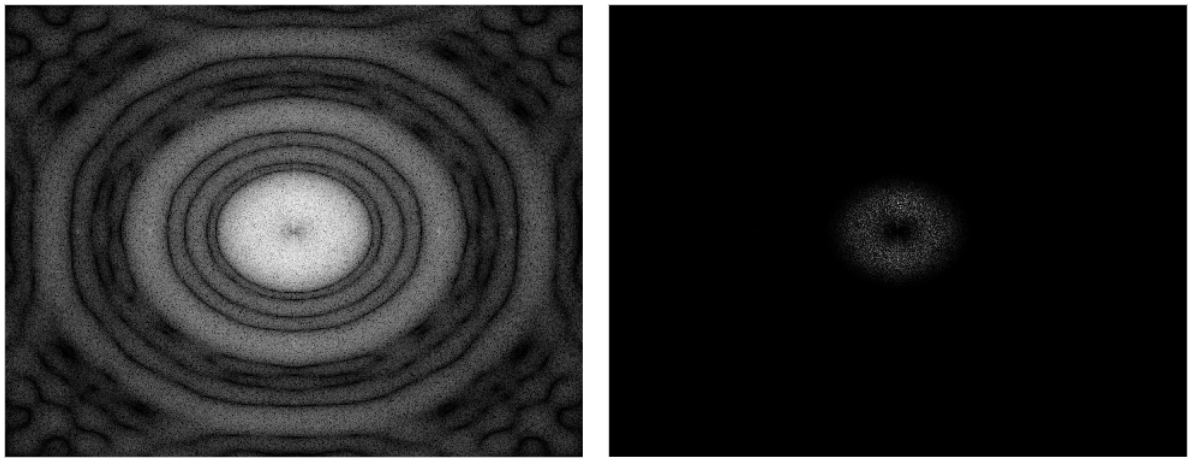

Figure 4.2: *Left* Absolute value representation of the multiplication between the FFTs of $\tilde{F}_m$ and $U_{j,k}$ using a logarithmic scale for a better perception. *Right* Absolute value representation without a logarithmic scale, where it is possible to see clearly that the essential information is at the low frequencies. Note that we used the *fftshift* MATLAB command to shift zero-frequency component of the Fourier Transform to the center of the spectrum. In this case, $m = 2$, $j = 1$ and $k = 3$ (several tests were performed to every possible combinations, achieving the same conclusions).

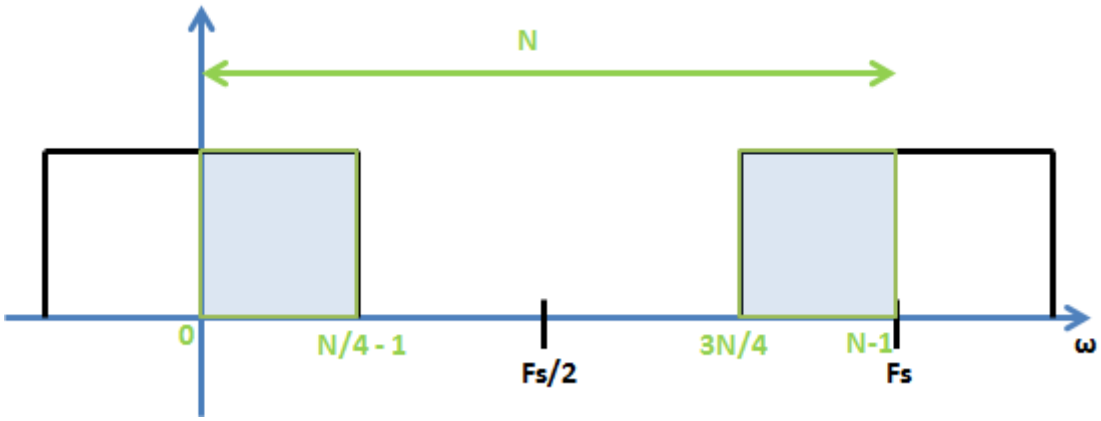

Figure 4.3: 1D signal which has its information concentrated at the low frequencies. Performing the correct frequency downsampling can help to reduce the computational cost.

To confirm this idea, we made some tests. Basically, we studied different possibilities of downsampling in the frequency domain before we compute the IFFT. We did a downsampling equal to the one presented in Fig. 4.3. However, in our case, since we are dealing with matrices, we did it along the 2 dimensions. In this case, the result of the multiplication between the FFTs of the image and kernel will be 4 times smaller, since we just take half the elements along each dimension. Thus, performing this downsampling, we are saving time computing the IFFT. We also tested this strategy in order to reduce it 16 and 64 times. The absolute value of the IFFT is represented

in Fig. 4.4. As it is possible to visualize, we have the same result but downsampled. We are still representing the important information which was concentrated, as we checked, at the lower frequencies.

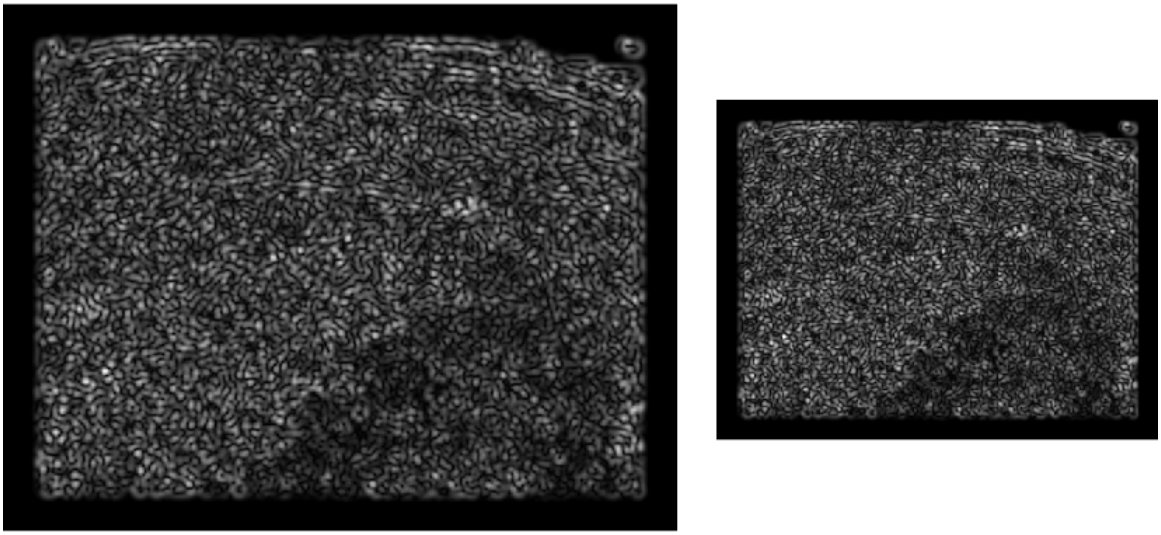

Figure 4.4: Example of the IFFT of the multiplication between the FFT of $\tilde{F}_m$ and $U_{j,k}$ (in this case, $m = 3$, $j = 1$ and $k = 3$). *Left* Without frequency downsampling. *Right* Frequency downsampling, reducing the time consumed in the computation of the IFFT for a quarter.

Actually, the Nyquist-Shannon sampling theorem states that, when a continuous function is reduced to a discrete sequence and then interpolated back to a continuous function, it is possible to perfectly reconstruct the continuous signal if a sample-rate criterion is met. In order to avoid loosing information during the sampling process, the sampling theorem introduces the concept of a sample-rate (in terms of the function bandwidth) that is sufficient for perfect fidelity. Considering the Nyquist rate as twice the bandwith of a bandlimited function and the Nyquist frequency as half the sampling rate of a discrete-time system (in Fig. 4.3, 2x$\frac{N}{4}$ $samples/second$ and $\frac{F_s}{2}$ respectively), a sufficient sample-rate is the Nyquist rate or anything larger. Therefore, considering the sample rate as F_s, the bandlimit for perfect reconstruction is $B \leq \frac{F_s}{2}$, where B is the bandwidth of the bandlimited signal. Since the downsampling we performed meet that condition, we can hope to reconstruct the continuous signal without missing information. However, the Nyquist criterion is only valid to functions whose Fourier transforms are zero outside of a finite region of frequencies (as in Fig. 4.3). Despite in our case it is not truth, we previously proved that the important information is concentrated at the lower frequencies. Hence, we will be able to reconstruct a signal which has the features we need to correctly characterize our targets.

More than confirm this proposed frequency downsampling visually (refer to Fig 4.4), we have to prove that it is indeed working. The Parseval's theorem states that the integral of the square of a continuous function is equal to the integral of the square of its continuous Fourier transform. Thus, it is possible to conclude that the total energy contained in a function summed across all of time is equal to the total energy of its Fourier Transform summed across all of its frequency components. Therefore, in order to ensure that the frequency downsampling performed before the IFFT is not missing the important information, we computed the ratio between the energy of the spectrum which we are throwing away and the total energy of the Fourier Transform without frequency downsampling.

The result was approximately 0.005% which means that we are keeping the important information concentrated at the lower frequencies.

Chapter 5

Results

This chapter presents the results attained using the methods described in chapter 3 and implementations described in chapter 4. First, the dataset available will be described. The evaluation methods will also be explained. Some implementation details will be given. Then, the results achieved for each video sequence will be presented. Beyond that, the drawbacks of our solution will be stated, in order to understand what can be done to overcome those limitations.

5.1 Dataset

The Portuguese Air Force and Navy provided 3 sets of videos with different sorts of vessels we may find in a real situation. The first set has 2 videos, both containing a big ship filmed at high altitude. In the first video, the ship is really small since it is recorded at a faraway distance, while in the second video the ship is closer and it is also possible to visualize another ship during a few seconds. The second set has 3 videos with a single vessel in each one but at different altitudes providing a wide range of scales for the same boat. Finally, the third set has 2 videos containing a small motorboat. In the first video, the motorboat is moving, thus there is a lot of wake difficulting our task. In the second one, the vessel is stopped. Except for the first video sequence of the first and second set, we are facing challenging illumination conditions, namely the sun reflections which are the main source of false positives. Figures 5.1, 5.2 and 5.3 shows an illustrative frame for each video sequence.

We also analyzed the size of the boats in the available videos. Figures 5.4, 5.5, 5.6, 5.7, 5.8, 5.9 and 5.10 show the size (considered as the maximum value between the BB width and height, in pixels, of the vessel) of the labelled boats for each video of the dataset. This study allows to have an idea of what we should expect to find. As it is possible to see, the size of the vessels varies a lot during the frames, revealing quite challenging. In section 3.3.4, it was stated that the kernels we are using to slide over the Fourier coefficients cover the range between 20 and 50 pixels, approximately. In the second and third clip of the second set, some vessels have a larger size. Here, even not detecting the boats completely, it is possible to detect certain parts since for larger boats we gathered features for those parts using smaller kernels. Fortunately, the boats which are really small appear in perfect daylight conditions (absence of sun reflections), whereby we can easily detect them by just applying a saliency method.

However, if necessary, we could train a model using samples gathered using smaller kernels than the ones we used. Table 5.1 presents further details about the available dataset. Note that we assigned an ID to each video in order to easily understand which video we are referring to. Since there are few videos provided until now, boats reveal themselves as the only targets to detect. As it was stated before, our prototype is ready to detect other kind of objects, provided that we use representative samples when training the model. Thus, videos containing oil spills, shipwrecked and lifeboats are necessary to test the outlined approach in different situations.

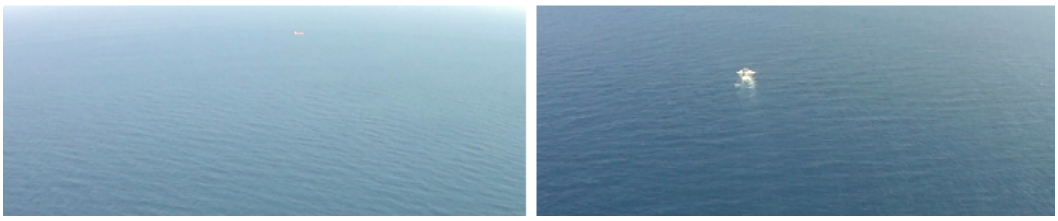

Figure 5.1: First available set. *Left* Big ship far from the plane; very small but with perfect light conditions. *Right* The same ship but closer to the drone; challenging illumination conditions in some frames.

Figure 5.2: A vessel recorded at different altitudes. *Top Left* Situation when the drone is farthest from the boat; perfect light conditions. *Bottom Left* The drone is closer to vessel; in some frames, sun reflections reveal as a problem. *Right* Drone relatively close to the boat; challenging illumination conditions.

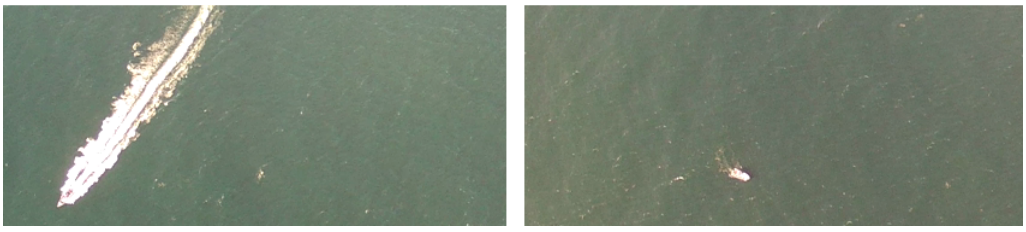

Figure 5.3: Two different situations with a motorboat. *Left* The motorboat is moving; sun reflections and especially the wake reveal as a major problem. *Right* Here, the motorboat is stopped; extremely difficult light conditions - in some cases the vessel is almost imperceptible being camouflaged by the sun reflections.

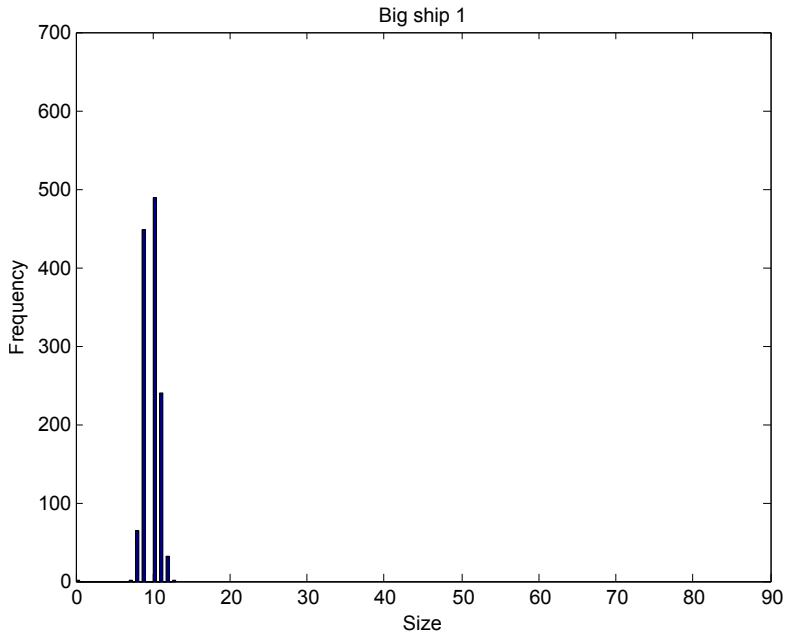

Figure 5.4: Representation of the size, in pixels, of the vessel present in the first clip of the first set. There, it is possible to see a big ship distantly, whereby the average size rounds 10 pixels.

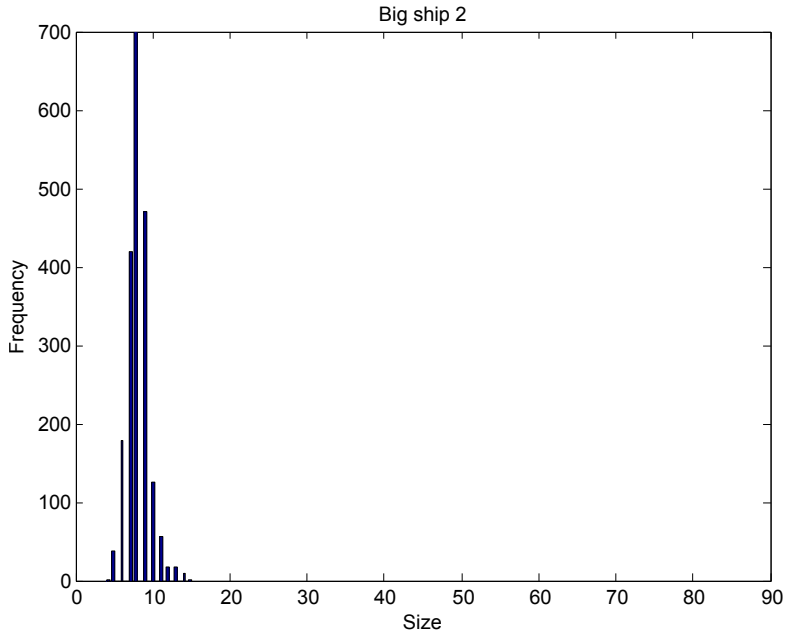

Figure 5.5: Representation of the size, in pixels, of the vessels which appear in the second clip of the first set. Here, it is possible to see a big ship distantly in addition to other one which is really far away from the boat, whereby the average size is lower than the other clip of the same set.

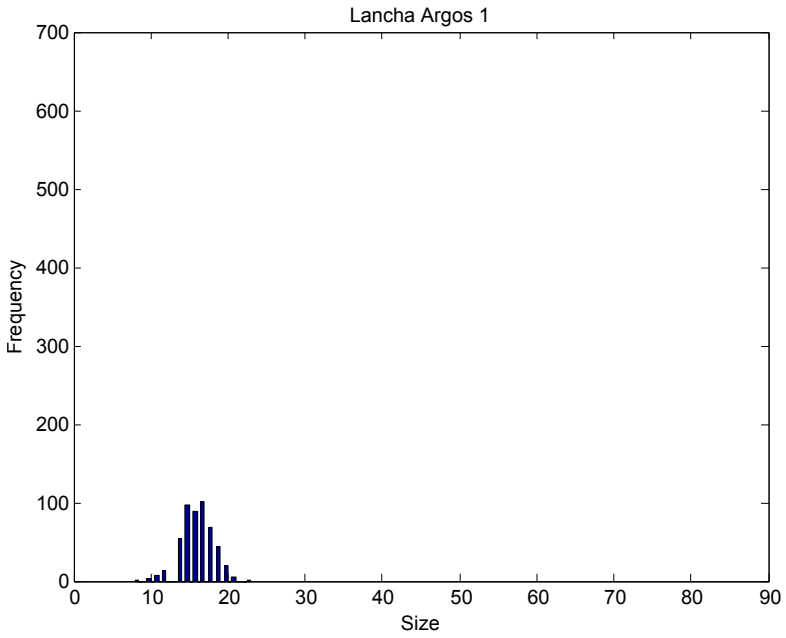

Figure 5.6: In the first clip of the second set, the drone approaches a vessel, whereby the average size is approximately 15 pixels.

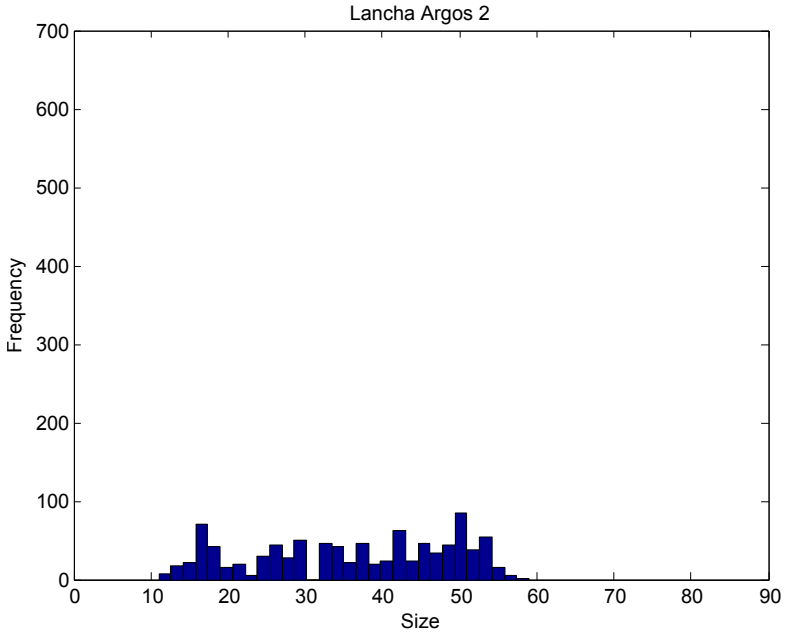

Figure 5.7: In this clip, the UAV goes arround the vessel which forces to deal with different scales in short time.

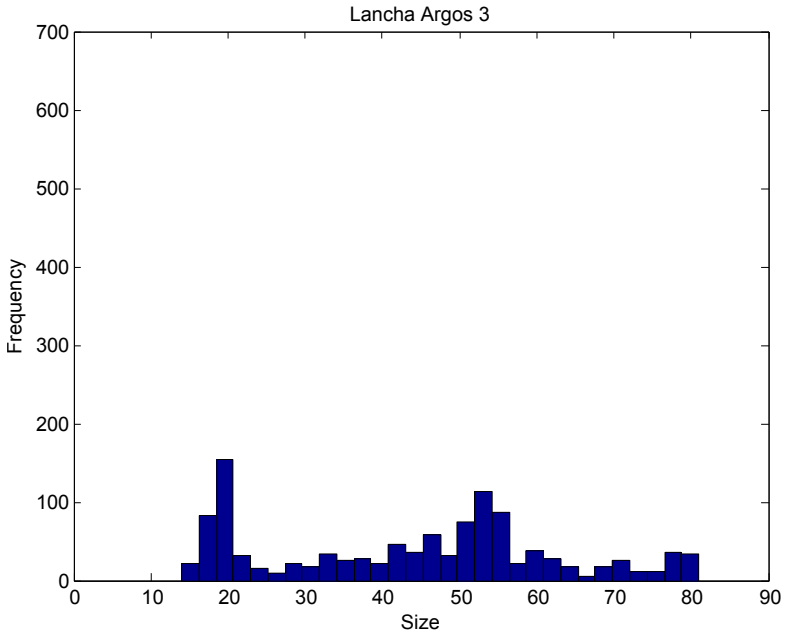

Figure 5.8: In the last clip of the second set, we face the same challenging situation of the previous video. Since the plane approaches and moves away from the vessel, the boat assumes a large range of sizes.

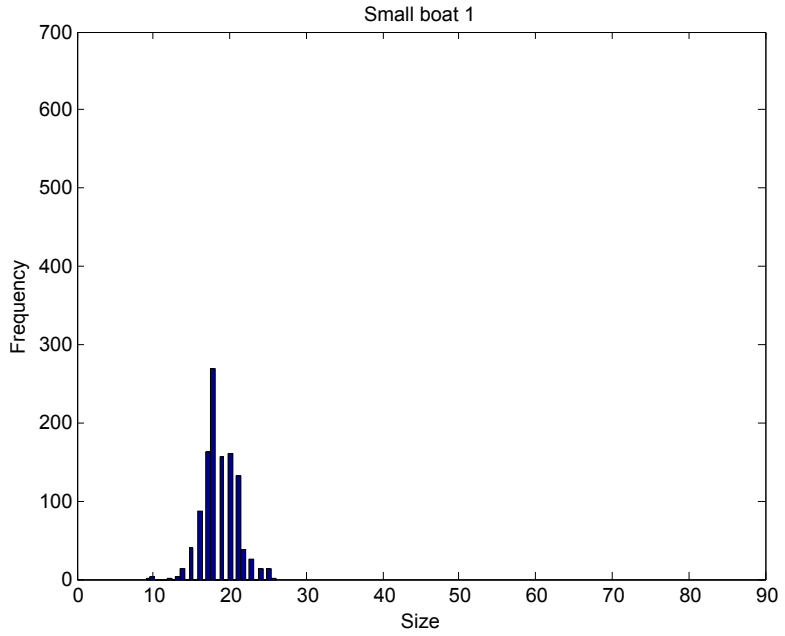

Figure 5.9: The first video of the last set presents a motorboat moving at high velocity. Its average size rounds 20 pixels but due to different flight altitudes, the boat size varies a bit.

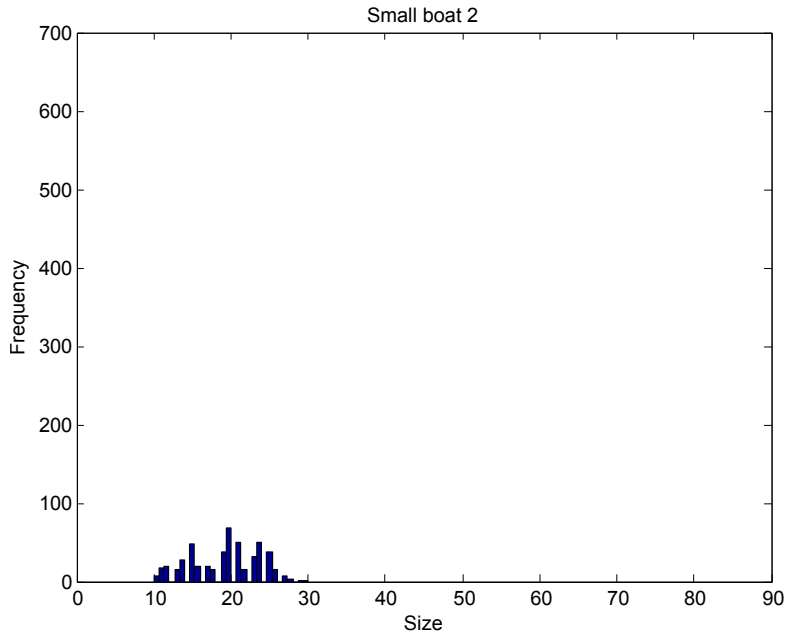

Figure 5.10: In last video, we have the same motorboat but in this situation it is stopped. The range of sizes is approximately the same.

Table 5.1: Available dataset: further details about each video sequence.

				BBs Width	BBs Height	
ID	Set	Video	Frames	Min/Max/Avg	Min/Max/Avg	Observations
1	1	1	1275	8/13/10	3/8/4	Small vessel; Clean conditions
2	1	2	2250	3/15/7	3/12/8	Medium vessel; Sun reflections; A small boat appears
3	2	1	505	8/23/16	5/13/9	Small vessel; Clean conditions
4	2	2	1070	8/59/36	3/26/17	Medium vessel; Sun reflections; Scale changes
5	2	3	1400	9/81/42	9/31/19	Large vessel; Sun reflections; Scale changes
6	3	1	749	7/30/18	6/24/15	Medium vessel; Sun reflections; Wake
7	3	2	2249	7/26/17	5/22/14	Medium vessel; Sun Reflections

5.2 Evaluation Methods

The evaluation of an information retrieval system generally focuses on two parameters: how relevant are the retrieved results (**precision**); if the system retrieved many of the truly relevant instances (**recall**). Considering that relevant instances are the events which we want to detect and detected instances are the ones our model really detected, we can define precision (also called positive predictive value) as the fraction of the detected instances which are relevant

$$precision = \frac{\#RelevantInstances \cap \#DetectedInstances}{\#DetectedInstances} = \frac{\#TruePositives}{\#TruePositives + \#FalsePositives}, \tag{5.1}$$

and recall (also known as sensivity) as the fraction of relevant instances which are detected

$$recall = \frac{\#RelevantInstances \cap \#DetectedInstances}{\#RelevantInstances} = \frac{\#TruePositives}{\#TruePositives + \#FalseNegatives}. \tag{5.2}$$

These concepts can be used to draw PR curves which present the trade-off between precision and recall for a specific test. Each point of the curve represents a possible working point of the algorithm, in other words, a set of parameters which allow to achieve that precision with the associated recall. Since it is almost impossible to obtain, simultaneously, values closer to 100% for both precision and recall, the working point should be chosen taking into account two kinds of possible errors: miss a target - FN - or detect an event which is really not a target - FP. Based on that, it is possible to infer some conclusions according to the customer's needs. If the client considers that is not so important to have a high sensivity, meaning that it is not a big concern if we have some FNs, our classification precision will significantly improve (less FPs overheading communications). This condition fits to our situation, since even if a boat is not being detected in a specific frame, it may be detected later (for instance, 10 frames after), because each boat appears for several frames. On the other hand, if communications are not so expensive, meaning that it is not critical to receive some FPs, we can select a different working point, with smaller precision but higher recall.

Rather than comparing curves point by point, it is better to have a single number which characterizes the performance of a classifier, giving an idea of how our detector is behaving. A common metric is the average precision which is precision averaged across all values of recall between 0 and 1:

$$\int_0^1 p(r)dr \tag{5.3}$$

Eq. 5.3 is equal to taking the area under the curve. The integral can be approximated by a sum over the precisions at every possible threshold value, multiplied by the change in recall as

$$\sum_{k=1}^{N} p(k)\Delta r(k) \tag{5.4}$$

where N is the total number of images in the collection, $p(k)$ is the precision at a cutoff of k images, and $\Delta r(k)$ is the change in recall that happened between cutoff $k-1$ and cutoff k.

5.3 Implementation Details

This section is dedicated to details about our implementation, mainly the feature computation (explained in 3.2.2). The dataset we used consists of almost 10000 images duly annoted. The annotation of GT was done by saving the top left coordinate of the BB containing a boat as well as its width and height. The detection task is very challenging due to the varying illumination conditions (where the sun reflections reveal as a major problem), different sizes of boats, different altitudes of flight and rotations of the drone and boats. We are detecting the boats by running a sliding window classifier on densely computed rotation-invariant HOG descriptors (refer to 3.3).

As we explained in section 3.2.2, the rotation-invariant HOG descriptors have 3 main steps: computation of the Fourier HOG field, $\tilde{F}_m$ (m is the order of the Fourier coefficients); computation of features through an application of filters, $U_{j,k}$ (check Fig. 3.12 for concrete examples where j and k are the scale parameter and the order of the kernel, respectively), on each computed $\tilde{F}_m$ in order to get regional descriptors; and finally, the generation of rotation-invariant features, ensuring the condition that the kernel we are using to filter has the same order of the computed Fourier HOG field ($k = m$). We also take the magnitude of the non-invariant filtering results, which are naturally rotation-invariant, as well as it is possible to generate many more rotation-invariant features using a coupling operation through eq. 3.13.

Looking to eq. 3.7, there are some details we need to clarify in this first step. About the normalization we are applying to the gradient magnitude (based on local gradient energy), it is being locally normalized as follows:

$$D' = \frac{D}{||D|| * K_2} \tag{5.5}$$

where D is the gradient of an image and K_2 is an isotropic triangle kernel of half-width equal to 12 pixels. The image gradients are projected into Fourier space according to eq. 3.4, where we compute the first 5 coefficients ($m \in \{0, 1, 2, 3, 4\}$), since the authors found, empirically, that these orders are sufficient to encode the useful information.

About the second step, we are applying filters, $U_{j,k}(r, \varphi)$ (eq. 3.9), to each degree-m component of $\tilde{F}_m$ to get the regional descriptor which describes the configuration of HOG features in the region covered by that kernel. The radial profiles of these kernels are triangles with half-width $\sigma = 6$ pixels, sampled at 4 radii, which are $r_j \in \{0, 6, 12, 18\}$ pixels. Regarding the angular part, only the lower degrees were considered ($k = \{-4, -3, -2, -1, 0, 1, 2, 3, 4\}$). In Fig. 3.12, we can visualize the real and imaginary part of the filters constructed under these conditions for radii larger than zero (the smaller kernel has 23x23 pixels, the middle one has 35x35 and the larger has 47x47) and 5 different orders (in Fig. 3.12, only the non-negative k are represented). We just take the features computed from the filtering of these kernels with the Fourier HOG field whose rotation order is in the range of $-4 \leq k - m \leq 4$. Beyond that, when using $\tilde{F}_0$ ($m = 0$), which does not have imaginary part, we do not apply the filters with negative k, since the result would be the same as with non-negative k. Therefore, this leads to 31 combinations of k and m for each scale (31x3 = 93 complex features). For $r_{j=0} = 0$, we are performing spatial convolutions on the Fourier coefficients $\hat{F}_0$ to $\hat{F}_4$ with an isotropic triangle kernel of half-width equal to 6 pixels, leading to more 5 complex features. Together with the features resulting from filtering with kernels which have radii larger than 0, we have 98 complex features.

Regarding the coupling operation (eq. 3.13), which allows to generate many more rotation-invariant features, we can have as many possible combinations as we want. The authors tested two settings. The first setting considers only the magnitude of the computed features and the complex features which are rotation-invariant without the coupling operation (when $k = m$), leading to a 111-dimensional rotation-invariant descriptor. In the second setting, they add the rotation-invariant features resulting from the coupling between features on different radii. Considering r_1, r_2 and r_3 the radius of the kernels used to filter, we obtain a total of 233-dimensional feature vector coupling between r_1, r_2 and r_2, r_3. Note that the coupling used is only one possibility of many combinations which allows to have a larger feature vector.

The preprocessing algorithms developed have some parameters which can be adjusted considering each situation. By now, regarding the saliency method, we are splitting the image into 16x16 blocks. As explained in 3.1.1, first we compute the magnitude of the gradient for each pixel. Then, we look for blocks whose sum of the gradient magnitude of all pixels belonging to that blocks is greater than a certain threshold (2.0). About the sun reflections detector, we performed some studies in order to characterize pixels belonging to that phenomena. Hence, we are considering that a pixel belongs to a sun reflection it its saturation component is between 0.0 and 0.2 while its brightness component is greater than 0.8 and lower than 1.0.

Finally, considering the classifier, it was used the open source library provided by Fan et al. (LIBLINEAR, [12]). LIBLINEAR is a package for solving large-scale regularized linear classification and regression. Therefore, we performed L2-regularized L2-loss support vector classification (dual). For the penalty parameter of the error term, we set $C = 1$. Regarding all the remaining parameters, we used the default values.

5.4 Tests Performed

As it was previously explained in chapter 3, we outlined an approach in order to deal with vessel detection in an oceanographic environment. First, it is applied a saliency method to get the most relevant regions of the images captured by the drone. After that, since the sun produce many FPs, it was developed an additional preprocessing step to filter those events. Finally, the algorithm provided by Liu et al. ([22]) was implemented and improved in computational aspects, which embeds the HOG-like features into the systematic Fourier analysis framework, allowing to build rotation-invariant descriptors for our targets and achieving scale robustness within a certain range.

This section starts by presenting a comparison between our approach and the case in which the rotation-invariant HOG descriptors are applied without the useful preprocessing tools we developed. Once proved that those preprocessing stages significantly contribute to a better performance, we will present the results attained with the different video sequences available. Some characteristics of our solution such as the scale robustness and the generalization ability will be shown as well as a computational cost analysis will be presented. After that, in the next section we will compare our proposed solution with other state-of-the-art algorithms.

As it can be verified in Fig. 5.11, in presence of sun reflections and wake, the saliency algorithms (section 3.1) reveal some problems, since they are highlighting those regions. Indeed, these events are salient from background

(high contrast). However, applying the second preprocessing step (section 3.2), we can filter the problem of sun reflections to some extent. On the other hand, it is still needed to deal with the wake which sometimes is a source of FPs. On the remaining salient regions we apply the detector using the rotation-invariant HOG descriptors presented by Liu et al. ([22]).

Figure 5.11: Output of the saliency method applied, based on contrast, in the video sequences provided. *Left* Video frame (from clip 7) in which a boat is highlighted; sometimes, the algorithm also highlights foam. *Right* Video frame (from clip 4) with a boat and presence of sun reflections.

To illustrate the importance of using those preprocessing steps, in the next subsection we will show what happens when implementing rotation-invariant HOG descriptors without the referred complementary preprocessing steps. Then, applying our final solution which encompasses those stages before the main method, it will be proved that these steps are really useful to achieve better results. This comparison will be based on results obtained when training with video sequences 3 and 5 and testing in video 4.

In order to evaluate the performance of the detectors, we used the tool provided by Dollár et al. ([10]). Their interface allows to plot PR curves as well as Miss Rate/False Positive per Image curves which give us an idea how our solution performs for different operating points.

5.4.1 Rotation-Invariant HOG Descriptors Without/With Preprocessing Steps

Figures 5.12 and 5.13 illustrate the output of the proposed algorithm without preprocessing steps. Although vessels are easily detected in the majority of frames, their model also frequently detects sun reflections. Fig.5.13 illustrates examples of FPs and FNs.

Fig. 5.14 show the PR curves obtained by testing the detector in the forth video sequence which was trained using the third and fifth videos as training set. In that figure, a comparison between the results attained by implementing the method proposed by Liu et al. ([22]) without the preprocessing stages and our final prototype is presented. As it is possible to check, the introduction of the preprocessing methods proved to be really helpful.

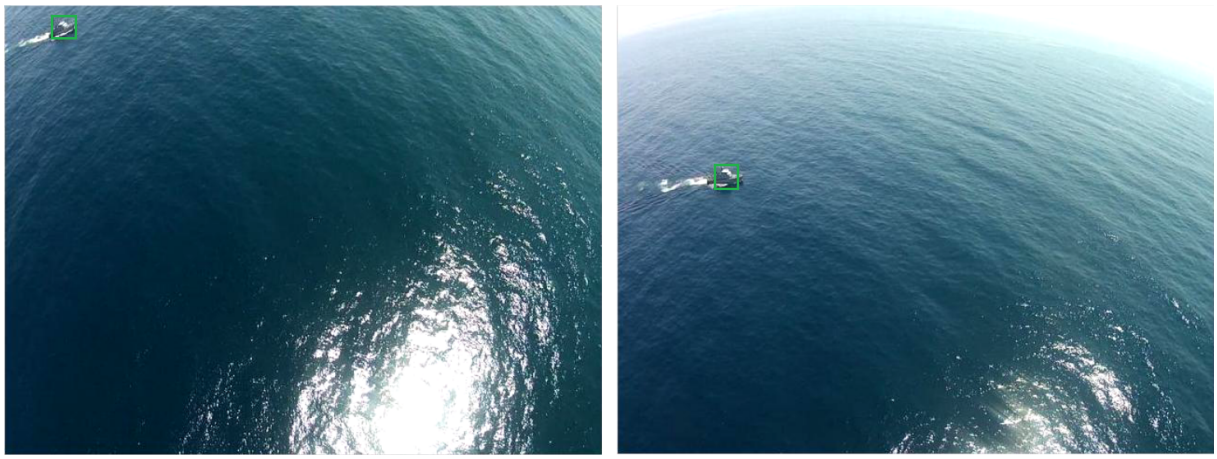

Figure 5.12: Examples of vessel detection in an oceanographic airborne imagery, using rotation-invariant HOG descriptors and a linear SVM for the classification task. Here, we are testing in video 4 using a detector trained with samples from clips 3 and 5.

Figure 5.13: FPs and FNs using method presented by Liu et al. ([22]). The testing sequence is video 4 while clips 3 and 5 were used as training sequences. *Left* The vessel was correctly detected but also two FPs are marked. *Right* FN: the vessel should be detected.

In our case, for the optimal selection of parameters, we retrieved 80% of the vessels (20% of the boats were incorrectly considered as background - FNs) achieving more than 90% of precision (10% of the results marked as vessels were background - FPs). The first stage, the saliency method, has a contribution only in the computational cost domain, since it just restricts the areas where powerful methods will be applied. The second preprocessing step, the sun reflections detector, is the responsible for the increase of performance presented by our solution. Actually, it filters most of sun reflections which are the main source of FP, contributing to increase the precision of our detector. Therefore, for the remaining sequences, it will be only presented the results attained by our full algorithm. Note that the percentage values in the legend are the average precision for each test performed.

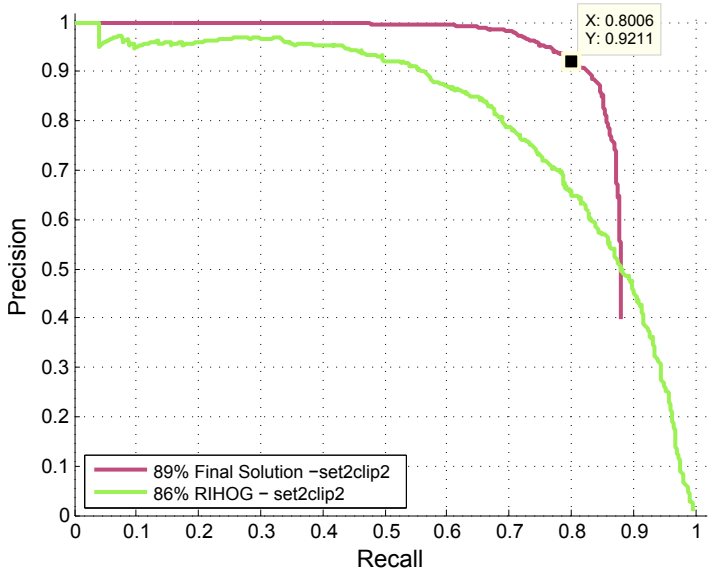

Figure 5.14: PR curves for detection results tested in video 4 using a model trained with samples from video sequences 3 and 5. Here we compare the result achieved with our final prototype and the one achieved only using the method proposed by Liu et al. ([22]). 92% of the retrieved results were vessels while 80% of the vessels were retrieved for the best working point.

Fig. 5.15 is an alternative way to study the behavior of Liu's detector with and without preprocessing steps. This curve allow to have an idea how both solutions are behaving. As it is possible to see, for 10^{-1} FPs per image, our solution only missed 15% of the vessels.

5.4.2 Final Solution - Saliency Method + Sun Reflections Detector + Rotation-Invariant HOG Descriptors

For the remaining video sequences of the previously presented dataset (5.1), we will analyze the results attained for each one by running our final prototype. First, we will prove that our solution presents some scale robustness taking into account the range of scales we use to train the model. Then, it will also be proved that our solution can be generalized to detect other kind of vessels. After that, we will state the main reasons why our solution revealed difficulties in some available videos. Lastly, the computational improvements achieved by implementing the preprocessing steps will be highlighted.

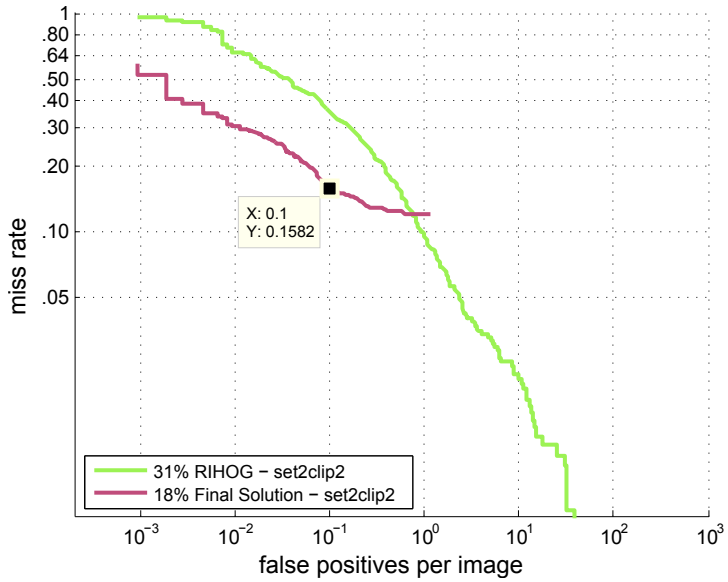

Figure 5.15: Miss rate/FPs per image curve obtained by implementing the method proposed by Liu et al. ([22]) with/without our preprocessing methods. The forth video was used as testing set while features of the third and fifth videos were used as training samples.

5.4.2.1 Scale Robustness

◦ **Video 5**

Fig. 5.16 shows the detection results for video 5 using as training examples the first and second video sequences. As it is possible to check, the results are very satisfactory. For the best threshold, 50% of the targets were really detected while 90% of the retrieved results were really a vessel. The reason why our detector did not detect a lot of vessels (lower recall) is the fact that the training set is composed by samples of boats in which the scale is lower than some periods of the video sequence used as testing set. Therefore, in our final model, it is really important to ensure that the training samples cover all possible sizes of vessels we want to detect. Despite of the recall is considerably lower than the one achieved in the previous test, it is still reasonable, since it means that we retrieved half the number of total vessels. Recall that a vessel appears during several frames, therefore it is not mandatory to detect it along all the frames.

Fig. 5.17 shows the Miss rate/FPs per image curve for tests performed in video 5 using video sequences 3 and 4 as training set. It shows that for a single FP per image, our solution only missed 14% of targets which is a satisfactory result.

5.4.2.2 Generalization Ability

◦ **Video 2**

After, the next goal was to prove that our solution could generalize for a variety of boats. In the previous cases, we were training the model with a specific type of a vessel and testing it in a different video sequence but with the same kind of boat. In this test we evaluate the ability of our algorithm to detect different kinds of vessels from the trained ones. Fig. 5.18 illustrate the detection results attained by testing our solution in the second video using a

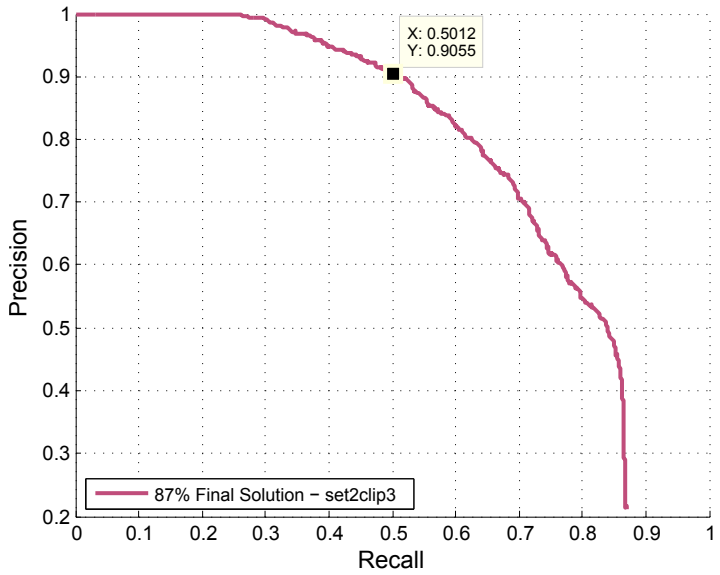

Figure 5.16: PR curve for detection results tested in video 5 using our outlined approach. The training set was composed by videos sequences 3 and 4. 90% of the retrieved results were vessels while 50% of the vessels were retrieved for the best threshold.

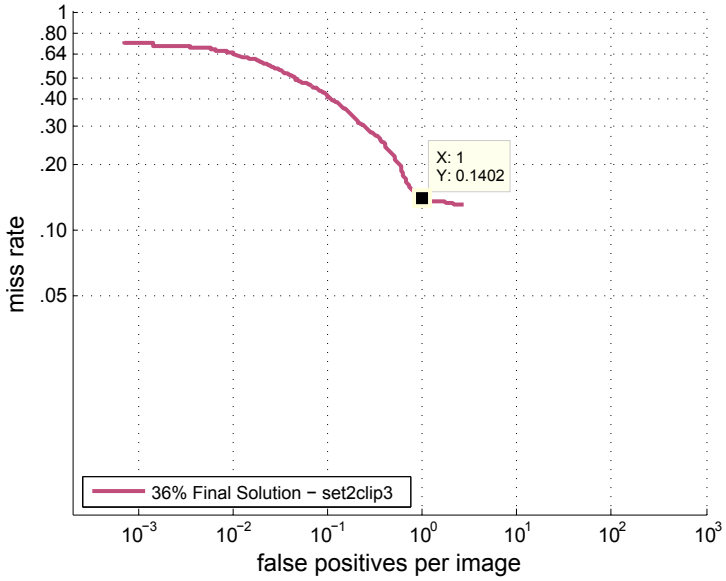

Figure 5.17: Miss rate/FPs per image curve obtained by testing our solution in the fifth video. Video sequences 3 and 4 composed the training set.

model trained with samples from the third and forth videos. The results attained were really encouraging. This video in which we tested our model, contains a big ship recorded at a longer distance. The low value for recall is due to the fact that, at some point, another ship appears which is not being detected since it is really far from the drone (its size is 4-8 pixels). Thus, our detector is not able to detect it. For the rest of the sequence, our prototype detects the big ship easily. Beyond that, our classifier detected very few FPs which justifies the higher precision values.

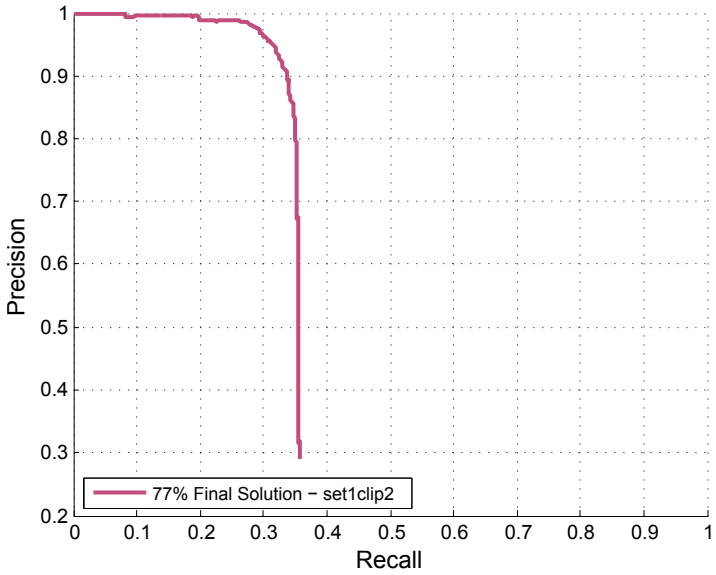

Figure 5.18: PR curve for detection results tested in the second video using a model trained only with samples of video sequences 3 and 4 in order to prove the ability to generalize in terms of detecting other kind of vessels.

◦ **Video 7**

Fig. 5.19 represents the detection results tested in second clip of the third set, using again a model trained with samples from the first two clips of the second set. Regarding this video, the PR curve is not so appealing, but it is possible to state that the results were really good. What explains this unusual curve is that during long periods, the boat is completely masked by solar reflections. Even an human has difficulties to found the vessel (which explain a lot of FNs). Beyond that, this is the video containing more challenging illumination conditions. Sometimes we are detecting sun reflections as well as wake (explaining the large number of FPs). For the remaining periods, the boat is detected almost every time.

5.4.2.3 Unsolved Cases

◦ **Video 3**

In this video sequence, as you can in Fig. 5.2, the vessel is recorded at a low scale. In this case the model is being trained with samples which represent higher scales (from videos 4 and 5). Thus, the boat is not detected. On the other hand, since this video presents perfect daylight conditions, the boat is easily detected just by applying our saliency method.

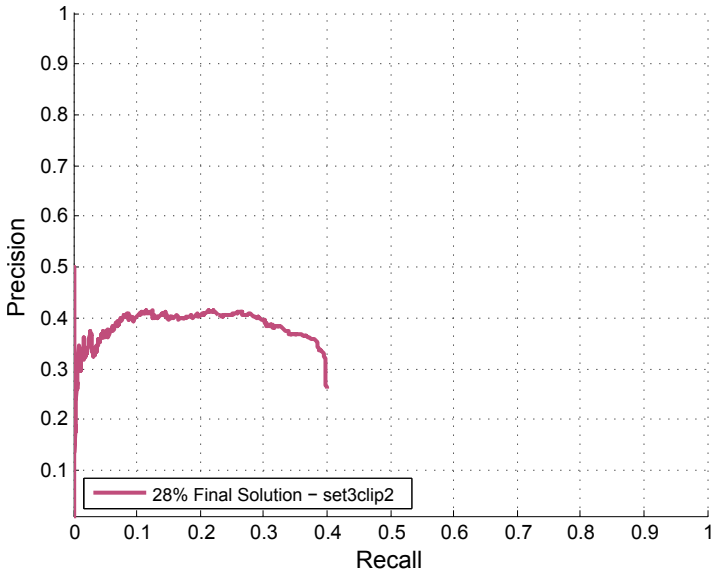

Figure 5.19: PR curve for detection results tested in video 7 using a model trained only with samples of the third and forth video sequences in order to prove the generalization of our solution.

◦ **Video 1**

As it can be checked in Fig. 5.1, the ship in this video is almost unnoticeable. Even the saliency algorithm does not detect it. In order to detect it, we would need to use smaller blocks than the current ones (16x16 blocks) which would increase the computational cost. However, we believe that is not so important to detect vessels in such lower scale, since even a human would have difficulties to spot it.

◦ **Video 6**

This video sequence presents a different and also really interesting situation. As you can see in Fig. 5.3, a motorboat is moving at high speed, creating a lot of wake. Therefore, our sun reflections detector equivocates, filtering the regions where there is wake and, consequently, it also filters the vessel. Sometimes, it does not filter it and in those situations our classifier detects the boat easily. This kind of situations obviously need some special processing which still need to be developed.

5.4.2.4 Computational Improvements

Most of the time available to perform this work was spent in order to decrease significantly the computational cost of our algorithm. As it was explained in 3.1, saliency algorithms give a considerable boost in the processing time, since they restrict the areas where a powerful method will be applied. Hence, table 5.2 shows the average time spent per frame for each video. The average time spent in processing a frame is considerably lower than $1s$ for any video, fulfilling the requirements originally set forth. As a matter of fact, in almost all the sequences tested, the average time spent per frame is lower than half a second meaning that it is possible to process $2frames/s$. However, the way the average time was computed is not very fair since the time spent in each frame depends on the size of the regions to process. Thus, each frame has different processing time but this value gives one idea of what we can expect. The variance for each sequence is also presented in order to measure how far the processing

Table 5.2: Computational cost analysis of our approach applied into several available video sequences.

	Set 1		Set 2			Set 3	
	Clip 1	Clip 2	Clip 1	Clip 2	Clip 3	Clip 1	Clip 2
Frames	1275	2250	505	1070	1400	749	2249
Total Time (s)	110.89	437.25	35.85	476.86	751.98	178.41	384.03
Avg. Time (s)	0.09	0.19	0.07	0.45	0.54	0.24	0.17
Variance (s^2)	0.02	0.04	$1.06\text{x}10^{-5}$	0.21	0.41	0.19	0.05

values for each frame are spread out. The results were computed in a $2.30GHz$ CPU, running a pure MATLAB implementation.

5.5 Comparison Against Other Methods

In order to have a clear idea of our solution performance, it was decided to compare with other approaches, more specifically, an algorithm developed to detect vessels (J. Marques, A. Bernardino et al., [26]) based on simple blob analysis rules and the Aggregated Channel Features (ACF) detector ([9]).

The method proposed by Marques et al. ([26]) bases its blob analysis on spatial and temporal constraints presenting robustness to variable background lighting. Their strategy consists of three stages: first, they apply a vessel detection algorithm to identify possible candidates (blobs); then, by making some spatial assumptions, they estimate the sky and sun reflection blobs and consequently elimate them; lastly, by establishing a temporal coherence between consecutive frames, they filter blobs which are not consistent.

Dollár et al. ([9]) developed an approach for object detection based on fast feature pyramids, more specifically for pedestrian detection. The ACF detector presents the following workflow: first, given an image, they start by filtering it and then they compute several channels (normalized gradient magnitude, HOGs and LUV color channels); after, the channels are splitted into blocks and the pixels belonging to each block are summed; finally, the channels are also smoothed with the same filter used in the image. Features are single pixel lookups in the computed aggregated channels. Regarding the learning process, these features are used to train and combine decision trees in order to distinguish target objects from background (implementing AdaBoost, [14]). A multiscale sliding-window approach is employed.

Figures 5.23, 5.20, 5.21 and 5.22 show the comparison between the built prototype, composed by the preprocessing methods followed by the rotation-invariant HOG descriptors, and the methods proposed by Marques et al. and Dollár et al. for tests performed in the available dataset (refer to section 5.1). Regarding a detailed explanation of the results achieved using our solution, please refer to section 5.4. Note that the results presented in those figures attained by implementing Marques et al. method are the ones before they apply the temporal coherence between consecutive frames, since we are treating each image as an independent event.

Our solution and Marques et al. method achieve higher precision values and similar values of recall for videos 4 and 5, mainly in the forth (check figures 5.20 and 5.21). The ACF detector shows reasonable results when applied to video 4 but, as we can check, presenting a lot of false positives which contribute to lower precision values. The

results attained implementing this detector were obtained by training a model with features from video sequences 3 and 5 as in our case. For that reason, it does not make sense to test the available model for that method in video 5, since we would need a model trained with samples from videos 3 and 4.

Although Marques et al. method has achieved slightly better results, it is possible to conclude that our versatility is an advantage. Since we have different working points, it is possible to achieve significantly better precision values without compromising the recall.

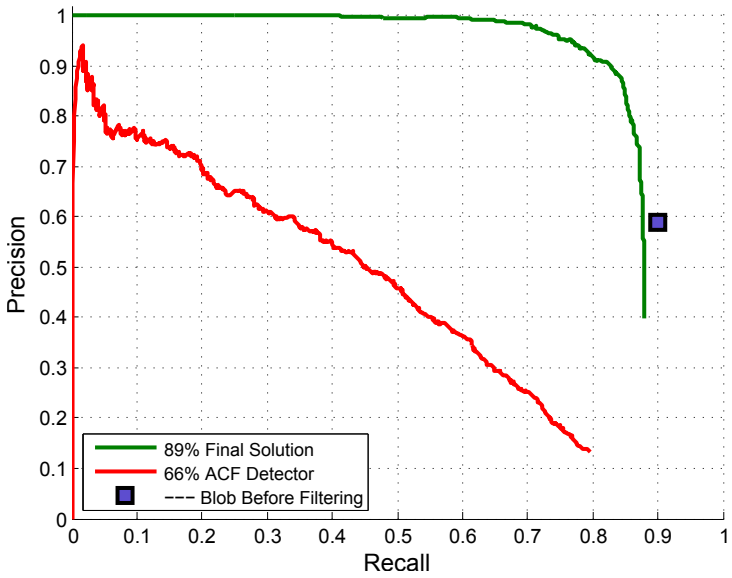

Figure 5.20: PR curves for detection results tested in video 4. Here we compare the result achieved implementing our final prototype and the ones achieved using Marques et al. ([26]) and Dollár et al. ([9]) methods.

Regarding video 7, the solution proposed by Marques et al. detects more FPs than ours. The sun reflections detector we used as preprocessing step revealed useful. The lower recall presented by both solutions is explained by the fact of the vessel be masked by sun reflections during long periods. Lastly, the ACF method detects a huge amount of FPs per image which explains the lower precision values.

Finally, the results achieved in video 3 (which contains a big ship captured at a faraway distance) show that Marques et al. method outperforms our solution (refer to Fig. 5.23). Clearly they can easily detect the other ship which appears at some point which justifies the higher recall. The performance of ACF detector in this video sequence was null. It does not detect a single vessel.

Concluding, we are very satisfied with the achieved results since our solution outperforms or at least equals state-of-the-art methods in almost every available videos. In video 2, in which our prototype revealed some problems, the main challenge was to deal with a ship which was really far. Thus, this reinforces the need to develop a fully scale-invariant approach in order to detect vessels of any size.

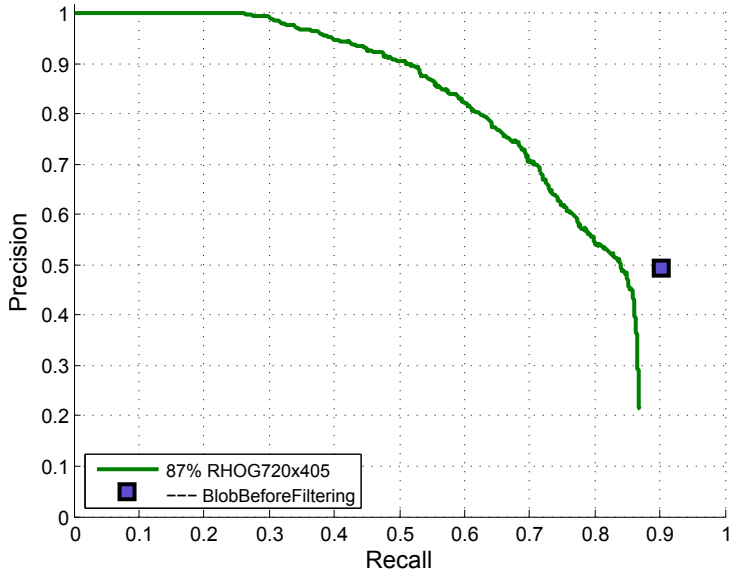

Figure 5.21: PR curves for detection results tested in clip 5. Here we compare the result achieved implementing our final prototype and the one achieved using Marques et al. ([26]) method.

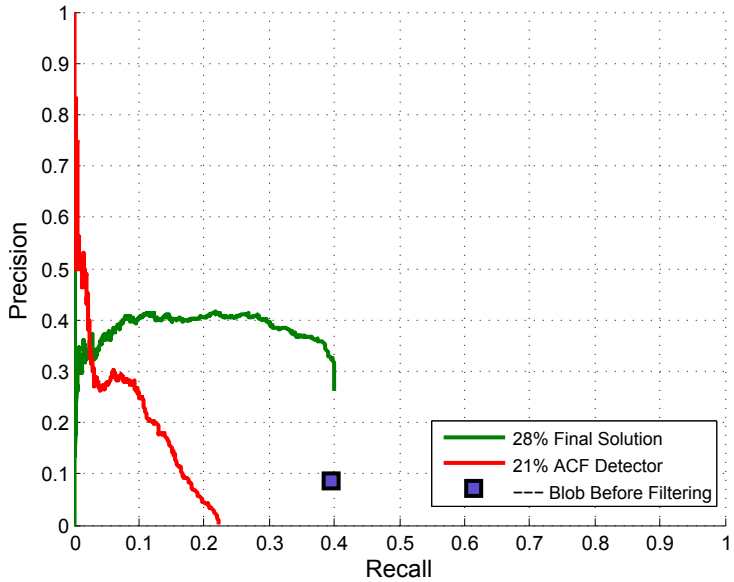

Figure 5.22: PR curves for detection results tested in video 7. Here we compare the result achieved implementing our final prototype and the ones achieved using Marques et al. ([26]) and Dollár et al. ([9]) methods.

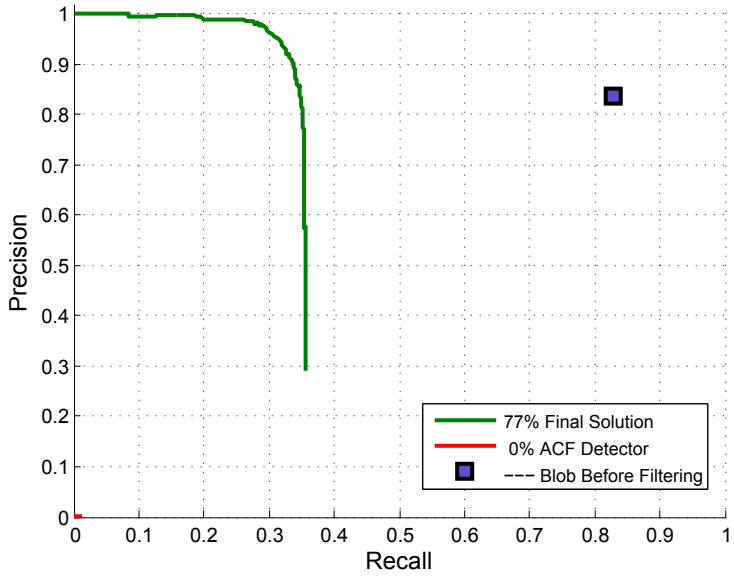

Figure 5.23: PR curves for detection results tested in video 2. Here we compare the result achieved implementing our final prototype and the ones achieved using Marques et al. ([26]) and Dollár et al. ([9]) methods.

5.6 Main Limitations

During the tests performed, we faced some limitations. One of them is related with rotation-invariance. The method proposed by Liu et al. aims to achieve 2D rotation-invariance. In their dataset, they were just trying to detect cars in aerial images of a city (check Fig. 5.24). Since the camera is fixed and the cars are only visualized by a top perspective, their method works very well since they are only facing 2D rotations. However, our case is more challenging. Despite of the images are 2D, the rotations of boats are in 3D. Thus, to compensate it, we have to add more training examples to the SVM covering a lot of perspectives, overloading our classifier.

Figure 5.24: Aerial images and the qualitative results (TPs marked as green and FPs as red) from their descriptor. They only deal with 2D rotations. (Adapted from [22])

Another noticed problem was the wake of the motorboats when traveling at high speed. Its wake, in some cases, is filtered by our sun reflections detector, whereby the boat is equally filtered, contributing to a lot of FNs.

Finally, one of the major requirements initially stated, scale-invariance, was only partially fulfilled. The way samples were given to train the classifier allowed to achieve scale robustness within a certain range. However, it is important to design a totally scale-invariance approach.

Chapter 6

Conclusions & Future Work

The work described in this report encompasses vessels detection in oceanographic environments. The main goal was to detect boats in different challenging situations. Beyond that, this work can be extended to detect other kind of events in a maritime environment such as shipwrecked, oil spill, lifeboats, among others. These detected events will be sent to a base station in order to be assessed by an human operator which will decide if these events need, or not, special attention.

The main constraint was the bandwidth limitation which did not allow to overload the communications, by sending all the images captured by the drone. Therefore, it was necessary to develop a semi-autonomous system which was able to detect vessels and send them to the base station. To deal with the formulated problem, it was outlined an approach to tackle the requirements stated. Our solution consisted in three steps: first, it was applied a saliency algorithm to reduce the areas which need a deeper processing using a powerful method; then, since sun reflections were highlighted by the previous method, it was applied a filter to remove them; finally, to confirm the presence or absence of ships in the detected regions, it was implemented the method presented by Liu et al. ([22]) which creates rotation-invariant HOG descriptors, connecting the HOG features with the analytical rotation invariance from Fourier analysis. A careful choice of training samples also promoted scale tolerance within a certain range. Very satisfactory and promising results were shown, as well as the main drawbacks were stated.

As main contributions, a new saliency method was developed which revealed more efficient in terms of computational cost than the state-of-the-art algorithms and presented very satisfactory results in our problem. The development of an algorithm to detect regions of sun reflections revealed very useful, avoiding to waste time by processing those regions of an image. The implementation of this preprocessing stages allowed to reduce the computational cost, restricting the areas which need to be deeply processed, and improved the performance of Liu's detector, mainly reducing the number of FPs. In parallel, two strategies (refer to chapter 4) were developed in order to improve the computational cost of the method proposed by Liu et al. ([22]): we are now performing convolutions in the frequency domain instead of the spatial domain; beyond that a downsampling in the frequency domain allows to keep the essential information, reducing singnificantly the time consumed. Tests conducted in this kind of environment are also new.

This research project has a lot of areas which still need to be developed in order to improve the final results.

Beyond the detection of vessels in each frame as an individual event, it is required to link these events along several frames, establishing a temporal coherency. There is some prior knowledge about the maximum speed ships can travel that allows us to temporally filter the individual detections.

The recognition of behavior patterns in vessels is also very important in order to predict their movements. Obviously, this forces to analyze temporal trajectories and grow our training and test datasets with ground truth sequences showing these patterns.

Likewise, the generalization capacity of our classifier, allowing to detect other kind of events such as lifeboats, to give an immediate answer to people who are in danger, or the classification of oil spills, in order to control environmental disasters, requires the acquisition and tests with datasets containing those events.

An algorithm which can deal with the wake created by motorboats when moving at high speed still needs to be developed, since our sun reflections detector is filtering the vessels in those situations.

Lastly, as it was stated in section 5.6, it is really important to develop a fully scale-invariant approach instead of achieving scale-tolerance within a certain range of interest by choosing carefully the samples to add when training the classifier.

Bibliography

[1] Support Vector Machine - Wikipedia, the free encyclopedia. `http://en.wikipedia.org/wiki/Support_vector_machine`.

[2] A. Andreopoulos and J. Tsotsos. 50 Years of Object Recognition: Directions Forward. *Computer Vision and Image Understanding*, 2013.

[3] E. Boser, I. Guyon, and V. Vapnik. A Training Algorithm for Optimal Margin Classifiers. *Proceedings of the Fifth Annual Workshop on Computational Learning Theory*, 1992.

[4] L. Breiman. Random Forests. *Machine Learning*, 2001.

[5] V. Chandola, A. Banerjee, and V. Kumar. Anomaly Detection: A Survey. *ACM Computing Surveys*, 2009.

[6] C. Cortes and V. Vapnik. Support-Vector Networks. *Machine Learning*, 1995.

[7] N. Dalal and B. Triggs. Histograms of Oriented Gradients for Human Detection. *Conference on Computer Vision and Pattern Recognition*, 2005.

[8] M. Dawkins, Z. Sun, A. Basharat, A. Perera, and A. Hoogs. tracking Nautical Objects in Real-Time via Layered Saliency Detection. *Proc. SPIE 9089, Geospatial InfoFusion and Video Analytics IV; and Motion Imagery for ISR and Situational Awareness II, 908903*, 2014.

[9] P. Dollár, R. Appel, S. Belongie, and P. Perona. Fast Feature Pyramids for Objects Detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2014.

[10] P. Dollár, C. Wojek, B. Schiele, and P. Perona. Pedestrian Detection: An Evaluation of the State of the Art. *Pattern Analysis and Machine Intelligence, IEEE Transactions*, 2012.

[11] A. Fadeev and H. Frigui. Dominant Texture Descriptors for Image Classification and Retrieval. *IEEE International Conference on Image Processing*, 2008.

[12] R. Fan, K. Chang, C. Hsieh, X. Wang, and C. Lin. LIBLINEAR: A Library for Large Linear Classification. *The Journal of Machine Learning Research*, 2008.

[13] P. Felzenszwalb, R. Girshick, D. McAllester, and D. Ramanan. Object Detection with Discriminatively Trained Part-Based Models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2010.

[14] Y. Freund and R. Schapire. A Short Introduction to Boosting. *In Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence*, 1999.

[15] J. Harel, C. Koch, and P. Perona. Graph-Based Visual Saliency. *Proceedings on Advances in Neural Information Processing Systems*, 2007.

[16] X. Hou, J. Harel, and C. Koch. Image Signature: Highlighting Sparse Salient Regions. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2011.

[17] L. Itti and P. Baldi. Bayesian Surprise Attracts Human Attention. *NIPS*, 2005.

[18] L. Itti and C. Koch. A Saliency-Based Search Mechanism for Overt and Covert Shifts of Visual Attention. *Vision Research*, 2000.

[19] L. Itti, C. Koch, and E. Niebur. A Model of Saliency Based Visual Attention for Rapid Scene Analysis. *IEEE Transactions on Pattern Analysis and Machine*, 1998.

[20] T. Kadir and M. Brady. Saliency, Scale and Image Description. *International Journal of Computer Vision*, 2000.

[21] W. Lin, L. Liu, Y. Matsushita, K. Low, and S. Liu. Aligning Images in the Wild. *IEEE Conference on Computer Vision and Pattern Recognition*, 2012.

[22] K. Liu, H. Skibbe, T. Schmidt, T. Blein, K. Palme, T. Brox, and O. Ronneberger. Rotation-Invariant HOG Descriptors Using Fourier Analysis in Polar and Spherical Coordinates. *International Journal of Computer Vision*, 2013.

[23] K. Liu, Q. Wang, W. Driever, and O. Ronneberger. Rotation-Invariant Detection Using Equivariant Filters and Kernel Weighted Mapping. *IEEE International Symposium on Biomedical Imaging*, 2012.

[24] D. Lowe. Distinctive Image Features from Scale-Invariant Keypoints. *International Journal of Computer Vision*, 2004.

[25] J. Malik, S. Belongie, T. Leung, and J. Shi. Contour and Texture Analysis for Image Segmentation. *International Journal of Computer Vision*, 2001.

[26] J. Marques, A. Bernardino, G. Cruz, and M. Bento. An Algorithm for the Detection of Vessels in Aerial Images. 11^{th} *IEEE International Conference on Advanced Video and Signal-Based Surveillance*, 2014.

[27] M. Reisert and H. Burkhardt. Equivariant Holomorphic Filters for Contour Denoising and Rapid Object Detection. *IEEE Transactions on Image Processing*, 2008.

[28] U. Schmidt and S. Roth. Learning Rotation-Aware Features: From Invariant Priors to Equivariant Descriptors. *IEEE Conference on Computer Vision and Pattern Recognition*, 2012.

[29] H. Skibbe and M. Reisert. Circular Fourier-HOG Features for Rotation Invariant Object Detection in Biomedical Images. *IEEE International Symposium on Biomedical Imaging*, 2012.

[30] A. Vedaldi, M. Blaschko, and A. Zisserman. Learning Equivariant Structured Output SVM Regressors. *International Conference on Computer Vision*, 2011.

[31] P. Viola and M. Jones. Rapid Object Detection Using a Boosted Cascade of Simple Features. *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2001.

Appendix A

Integral Image

Let us see a simple example in order to understand how this concept works. Imagine you have an intensity image I. The integral image, J, is a padded version of the cummulative sum along the columns and lines of I as $J(r,c) = \sum_{\substack{0<r'<r \\ 0<c'<c}} I(r',c')$. J is zero-padded at the top and left side, thus its size is $size(J) = size(I) + 1$. In Fig. A.1, it is possible to check a concrete a example.

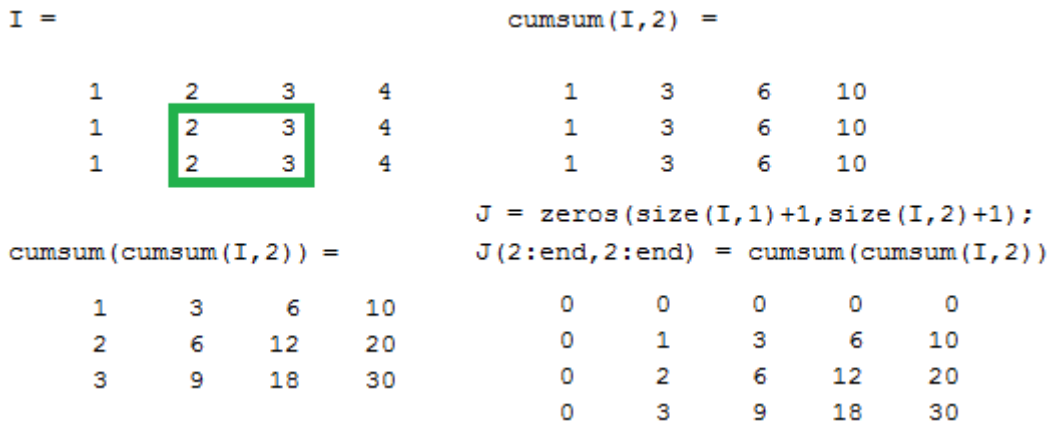

Figure A.1: Computation of an integral image, given an intensity image.

This technique can be helpful since we can use the integral image J to compute the sum of pixels over a rectangular region of the intensity image I. For instance, imagine we define a rectangular region whose starting row (sR), ending row (eR), starting column (sC) and ending column (eC) are, respectively, 2, 3, 2 and 3 (marked green). Looking again for Fig. A.1, we can see that the sum of the intensity values in image I belonging to that area is 10 ($2 + 2 + 3 + 3$). Now, using the computed integral image, it is possible to compute the sum over any rectangular region as $regionSum = J(eR+1, eC+1) - J(eR+1, sC) - J(sR, eC+1) + J(sR, sC)$, achieving the same result. Hence, this concept allows to save all the information which is important to retrieve the sum of pixels belonging to a specific region.

Appendix B

Shift Property of the Fourier Transform

Fig. B.1 shows a concrete example where the Shift Property of the Fourier Transform is applied. For the illustrated example, $v(r, \varphi) = e^{i4\varphi}$ and, according to that property, after a rotation of $55°$ $(0.96rad)$, the Fourier basis $v(r, \varphi)$ will be multiplied by a single complex factor which depends on the order of $v(r, \varphi)$ and the angle of rotation α as

$$v'(r, \varphi) = v(r, \varphi - \alpha) = e^{-i4\alpha} v(r, \varphi). \tag{B.1}$$

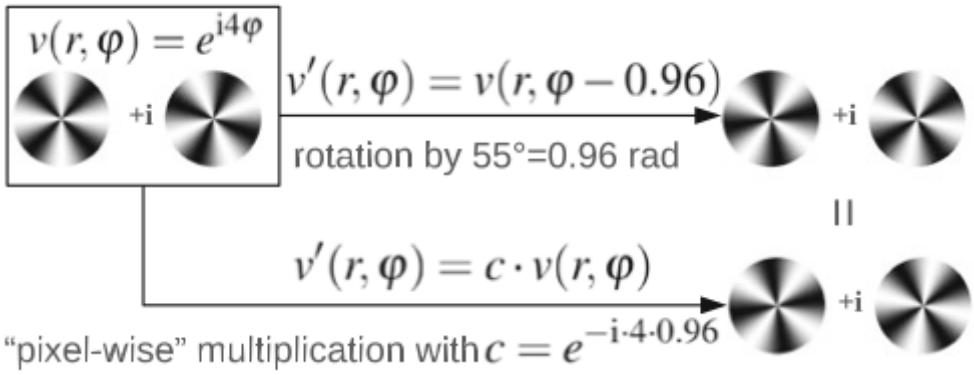

Figure B.1: Rotating a Fourier basis function in polar coordinates (Adapted from [22])