Master's thesis

NTNU
Norwegian University of Science and Technology

Amalie Kolsgaard

Improved Maritime Vessel Detection in Camera Images using Bag-of-Visual-Words

Master's thesis in Engineering Cybernetics
Supervisor: Annette Stahl
Co-supervisor: Rudolf Mester
June 2022

NTNU
Kunnskap for en bedre verden

Amalie Kolsgaard

Improved Maritime Vessel Detection in Camera Images using Bag-of-Visual-Words

Master's thesis in Engineering Cybernetics
Supervisor: Annette Stahl
Co-supervisor: Rudolf Mester
June 2022

Norwegian University of Science and Technology

NTNU
Kunnskap for en bedre verden

NTNU

Kunnskap for en bedre verden

TTK4900 - Engineering Cybernetics, Master's Thesis

Improved Maritime Vessel Detection in Camera Images using Bag-of-Visual-Words

Amalie Kolsgaard

June 2022

Faculty of Information Technology and Electrical Engineering
Norwegian University of Science and Technology

Supervisor 1: Annette Stahl
Supervisor 2: Rudolf Mester
Supervisor 3: Edmund Brekke

Preface

The work presented in this report was written as a part of the course TTK4900 - Engineering Cybernetics, Master's Thesis. The thesis was completed during spring 2022 for the Master of Science program at the Norwegian University of Science and Technology (NTNU), Department of Engineering Cybernetics.

The master's thesis is a continuation of the author's specialization project with the same title completed during the fall of 2021. The specialization project is not published. This means that important background theory and methods from the project report will be restated throughout this thesis to provide the best reading experience. A complete list of the material included from the specialization project is listed below.

- Parts of section 1.3.
- Parts of chapter 2: 2.1, 2.3, 2.5.

I would like to thank all the contributors who have supported me throughout the project. In particular I would like to thank my supervisor Annette Stahl and co-supervisors Rudolf Mester and Edmund Brekke for their valuable feedback and guidance throughout the work with this project. I would also like to thank Ph.D candidate Michael Ernesto Lopez for giving me programming advice when the implementation of the algorithm has been challenging. Edmund Brekke and Michael Ernesto Lopez has also contributed with important data, for which I am very grateful.

Amalie Kolsgaard

Trondheim, 8 June 2022

Abstract

Autonomous vessels relies on intelligent decision algorithms based on machine learning. As a major advantage in machine learning, the deep learning approach is becoming a powerful technique for object detection. The deep learning methodologies are applied in various fields in the maritime industry such as ship classification, collision avoidance, and navigation. However, challenges within the maritime environment give rise to the need for unique solutions. In this thesis, maritime vessels are classified and detected in optical camera images using the Bag-of-Visual-Words (BoVW) method for maritime environments. Features from vessel targets and background are first extracted using either interest point detectors or sliding window. Then, similar features are clustered using k-means clustering algorithm to create vocabulary for both categories. The vocabularies are used for testing the model performance on unseen data. The features corresponding to the vessel targets from the test set are matched to those of the training to classify each feature. The aim was to examine if BoVW could be used to extract information about the parts of the vessel to improve object detection in cluttered harbor environments.

The results from the tests done in this project shows that the method needs improvements. The method shows promising results in certain situations, however object detection in real-life cluttered harbor environments proved to be difficult.

Sammendrag

Autonome fartøy er avhengig av intelligente beslutningsalgoritmer basert på maskinlæring. En stor fordel innen maskinlæring er at dype nettverk utvikler seg mot å bli kraftfulle verktøy innen objekt deteksjon. Dype nettverk brukes i ulike felt innen den maritime industrien som blant annet til klassifisering av skip, kollisjonsunngåelse og navigasjon. Utfordringer innenfor havnmiljøet gir imidlertid behov for unike løsninger. I denne rapporten ble Bag-of-Visual-Words (BoVW) metoden brukt til å klassifisere og detektere skip i optiske bilder fra maritime omgivelser. En interessepunktdetektor eller sliding window blir først brukt til å hente ut deler av skipet. Så blir skips- og bakgrunnsdeler med like egenskaper gruppert sammen med k-means cluster algoritmen for å lage en ordbok for begge kategoriene. Ordbøkene vil så bli brukt for å teste ytelsen til modellen. De delene som tilsvarer skipsklassen, blir så matchet med treningsdataen for å klassifisere hver skipsdel. Målet med prosjektet var å undersøke om BoVW kan brukes til å hente ut informasjon fra fartøyet for å forbedre deteksjon av skip i havnemiljøer.

Resultatene fra testene som har blitt gjort i dette prosjektet viser at metoden har forbedringspotensial. Metoden viser lovende resultater under visse situasjoner, men skipsdeteksjon fra havnemiljøer viser seg å være utfordrende.

Table of Contents

List of Figures

List of Tables

Abbreviations

BoVW Bag-of-Visual-Words

CNN Convolutional Neural Network

CPU Central Processing Unit

FLANN Fast Library for Approximate Nearest Neighbors

GPU Graphics Processing Unit

HoG Histogram of Oriented Gradients

IDE Integrated Development Environment

IoU Intersection over Union

NN Nearest Neighbor

RGB Red Green Blue

SIFT Scale-Invariant Feature Transform

SVM Support Vector Machine

Chapter 1

Introduction

In this thesis the usage of the Bag-of-Visual-Words (BoVW) model is examined to efficiently classify and detect vessels in maritime environments. The BoVW representation is build using image analysis techniques, such as feature detection and description, clustering, and feature matching, in order to be able to recognize and localize the target objects in optical camera images. The aim of this thesis is to extract information about the parts of the vessel to improve object detection in cluttered harbour environments.

1.1 Motivation

Autonomous Surface Vessels (ASV) has in recent years made significant progress in the maritime research field. The potential advantages expand beyond reduced manual labor, increased traffic safety, and efficient route planning. A vessel's ability to communicate with other vessels, sense it surroundings and make decisions based on that information is crucial to the development of autonomous systems. The autonomous navigation system depends on a complex pipeline of situational-awareness and robust algorithms for collision-avoidance. Detection of objects, or locating surrounding objects is the first step in such a pipeline. Sensors such as LIDAR, radar and the AIS systems are commonly utilized for doing this. Such sensors are good at measuring distance. However, AIS relies on vessel-to-vessel communication and smaller vessels are not equipped with automatic identification system transponders. Other disadvantages are that the sensor may fail due to e.g. absence of satellite coverage, or it may provide false information for different reasons. Therefore, autonomous ships must be able to navigate among other autonomous and non-autonomous ships. This motivates research towards the exploration of camera-based navigation estimation. Optical images have proven to provide accurate object

localization information and has been extended into camera tracking systems. Camera sensors achieve situational awareness without relying on vessel-to-vessel communication [45].

In recent years object detection has been used to detect and classify objects in the surrounding of the ship. Based on this it is important to understand the environment where the detector is to be implemented. The object detection task in the maritime environment is placed in challenging conditions, where the environment naturally prevent the detector to perform robustly. Objects of interest are other maritime vessels, signs, buoys, and landmarks. Challenges to be addressed in a maritime environment include occlusion, blurred images, different variety of scales of objects, and changes in viewpoint. Object detection can perform well to high-quality custom datasets, however their robustness to new real-world images can not be guaranteed. A problem with using detection algorithms is that the predicted bounding boxes does not necessary cover the actual object perfectly. In addition, reflection in water and similar objects may results in false positive predictions. This is illustrated in Figure 1.1. Low accuracy and misclassification of other target objects are problems that often occur in research papers. Moreover, methods for classification that deal with maritime vessels in cluttered harbor environments are not widely known, or tested.

Object detection from maritime environments poses challenges that needs unique solutions. It needs algorithms with better adaptability to the various conditions encountered in maritime scenario. Hence, there is a need to do more research on solving these problems.

Figure 1.1: False positive maritime vessel detection.

In this project thesis, it is desirable to use Bag-of-Visual-Words (BoVW) for ship part detection to address the challenges within a harbor environment. The report involves using BoVW approach to extract features from maritime vessels on camera images and use these features to detect the vessel. The author's specialization project [20] had the same objective, but focused only on classification of the ship. The aim of this thesis is to extend the objective to localization of ship

targets. The methods used in this thesis are different from those in the specialization project. While it was used Support Vector Machine (SVM) to classify features, it will in this thesis be considered a comparison of visual words in the vocabulary with query features using *k*-Nearest Neighbor. With respect to this, the project will focus on implementing a detection framework to see if it is possible to use BoVW for ship detection in maritime environments.

1.2 Object Detection Methods Based on Deep Learning

In recent years, image classification and object detection have developed rapidly. The main contribution of this development is the implementation of convolutional neural networks (CNN). These are deep learning algorithms which is widely used in various classification applications. Being capable of taking intelligent decisions and handling large datasets are among the factors for their success [29]. Over the years, a greater number of variations of this architecture have been proposed. In object detection, the goal is to both classify and locate objects in the image. The first part is solved by CNNs and the remaining part is to find the subregions using object detectors.

Region-based CNN (R-CNN) proposed by Girshick et al. [13] and its variants like Fast R-CNN [8] and Faster R-CNN [40] are the typical deep learning-based object detection methods used today. Researchers developed R-CNN to deal with the task of object localization and classification. R-CNN is a concept that combine CNN with region proposals. For each input image, multiple windows which may contain the detected objects are generated by using Selective Search. Then these windows are classified using CNN. If the confidence of the the classification is high, the region proposals are returned from the algorithm. R-CNN model has achieved high performance with a mAP of 53,3 % on the PASCAL VOC dataset [13]. Despite the amazing achievements, none of the R-CNN architectures have managed to deal with real-time object detection. Fortunately, new architectures have been created to achieve this.

The "You Only Look Once" (YOLO) network proposed by Redmon et al. [39] is a CNN-based real-time detection method. In the past years, researchers have published several improved versions of the algorithm represented as YOLO v2, YOLO v3, YOLO v4, and YOLO v5 [37] [38] [5]. The core idea of YOLO is to divide the input image into SxS grid. The goal of each cell is to predict bounding boxes and confidence scores for these boxes. If the center of an object falls into a grid cell, that grid cell is responsible for detecting that object. Based on this, YOLO is a one-stage algorithm. This means that the tasks of object classification and localization are done in a single forward pass of the network. YOLO outperforms R-CNN with a mAP of 63,4 % while still maintaining real-time performance on PASCAL VOC dataset [39].

Single Shot Multibox Detector proposed (SSD) by Liu et al. [22] is a one-stage algorithm like YOLO. SSD runs a convolutional neural network on input image once and then computes a feature map. SSD has reached high performance and precision for object detection tasks, scoring over 74 % mAP (mean Average Precision) on the PASCAL VOC dataset. SSD achieves real-time processing speed and even beats the accuracy of the Faster R-CNN [22].

Currently, deep object detection architectures has achieved state-of-the-art results in both accuracy and speed in the object detection task in maritime environments. Deploying deep learning based object detectors in maritime environments is not a new phenomena. Moosbauer et al. [26] analyzed object detection task in maritime environments by training the Faster R-CNN architecture on the Singapore Maritime Dataset. In addition, Tangstad [41] trained a Faster R-CNN model to detect maritime objects for collision avoidance. Grini [17] worked with different type of deep-learning based detectors. Grini trained the YOLO v3 and SSD one-stage detectors for boat and building detection in maritime environments.

1.3 Related Work

Parts of this section were presented in the specialization project [20]

Traditional machine learning algorithms has grown in the field of computer vision as an alternative way to approach image classification problems. Bag-of-Visual-Words (BoVW) is one of these methods and was proposed by Csurka et al. [9] for image representation with the SIFT descriptors used in supervised image classification. This method has shown to have great capability for selecting and classifying the features by creating bags of each instance type [36]. Since 2003, the method has achieved state-of-the-art performance on several applications within computer vision such as human action recognition [33], scene classification [48] and face recognition [44]. Multiple papers have examined image classification using this approach and have come with different ways to improve it. Tsai [42] presents a review consisting of recent related works using BoVW for image classification. Jun et al. [2] proposed ways to improve performance of BoVW by comparing different feature descriptors. Pawara et al. [32] compared BoVW with three different classifiers; Support Vector Machine, k-Nearest Neighbor, and multi-layer perceptrons to deep CNN on plant datasets. Pawara et al. concluded that the deep CNN methods outperform the traditional methods.

As stated, this topic has been extensively studied, however few research papers examine the usage of image classification and localization of maritime vessel targets in maritime environments. Even fewer examine the usage of BoVW for object localization. To the author's knowledge, there

is no research papers that fully discuss these issues. Polap et al. [34] used CNN with the BoVW approach for faster feature extraction and classification of non-conventional ships. Costa [10] analysed the usage of BoVW model to efficiently classify and detect car targets using SVM and sliding window technique. Can et al. [6] evaluate detection and tracking of sea-surface targets on IR and VIS band videos based on the BoVW technique. The performance of the given BoVW models are evaluated on images consisting of only one object to classify and detect. Rarely taken into consideration of cluttered harbor environments.

1.4 Problem Formulation

The aim of this thesis is to use BoVW approach for supervised ship detection. The problem of interest is the detection of maritime vessel in optical camera images. The problem formulation is stated as follows:

Can BoVW be used to extract information about maritime vessel parts contained in camera images to improve ship detection?

According to the defined problem formulation, the objective of this thesis is to determine whether the BoVW model can produce a satisfying accuracy for object detection in harbour environments. This involves building a dataset and setting up a framework for training and testing of a BoVW approach. Experiments will be conducted in order to fulfill the aim of this master thesis.

1.5 Contributions

The main contribution of this thesis is the use of a BoVW approach to detect maritime vessels in maritime environments. To the best of the author's knowledge, BoVW has extensively been used for image classification, however few papers considers to use the approach for object detection. Especially, for maritime vessels in real-life harbour environments. Another contribution was labeling ship parts on a relatively large dataset. This was not used during this thesis due to limited time, however it can be used for future research.

1.6 Report Outline

The report is organized as follows. Chapter 2 presents the theoretical background needed for assessment and choices later in the report. Two different approaches for feature detection and description is presented, together with k-means clustering and feature matching with nearest neighbor algorithm. Chapter 3 introduces the experimental setup and methodology description of the BoVW approach used in the project. Chapter 4 contains the discussion of the results gathered from the experimental study. Results from comparison among the approaches are presented. Lastly, the conclusion of the report and future work suggestions are presented in Chapter 5.

Chapter 2

Theoretical basis

This chapter describes general concepts that form a theoretical basis for the proposed method. These concepts are aimed at establishing a common conceptual ground for the interpretation and understanding of the different terms used through the project. It is of interest for the project to obtain an overview of the Bag-of-Visual-Words (BoVW) approach and its building-blocks. Therefore, the theory behind each component will be presented in depth.

2.1 Interest Point Feature Extraction

Contents of this section were partially published in the specialization project [20]

In image processing, features are used to describe characteristic properties of images such as color, shape, or edges. When talking about features it is often referred to as local image features. Features include properties like corners, edges, or regions of interest points. A feature is called local if it is only influenced by a restricted region. Features can also be categorized as global, however local features are desired since it can be used as robust image representation that allows to find correspondence despite changes in viewing conditions, occlusion, and image clutter.

Feature extraction consist of feature detection and feature description. Feature detection refers to finding the features in an image, while feature description assigns quantitative attributes to the detected features [14]. In the following sections, each part will be presented.

2.1.1 Feature detector

The first step of feature extraction is to use a feature detector to find a set of local, distinctive interest points. Interest points are locations on an image that include the scale and orientation of the feature. The feature in this context is referred to as an image patch which is an image region containing rich local information. The image patch is specified by the coordinates and size of the feature's interest point as illustrated in Figure 2.1. The aim of the feature detector is to find the features that are likely to be relocated in other images with the same corresponding physical location [14]. The purpose of local features is to provide a representation that allows to efficiently match between local structures in images. It is desired to obtain a sparse set of local measurements that capture the essence of the underlying input images and that encode their interesting structure [32].

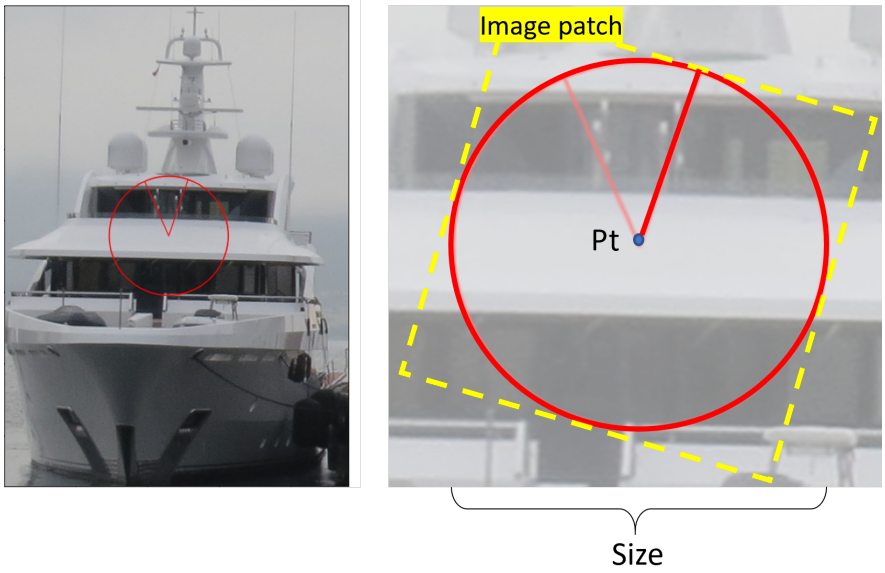


Figure 2.1: Illustration of how an image patch is defined from interest point

SIFT Detector

David G. Lowe [23] proposed a local feature extraction algorithm called Scale-Invariant Feature Transform (SIFT). SIFT detector has unique advantages since it extract features invariant to scale and rotation. The algorithm has therefore become a popular research topic.

The first stage of interest point detection is to identify locations and scales of the same object under different views. Detecting locations invariant to scale is accomplished by searching for stable features across multiple scales through smoothing and subsampling. SIFT defines the scale space of an image as a function $L(x, y, \sigma)$ that is produced from convolution between the

Gaussian function $G(x, y, \sigma)$ with an input image $I(x, y, \sigma)$.

$$L(x, y, \sigma) = G(x, y, \sigma) * I(x, y, \sigma) \tag{2.1}$$

where $*$ is the convolution operation in x and y. The Gaussian function is defined as in Equation 2.2. SIFT obtain features in various scales by changing the variable σ.

$$G(x, y, \sigma) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}} \tag{2.2}$$

To efficiently detect interest point locations in scale space, Lowe [23] proposed using scale-space extrema in the Difference-of-Gaussian (DoG) function $D(x, y, \sigma)$ shown in Equation 2.3.

$$\begin{aligned} L(x, y, \sigma) &= (G(x, y, k\sigma) - G(x, y, \sigma)) * I(x, y) \\ &= L(x, y, k\sigma) - L(x, y, \sigma) \end{aligned} \tag{2.3}$$

Scale invariance is then achieved when SIFT creates a DoG pyramid by subtracting the images which are adjacent in the same resolution. SIFT detect interest points by comparing each point with its adjacent 26 pixels, which is the sum of eight adjacent pixels in the same layer and nine in the upper and lower adjacent layers. This is represented in Figure 2.2. If the given point is a minimum or maximum, then the location and scale of this point are recorded. Because of this, SIFT gets all extreme points of DoG scale-space. SIFT then removes low contrast and unstable edge points.

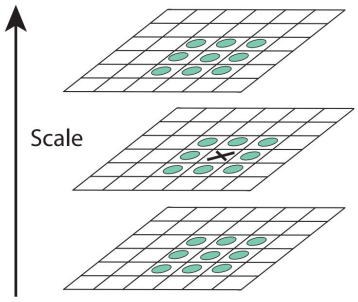


Figure 2.2: Maxima and minima of the DoG images. Image from [23]

2.1.2 Feature descriptor

Feature descriptors describes the area surrounding the detected keypoint by converting the keypoint into a feature vector. Based on this, descriptors can be compared in order to match features in different images. An ideal feature detector would be able to find distinctive features efficiently and their performance would be invariant to changes in illumination, rotation and scale. This is important since features can appear at different image locations and with varying orientations. Thus, a descriptor should be able to identify the same features under varying appearances [14].

SIFT Descriptor

In the phase of descriptor establishing, SIFT measures the local image gradient at the selected scale in the area around each interest point [23]. The SIFT descriptor is a 3D histogram of gradient locations and orientations that is stacked as a single 128-dimensional vector. The high dimension makes the descriptor relative slow to compute for larger sets. Before the SIFT descriptor is computed, an orientation is assigned to each interest point to make the feature robust to image rotation. To do this a neighborhood is taken around a interest point and the gradient magnitude and direction is computed for that region. From this an orientation histogram with 36 bins is created covering 360 degree. The direction of the interest point is defined as the highest peak of the histogram. Now interest point descriptor is created. SIFT takes a 16x16 neighborhood around the selected keypoint. Then SIFT divides this region into 4x4 sub-regions. For each sub-region, 8 bin orientation histogram is created. Thereby, SIFT gets a 128-dimensional feature descriptor (4x4x8) [46] [23].

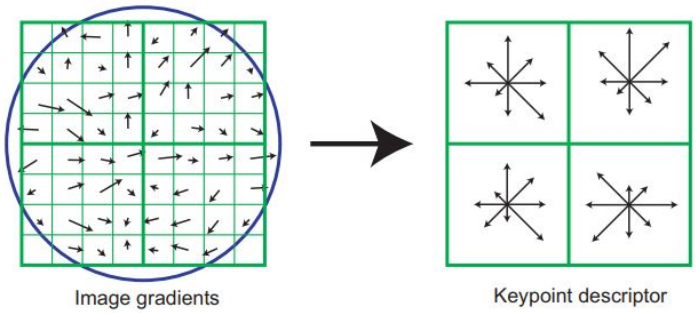


Figure 2.3: Illustration of how the SIFT descriptor is created. Image from [23]

2.2 Sliding Window Based Feature Extraction

In object detection, sliding window play an integral role as a technique to localize objects in an image. Sliding window is a rectangular region of fixed width and height that slides across the image. This is illustrated in Figure 2.4. For this project a sliding window is used to find features of the image where each window is analyzed to determine if the given region contain features of interest. When combined with image pyramids, it is possible to detect features at various scales and locations.

Sliding window require three arguments; The first is the image to slide across, while the second is the step size. The step size indicate how many pixels of the image to skip in both the x- and y-direction. The third argument defines the window size, or the size of the region to analyze.

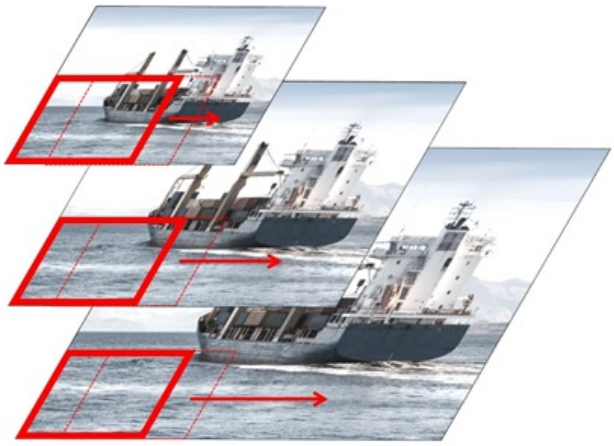


Figure 2.4: Illustration of sliding window with image pyramid

2.2.1 Classical Local Descriptors: Histogram of Oriented Gradients

Histogram of Oriented Gradients (HoG) is a feature descriptor that is used to extract features from images. HoG was initially introduced for human detection [11]. The HoG feature descriptor represents features by counting occurrences of gradient intensities and orientations in local image patches. The feature descriptor give a compact representation that only contains the most important information of the image. Typically, the feature descriptor converts an image of size (width x height x channels) to a feature vector of length N. The HoG descriptor focuses on the structure or the shape of an object since it uses magnitude and angle of the gradients to compute features. Magnitude of gradients is large around edges and corners which give information about object shape. The gradient is less, or none, at flat regions and thus give less information. The magnitude of gradients fires where there is a sharp change in intensity [24] [43].

The method is based on evaluating normalized local histograms of image gradient orientations in dense grid. Local object appearance and shape is characterized by the distribution of local intensity gradients or edge directions. The HoG descriptor is implemented by dividing the image into small cells of size *n*x*n*. Horizontal H_x and vertical H_y gradients of the cell are computed by applying the kernel [-1, 0, 1] as gradient detector. The gradient magnitude and orientation are computed as follow:

$$M_{(x,y)} = \sqrt{H_x^2 + H_y^2} \tag{2.4}$$

$$\theta_{(x,y)} = arctan\frac{H_y}{H_x} \tag{2.5}$$

A local histogram of gradient directions is computed over the pixels of the cell. The combined histogram entries form the descriptor. L2 normalization is applied to the histogram bins to reduce illumination variability [24]. HoG descriptor is used for this project to create a descriptor for image patches extracted using sliding window technique. This technique is not dependent on interest points and thus another type of descriptor was desired. Since HoG only considers the gradients on the image patch, the descriptor is neither invariant to rotation or scale. Therefore, it is applied a sliding window with image pyramid to get robustness to scale.

2.3 Classification models

In machine learning, classification refers to the machine's ability to assign instances to their correct groups. To be able to do this, the machine has to learn the patterns of the features available in the labeled training dataset. In this section, the algorithms used for classification of maritime vessels parts are presented.

2.3.1 K-means clustering

Contents of this section were presented in the specialization project [20]

k-means clustering is an unsupervised learning algorithm that aims to partition the data into a specified number, *k*, of clusters. The algorithm starts with selecting *k* points randomly from the dataset, and each point is assigned to the cluster with the nearest mean. A cluster refers to a collection of data points categorized together because of certain similarities [14]. Each mean is called the centroid of its cluster. The *k*-means algorithm then iteratively assign points to the nearest centroid in terms of the shortest Euclidean distance. The next step is to update the

cluster centroids. The process of data association and updating the centroids are then repeated until convergence is achieved [21].

The objective of the algorithm is to optimally satisfy the criterion:

$$\arg\min_{C} \left(\sum_{i=1}^{k} \sum_{z \in C_i} ||z - m_i||^2 \right) \tag{2.6}$$

where m_i is the centroid of the data in set C_i. The term $\|z - m_i\|$ is the Euclidean distance from a point in C_i to centroid m_i. The k-means clustering algorithm will always converge, but it is not guaranteed to yield the global minimum which result in bad clustering. An approach used frequently is to start the method several times with different starting conditions to increase the probability of finding the global solution. If the algorithm is run several times, then the attempt which best minimizes the Equation 2.6 is chosen [14].

The simplicity of the k-means algorithm sometimes affect the accuracy. An approach to tackle this problem is to use k-means++ initialization instead of running the algorithm from random starting points. k-means++ is similar to k-means, but the idea is to choose initial points that have the maximum distance from each other. For each point, the distance from the nearest, previously chosen centroid is computed. The next centroid is chosen such that the probability of selecting a point as centroid is directly proportional to its distance from the nearest, previously chosen centroid. This means that the next selected centroid is most likely to be the point which is as far away as possible from the closest centroid. This approach improves both speed and accuracy of k-means by helping the algorithm converge to the global minimum in fewer iterations. Because of this, k-means++ is a recommended initialization method [3] [7].

For this project, k-means++ is used as initialization method to cluster the feature descriptors computed from SIFT and HoG. These clusters will later be used to classify features using Nearest Neighbor (NN) algorithm.

2.3.2 Support Vector Machine

This section was published in the specialization project [20].

Multi-class classification is a classification task in machine learning where the machine should classify an image into one of many classes. Support Vector Machine (SVM) is a supervised machine learning algorithm for such classification problems. SVM aims to find an optimal boundary, known as hyperplane, between the different classes [16].

In the most basic setting, binary classification, SVM tries to find the boundary that maximizes the separation between the datapoints of the two-class dataset that are defined. The objective is to find the hyperplane that separates the datapoints into their potential classes in a n-dimensional space. A good choice is the hyperplane that leaves the maximum margin between the two classes, where the margin is determined as the sum of the distances of the hyperplane from the closest point of the two classes. The datapoints with the minimum distance to the hyperplane are called *support vectors*. In Figure 2.5, the support vectors are the two points laying on the solid lines, and the hyperplane separating the data is the dotted line [16] [30].

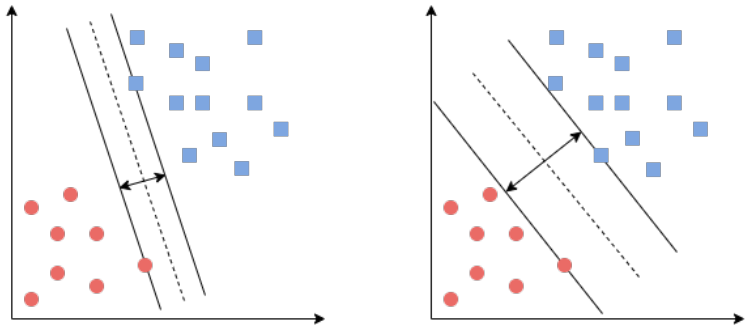


Figure 2.5: Illustration of SVM with (Left) a hyperplane with small margin and (Right) a hyperplane with greater margin

The computations of the separation of datapoints depend on a kernel function. Kernel functions are responsible for defining the decision boundary between the classes. A widely used kernel are the Radial Basis Function (RBF), which is defined by Equation 2.7:

$$K(x, x') = e^{-\frac{||x-x'||^2}{2\sigma^2}} \tag{2.7}$$

Where $||x - x'||$ is the Euclidean distance. The RBF kernel is used on non-linearly separable data where the decision boundary is needed to be curve-shaped. When using SVM in Scikit-learn,

the RBF kernel has two user-defined parameters; γ and C.

The γ parameter adjust the spread of the decision region. When γ is low, the curvature of the decision boundary is pretty broad. When γ is high, the curvature is high which creates decision boundaries around datapoints. The parameter C is the penalty for misclassifying datapoints. When C is small, the classifier accept misclassified datapoints, while penalizes the classifier for misclassified datapoints when C is large. The decision boundary then bends to avoid misclassifying points [1]. SVM for multiclass classification utilize multiple binary classification problems. The aim is to map datapoints into high dimensional space to generate similar linear separations between every two classes.

2.3.3 Feature matching with K Nearest Neighbor

Feature matching is how to match features in one image with other images. Feature matching involves comparing an unknown feature vector, in this thesis called descriptor, against a class of feature vectors. Then an unknown vector from test dataset is assigned to the class that is most similar to the unknown. The set of feature vectors are in this thesis referred to as the mean vectors computed from k-means cluster algorithm. The measure used to determine similarity is used to distinguish between one matching method from another [14].

One of the simplest and most widely used feature matching method is the Nearest Neighbor (NN) classier. NN classifier computes the similarity between an unknown descriptor and each of the feature classes using a distance-based measure. The NN algorithm assumes that similar features exist close to each other and is determined by the minimal distance. The classifier then assigns the unknown vector to the class containing its closest feature [14].

There are various ways of determining similarity, however the Euclidean distance is the most commonly used. The NN classifier computes the distances with Equation 2.8.

$$D_j(x) = ||x - c_j||, j = 1, 2, ..., N \tag{2.8}$$

where $||a|| = (a^T a)^(1/2)$ is the Euclidean distance. During classification the NN classifier assigns an unknown feature x to class c_j if $D_i(x)$ for $j = 1, 2, ...N$, $j \neq i$.

The NN algorithm is considered as a “lazy learning” algorithm, meaning that it only stores the training dataset. When asked to classify a test feature, the algorithm looks up the nearest entry in the training set and returns its associated class [15].

2.3.4 Fast Library for Approximate Nearest Neighbor (FLANN) library

The Nearest Neighbor algorithm is one of major importance in a variety of computer vision applications such as image recognition, pattern recognition and classification, and machine learning. The simplest way of matching features are to use a Brute-Force approach. This approach is simple. It match one feature descriptor with all other features in another set using a distance measure. A disadvantage with this NN algorithm comes when solving the problem in high dimensional space. The process of looking for feature matches can be computationally expensive when there are a high number of features. The feature descriptors used during this thesis is a high dimensional vector. Therefore, the Fast Library for Approximate Nearest Neighbor (FLANN) library is used to manage the task. FLANN is written in C++ programming language, but can easily be used with Python [27] [28].

2.4 Evaluation Metric for Object Detection

2.4.1 Intersection over Union

Intersection over Union is a measure that evaluates the predicted bounding box for object detection. IoU is given by the overlapping area between the predicted bounding box and the ground truth bounding box divided by the area of union between them. The ground truth is referred to as a manually drawn box over a correct object in the image [31].

$$\text{IoU} = \frac{\text{area}(B_p \cap B_{gt})}{\text{area}(B_p \cup B_{gt})} \tag{2.9}$$

By applying the IoU, one can define if the detection is valid (True Positive) or not (False Positive) by comparing the IoU to a threshold. Figure 2.6 show the IoU between the ground truth box (green) and predicted box (red).

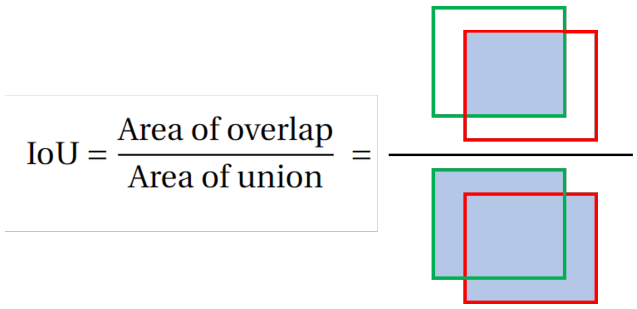


Figure 2.6: Intersection over Union

2.5 Bag-of-Visual-Words

Contents of this section were partially published in the specialization project [20].

Bag-of-Visual Words (BoVW) approach is commonly used in computer vision applications such as image classification. The concept was first introduced in Natural Language Processing (NLP) for text categorization. In the classic Bag-of-Words (BoW) algorithm, a document is represented as a histogram of frequencies of order-less words. Later, the methodology of BoVW was inspired by the BoW model to be applied for images in the field of computer vision. This idea was proposed by Csurka et al.[9]. To apply the BoW model in computer vision applications, an image is treated as a document, which can be considered as a collection of image features [19].

The classification approach is divided into two phases, where the first phase consists of training a classifier with samples built with a visual vocabulary, and a recognition phase where the classifier is used to identify new instances of the target object [10]. The visual vocabulary is generated by applying the *k*-means clustering algorithm on the extracted image features. As mentioned in Section 2.1, these features can be extracted in different ways using feature detectors. The local region around the detected features are represented by feature descriptors. Descriptors with similar features are then grouped into clusters using *k*-means algorithm. The obtained clusters are considered as visual words which represent specific local patterns presented in that cluster. Thus, a set of visual words generates a vocabulary which then represent different patterns in the image. The size of the vocabulary is defined by the number of clusters. Each image can be identified as a "bag of visual words" by mapping the features to visual words [**?**]. The "bag of visual words" corresponds to a histogram which is a representation of the occurrences of each visual word of the vocabulary. Based on this, histograms can be compared with each other to search for similar images[12]. The histogram is frequently used in the classification approach, however it is not used during this thesis. The reason is that it is used to classify whole images, which is not desired for an object detection task.

The last step of the approach is to classify the image by applying a classifier on the BoVW model to determine which category to assign to the image. This is usually done with Support Vector Machine (SVM), where the classifier is trained on the obtained features. Another option is to use Nearest Neighbor algorithm for features matching. The new image is classified by finding the smallest distance between the feature descriptors of the image and the visual words of the vocabulary obtained during training phase [**?**]. The last option is used during the thesis. A general overview of the BoVW approach is illustrated in Figure 2.7.

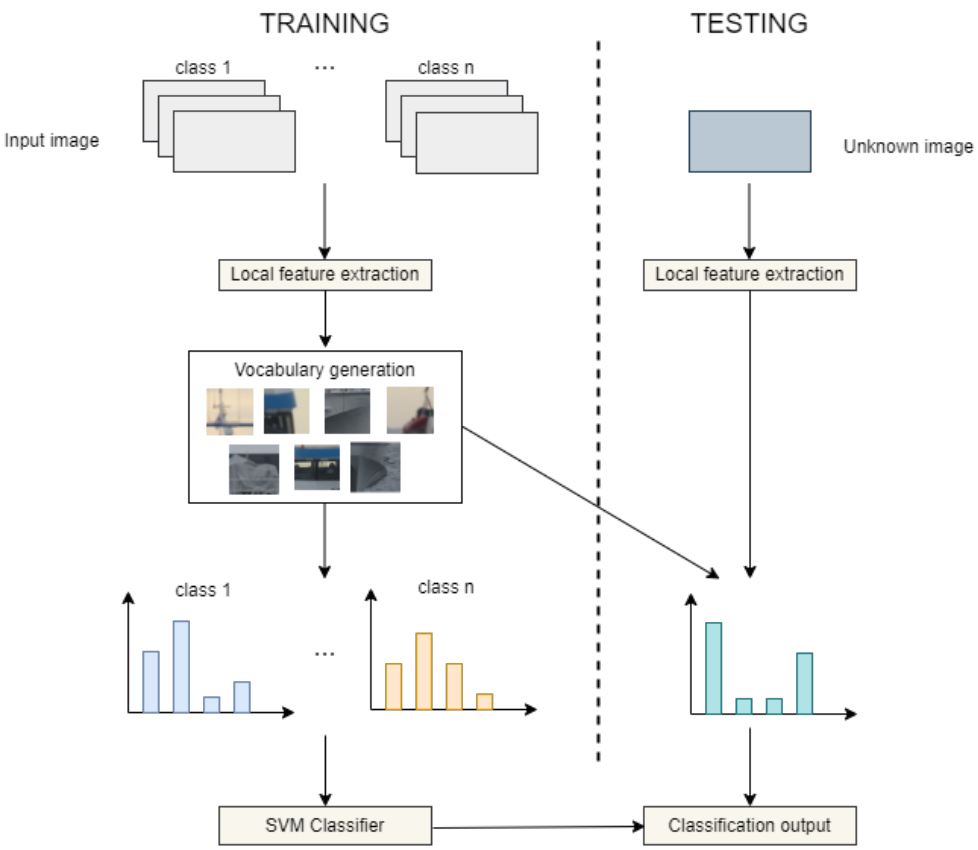


Figure 2.7: Schematic overview of the Bag-of-Visual-Words approach

Chapter 3

Method and Materials

This chapter presents the method used in this thesis in order to detect ships using BoVW approach. The proposed solution consists of several modules: preparation of dataset, feature extraction, vocabulary generation, and lastly evaluation of model performance.

3.1 Experimental Setup

3.1.1 Software requirements

The source code written in order to produce and present the results in this thesis is written with Python 3.8. Python is chosen as programming language as it arguably has the best open-source libraries for computer vision and machine learning. OpenCV library has been extensively used for computer vision tasks as image processing, feature extraction with SIFT, and descriptor matching using FLANN. The machine learning framework of choice has been Scikit-learn since it support k-means clustering. The source code has been written with PyCharm IDE and are available at Github [1].

To train a supervised classification algorithm on the dataset, the open-source graphical image annotation tool *LabelImg*[2] was used to label bounding boxes on the train and test images. The annotations were saved as .xml files with the PASCAL VOC XML format containing the ground truth box coordinate information and category labels. The labeling is further presented in Section 3.3.3.

[1] https://github.com/amalieKo/master-thesis
[2] https://github.com/tzutalin/labelImg

3.1.2 Hardware requirements

Training and testing with traditional machine learning techniques does not require specific hardware. The machine learning algorithm was not implemented with support for Graphics Processing Unit (GPU) computations since it performs well with simply using a Central Processing Unit (CPU). Therefore, all training of the BoVW model and experimentation has been performed on a 64-bit personal computer with the following relevant technical specifications:

- **Processor:** Intel Core i7-9750H. 6 cores / 2.59 GHz base clock
- **Memory:** 16 GB
- **Storage:** KINGSTON 1 TB M.2 SSD

With these specifications the training time was of approximately 3 hours. Naturally, this training time was affected by the number of images in dataset, number of features to detect and describe, and number of clusters to generate with k-means clustering.

3.2 Maritime Environment Datasets

Large datasets are considered important when trying to implement object detection algorithms that generalize well to a given dataset. In addition, for large datasets it is required to use a large amount of effort to annotate all images. Annotated datasets for maritime vessels are particularly hard to find since there is no large public dataset available. Therefore, extensive annotated maritime datasets proves to be a significant contribution for future research. Some examples of such datasets are MARVEL [4] and Singapore Maritime dataset [35]. When a dataset was to be selected, it was important that it had diversity of images and was annotated. The MARVEL dataset mostly contains images of re-occurring large cruise and container ships. However, it does not contain bounding box annotation. Because of this, the MARVEL dataset was not selected for this thesis. The Singapore dataset was another option, however it was discarded since it consists of videos. This results in similar scenes when converting to image sequences.

3.2.1 Brekke&Lopez Dataset

The Brekke&Lopez dataset was collected by Edmund Brekke and Michael Ernesto Lopez. The annotated dataset consist of optical images captured from the Norwegian ferry Hurtigruten at the coast of Norway and from different harbour environments. Hurtigruten dataset consists of a large quantity of harbour-relevant images. In addition, some images are captured from canals in Amsterdam with city-like environments. Samples from the respective sub-datasets are visualized in Figure 3.1.

(a) Sample image from Hurtigruten sub-dataset

(b) Sample image from Amsterdam sub-dataset

Figure 3.1: Sample images from Brekke&Lopez

The two sub-datasets are treated as one large dataset. In total, there are 1448 annotated optical images as shown in Table 3.1.

Table 3.1: Overview over number of images in Brekke&Lopez dataset

Optical images			
	Hurtigruten	Amsterdam	Total count
Image files	1334	114	1448

Utilizing a python script, developed by Lopez, to iterate through all annotation files, the amount of class-labels is presented in Table 3.2.

Table 3.2: Complete overview over class-labels of Brekke&Lopez dataset

Class-labels summary			
Class	Hurtigruten	Amsterdam	Total count
Airplane	3	0	3
Barge	48	62	110
Building	2488	1047	3535
Helicopter	1	0	1
Kayak	11	2	13
Motorboat	830	440	1270
Motorboat with priority	1101	45	1146
Sailboat with sails down	102	14	116
Sailboat with sails up	32	2	34
Class total	4616	1612	6228

The Brekke&Lopez dataset is class-imbalanced. *motorboat* and *motorboat with priority* comes much more frequent than *sailboat with sails up* and *sailboat with sails down*. *helicopter*, *kayak* do not count as notable classes. It was decided to not use *barge* class as it is similar to *building* class and it does not contain typical features of ship. Some images were excluded since they consisted of object that were too far away from the camera and thus too small. This makes the image no use for feature extraction. After these changes, the dataset contains insufficient number of images for object detection in maritime environment. Based on this it was of interest to combine the dataset with another dataset for this project.

Table 3.3: Overview over class-labels of Brekke&Lopez dataset after filtering our irrelevant images

Class-labels summary	
Class	Total count
Building	2764
Motorboat	876
Motorboat with priority	996
Sailboat	126
Class total	4762

3.2.2 ImageNet Dataset

The most established dataset in maritime environment, also containing bounding box labels is the ImageNet[3] dataset. ImageNet dataset consists of approximately 14 million images that are available for free to researchers for non-commercial use. ImageNet is used for training large-scale object recognition models. From ImageNet it was obtained 6 classes of different vessel types. An overview of the class-labels are shown in Table 3.4.

Table 3.4: Complete overview over class-labels of ImageNet dataset

Class-labels summary	
Class	Total count
Aircraft carrier	330
Catamaran	405
Container	420
Cruise ship	455
Lifeboat	455
Speedboat	386
Class total	2451

Some images were removed since they contained vessels that were too unique and thus affect the generalization in a negative way. *speedboat* class consisted of multiple motorboats with unique colors and patterns. Some boats had text livery which would been easily picked up by the feature detector. Liveries are not features that are desired for the training dataset. In addition.

[3] https://www.image-net.org/

aircraft carrier contained some images with irrelevant viewpoints, e.g on deck showing fighter aircraft. Sample images removed from ImageNet are shown in Figure 3.2.

(a) Speedboat with unique text livery

(b) Bad viewpoint at aircraft carrier

Figure 3.2: Sample images removed from ImageNet

Figure 3.3: Sample images from ImageNet dataset

3.3 Preparation of the Dataset

3.3.1 Merging of Datasets

To increase the number of images of the dataset, the two datasets presented in Section 3.2 were merged. Firstly, in attempting to do this the structure of the annotation files were changed to be equal in structure. Both Brekke&Lopez and ImageNet follow the PASCAL VOC XML format, however some elements of the file differed from each others. Thus, it was decided to change the elements of the annotation files. Figure 3.4 illustrate the structure of the annotation files of the two datasets together with the new merged dataset. The merged dataset will be used as the main dataset further in the thesis. The aim is to get a simplified annotation file with only the most relevant elements for further processing. This was done with a python script developed by Ph.D candidate Michael Ernesto Lopez.

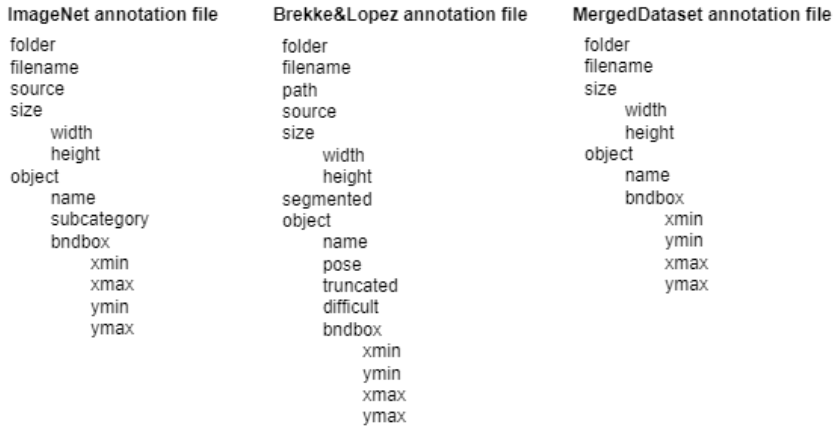


Figure 3.4: Overview of the structure of annotation files. ImageNet and Brekke&Lopez are changed to have the structure of MergedDataset

Then, the ship class-labels were changed so that they became similar before merging the two datasets. Two different annotation-merging approaches were considered. Ship sub-classes with high resemblance can be divided into *motorboat* and *sailboat*. Following such labelling methodology would place *motorboat with priority* into the *motorboat* class-label. Similarly, the two sailboat classes *sailboat with sails down* and *sailboat with sails up* can be treated as *sailboat*. For imageNet, *aircraft carrier*, *container*, *cruise ship*, *lifeboat*, and *speedboat* can be considered as *motorboat*, while *catamaran* as *sailboat*. The purpose of such a partition would be to train a detector to separate between motorboats and sailboats. In total there would be three classes to consider: motorboat, sailboat and building. However, this partition would yield an minority-class of sailboat. Therefore, another approach would be to treat all ship types as ship. This option would result in a two-class detection task; ship and non-ship. The last approach is used for this project. Implementing more classes was left for future work.

3.3.2 Collection of non-ship images

For the thesis it was desirable to learn the model to distinguish between ship and non-ship. Thus more images of background elements (i.e buildings) were obtained. Images were downloaded from the web and labeled manually with *LabelImg* since no such dataset was available. 124 new images of *building* were obtained. Adding more images would be too time consuming regarding labeling. Sample images from the non-ship dataset is shown in Figure 3.5.

Figure 3.5: Sample images from the non-ship dataset

The total number of class-labels used for training the model is shown in Table 3.5.

Table 3.5: Overview over the final class-labels in MergedDataset

Class-labels summary	
Ship	4146
Non-ship	2961
Total	7107

3.3.3 Labeling of The Dataset

To be able to correctly classify maritime vessels that is partly covered by other vessels, the object detection model need to recognize ship parts as ship. To achieve this, it was desired to detect three parts of the ship class; *front*, *side*, and *back*. The dataset was annotated with new labels using *LabelImg*. The three new labels were annotated inside the ground truth box of the ship class. The ship parts are shown in Figure 3.6. Manually labeling 4619 images was a huge workload for a single person. It took the author 14 days to complete the task which resulted in a huge

deviation in the progress plan. Although a lot of time was spent on labeling, these classes were not completely implemented into the detection algorithm. Due to the limited time, it was not an easy system to implement. As the submission deadline came closer, it became more important for the author to focus on improving the results that were already available. It was thus decided to only focus on ship/non-ship detection. Detection of these ship part classes are set as future work.

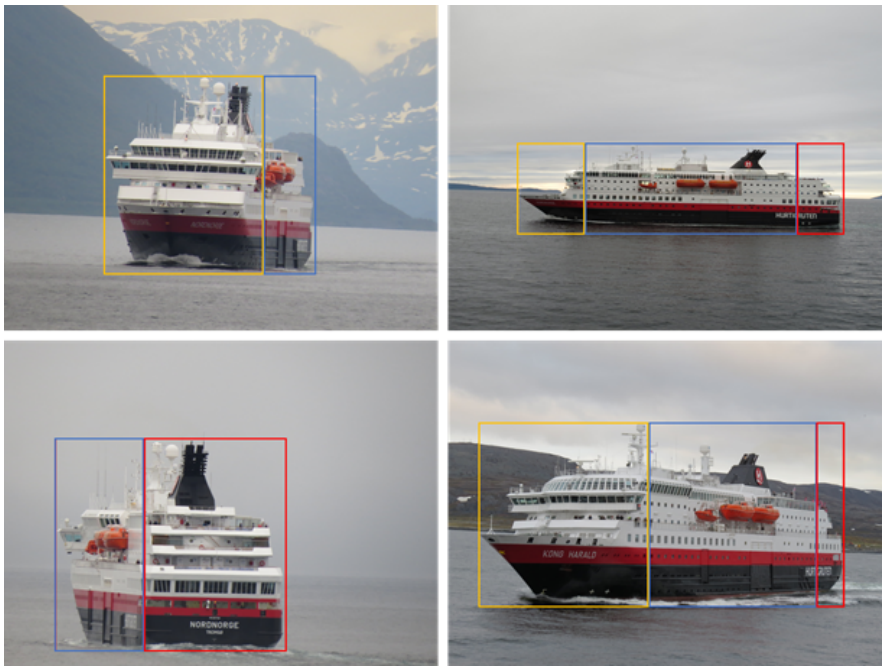


Figure 3.6: Dividing ship into three parts from different viewpoints. "Front" are represented by yellow bounding box, "side" as blue, and "back" as red

During labeling the MergedDataset, some of the points from the VOC Annotation Guidelines[4] have been used. These are:

- What to label
  - All objects of the defined categories, unless:
    - you are unsure what the object is.
    - the object is very small (at your discretion).
    - less than 10-20 % of the object is visible.
- Bounding box
  - Mark the bounding box of the visible area of the object (not the estimated total extent of the object). Bounding box should contain all visible pixels, except where the bounding box would have to be made excessively large to include a few additional pixels (<5 %) e.g. a car aerial.

3.4 Extraction of features

The purpose is to get features which are typical for both ship and non-ship. A large part of the project has been to find good features from each category. It is therefore important to use feature extractors that can detect and describe these features. Based on this, it has been important to test different feature extraction methods.

Decomposing an image into features is a widely used technique in Computer Vision. A popular approach is to use local interest points as features. There is also other approaches, like sliding window, to extract features. During the proposed method these two methods has been used and tested for feature extraction. In the first step, image patches are extracted from all training images using a feature detector. Then all features are represented as a compact feature vector. This step is called feature description. In the following sections the approach for extracting features from images are presented.

3.4.1 Approach 1: Interest point feature extraction

In this section, the feature extraction method done with interest point extractors is presented. The feature extractor used in this thesis is the Scale-Invariant Feature Transform (SIFT).

[4] http://host.robots.ox.ac.uk/pascal/VOC/voc2011/guidelines.html

Feature detection

As mentioned in Section 2.1, the ideal keypoint detector finds image regions that can be repeatably detected despite change of viewpoint. In other words, it is robust to all possible image transformations. Similarly, the ideal interest point descriptor should capture the most important and distinctive information of the image region.

Based on the specialization project, the Scale-Invariant Feature Transform (SIFT) was used as feature extractor in this thesis. SIFT was also chosen as an extractor because of the good performance and results within image recognition [25] [23]. To be able to extract good features from the image, it was conducted experiments with different parameter values of SIFT from the OpenCV library. Different parameter values were also tested during the specialization project, however some tests were performed once more to visualize the corresponding image patches of the interest points. Visualization of image patches was not achieved during the specialization project. The parameter values for the SIFT extractor is shown in Table 3.6. These are the default values used in Lowe [23], except from *contrastThreshold*. This parameter was set to 0.06 to reduce the number of interest points detected on the background.

Table 3.6: Overview over parameter values for SIFT in OpenCV

SIFT parameter values	
nfeatures	0
nOctaveLayers	4
contrastThreshold	0.06
edgeThreshold	5
sigma	1.6

Feature descriptor

After features are detected from the training images, descriptors were computed to create a compact representation of each feature. Descriptors are computed as presented in Section 2.1.2. To be able to visualize the features that are extracted with the interest point detectors, image patches were created based on the interest point's centre coordinates and size. Image patches were used for visualization since it it not possible to visualize descriptors directly. Further in the report, a feature of the image will be represented by the descriptor.

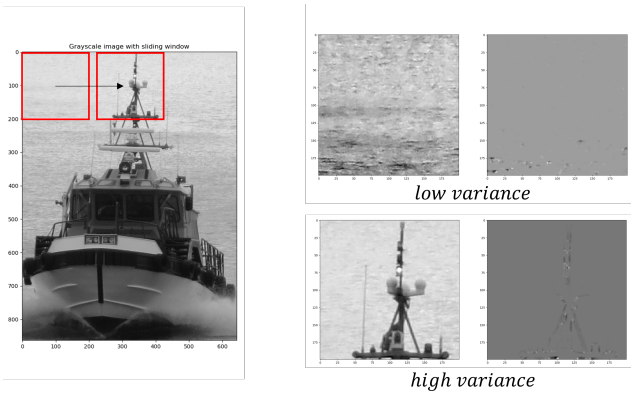


Figure 3.7: Principle behind variance threshold

3.4.2 Approach 2: Feature extraction with sliding window

Feature detection

To be able to gather data for training, or gather features of ships, a sliding window was used over the training images. The sliding window technique extracted features in three steps. At first every window was inspected. The window size was fixed with a size of 64x64 pixels. The size was selected according to what the feature descriptor computation expects. The window was moved with a stride of 1/2 of the window size to cover all the potential features of the image. In addition, it was tested with a stride of the same size as the window to avoid overlapping features. Sliding window was used to extract potential image patches with interesting features. Since the window was fixed, images were downsampled to create an image scale pyramid and the window was sliced over again to find features of different sizes.

As the window was iterated through the image, the variance of the intensity values was computed for each window. The window was stored as an image patch if the variance score was above an empirically chosen threshold. The threshold was used as an operator to exclude patches with less content. The principle behind this technique is illustrated in Figure 3.7. The variance threshold was selected based on two techniques. The first technique created a histogram of the variance frequencies from the whole training set. The histogram was used to get an intuition of where to initially set the threshold. In addition, the effect of the window selection was visualized by copying all pixel values inside of the above-threshold window into an output image. The output image was predefined as an image with 255 pixel values only and with same size as input image. By using these output images the threshold was effectively selected for the whole dataset by looking at which patch were extracted. This made it possible to visualize the features that were extracted. Each image patch from the ship and non-ship training set were stored in separate lists dependent on the image label.

Feature descriptor

In the same way as with interest point extractor, a compact representation of each patch was created. Histogram of Oriented Gradients (HoG) was used as descriptor. As mentioned in Section 2.2.1, HoG computes the histogram of gradients for the image. This is suitable when extracting features with sliding window since the descriptor is not dependent on interest points.

The HoG descriptor is implemented on each patch by using scikit-image library in Python where the default values are used. These parameter values were selected based on the recommendation from the paper by Dalal and Triggs [11]. With the default values, the images is divided into 8x8 cells where the histogram of gradients is computed for each cell. Orientations of the gradients are divided into 9 bins. The HoG features were extracted from 64x64 image patches. The patch can have any size, however the only constraint is that the patches being analyzed have a fixed aspect ratio. These parameter settings resulted in a 1764 dimensional real value vector. The HoG is not a scale invariant descriptor. Typically, scale invariant detectors run the feature detector at different image scales. To make the HoG descriptor invariant to scale the sliding window was implemented with image pyramid.

3.5 Generation of Vocabulary using K-means clustering

Before a classifier can be used, it must be trained with several samples of target objects. As such, a training database is built using a vocabulary of visual words. This vocabulary is made out of descriptors computed at the previous step. The approach for generating vocabulary is the same for both extractor methods explained in Section 3.4. Generation of vocabulary is the final step for training the BoVW model.

To generate the vocabulary the k-means algorithm is initiated with the descriptor lists for both categories. It was generated a vocabulary for each category. This is done such that the model are able to differentiate between positive and negative target objects. Excising work usually determine the vocabulary size empirically and the sizes varies from hundreds to thousands depending on the dataset. There exists no method or theory on how to guide the selection of the optimal vocabulary size [18]. Therefore, it was tested with different values of k to find the desired vocabulary size for both ship and non-ship datasets. The desired vocabularies were determined based on how good the features were clustered. A good vocabulary would have clusters with similar features. In addition, it is necessary to select a vocabulary size that produces the best performance. It has been proven that a large vocabulary tends to improve matching accuracy. However, this does not mean that a vocabulary with a large size guarantee higher matching accuracy. Yang et al. [47] proved with extensive experiments that the matching accuracy first rises

dramatically when increasing the size, then the accuracy peaks, and after that levels off or drops. Not only does a large vocabulary increase the computation load in clustering and matching, it can also impact the performance negatively.

Both *k*-means models were initialized with *k-means++* method. As mentioned in Section 2.3.1, *k-means++* is recommended since it is better at avoiding poor clustering than the standard, random *k*-means initial values. The desired vocabulary was chosen based on the content of the image patches, where it was important to get informative features. As mentioned earlier, feature detector was an important factor. In theory, a visual world is defined as the cluster centre. This is a mean vector computed to fit all the descriptors of the corresponding cluster. Therefore, it was of interest to visualize these cluster centres since they represent the vocabulary. However, since the mean vector is a real-valued feature vector it is not possible to directly visualize the vocabulary. To be able to visualize it, Euclidean distance was used to measure the similarity between the cluster centre with its corresponding clustered descriptors. In addition, represent visual words of the vocabulary as a representation of multiple similar image patches to the cluster centre. To achieve this the top 20 most similar descriptors to the cluster centre was returned. In this way, one can visualize the corresponding image patch to the closest descriptors to be able to illustrate the content of the vocabulary. The vocabulary is visualized by showing the 20 most similar patches for 10 random clusters. In this way it is possible to get an intuition of what pattern the patches consists of for some of the clusters.

Generating vocabulary is the final, crucial step of training the BoVW model. The vocabulary is used as a dictionary of words representing ship features. Based on this, new query descriptors can be compared to the training set and be classified based on similarity. It was created one vocabulary for each category to be able to compare descriptors to each vocabulary. This will be further described in the following sections.

3.6 Descriptor Matching with Nearest Neighbor

Once a vocabulary is generated for each category, features that are similar between the vocabularies are removed. Many features from an image containing a ship will have similarities to features from buildings. Therefore, it would be useful to have a way to discard features that do not have unique pattern typical for ship or non-ship, respectively. In this way it will be easier to distinguish between the ship or non-ship category since descriptors from the ship vocabulary could have similar features to the non-ship, and vice versa. An effective measure was obtained by comparing the distance of the closest neighbor of a ship descriptor with a non-ship descriptor using nearest neighbor. The nearest neighbor is defined as the descriptor from one

vocabulary with minimum Euclidean distance to a descriptor in another vocabulary. If the two descriptors are similar, with a distance less than a empirically chosen threshold, then the ship descriptor is removed from the vocabulary. This distance threshold was chosen based on observation. This is done for all descriptors in ship vocabulary and non-ship vocabulary.

3.7 Training Bag-of-Visual-Words model

Training a Bag-of-visual-words model consist of 3 steps: feature extraction, feature description and vocabulary generation. All steps have been presented in details during previous sections. This section is intended to get a better understanding of the whole training procedure used in the project by combining previous steps.

Initially, images are retrieved from a folder on the computer containing all training images and corresponding annotation files (.xml). The corresponding image and annotation file have the same name to know which file belongs to which image. Before training, each image of the training set is pre-processed. The first step of the pre-processing was to convert images to grayscale. Grayscale is used since it is more computationally efficient to work with single channel images compared to 3 channel RGB color images. After this all objects were cropped out from the training dataset based on their ground truth boxes in the annotation files. The reason behind this is that many images contain multiple objects and a lot of background material. If one would use feature extraction over these images, it would be harder to distinguish between target objects and other objects that are not of interest. Feature extraction is the most important step of the approach. Therefore, it is important to extract the best features from the training set. This results in a training set consisting of images with only one object and nothing else. The cropped images were separated into the corresponding classes, ship or non-ship, according to their label. How these images are separated based on its label is described in Section 3.3.1. The pre-processing steps are illustrated in Figure 3.8.

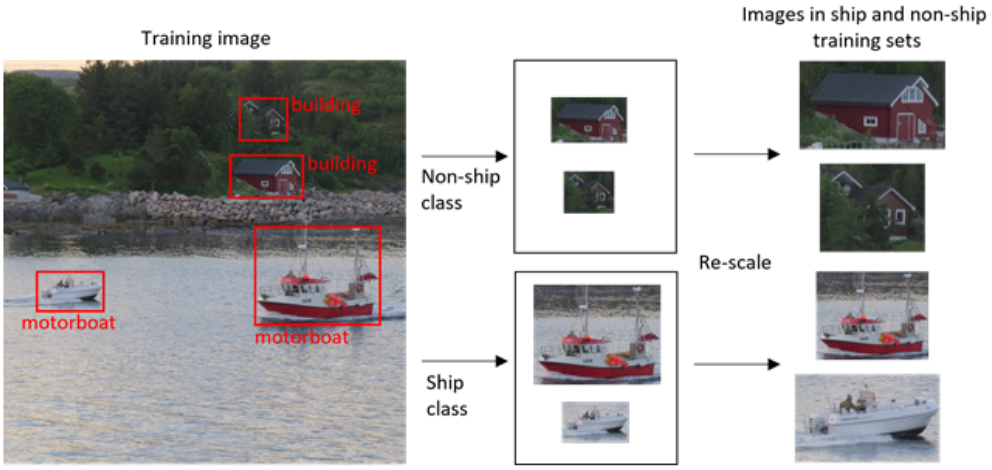


Figure 3.8: Illustration of how training images are preprocessed to gather new training set for feature extraction

Since the ship appear at different locations on the image, the resulting cropped images will have different resolutions. Therefore, the next step was to re-scale the new images to approximately same size. The images are re-scaled based on their image height. The aim was to get all images to have height of 650 pixels. The ratio between the image height and image width was unchanged. Images with a higher height would be downscaled, while images less than would be upscaled. The decisions are listed below.

- If image height is larger than 650, then downscale image height to 650 pixels.
- If image height is between 150 and 350, then upscale image height to 650.
- If image height is less than 150, then exclude the image from training set since the image is too small.
- The rest are unchanged

After images were pre-processed, feature extraction was performed to collect a set of interesting features from the ship and non-ship category separately. The classifier must be trained with several samples of the target objects. As presented in Section 3.4, features are extracted with two different approaches. When using the SIFT interest point detector, image patches were gathered based on the location and size of each detected interest point. Since the SIFT detector does not enable parameter change of patch size, each interest point are evaluated and if its size is too small, then that interest point was excluded. The size of the interest point is referred to as the diameter. It was desired to get image patches that was understandable for human assessment. With this it is desirable to get features that describe the ship part more than just an

edge or corner. The image patches are then converted to a compact representation using the SIFT descriptor method. A list of descriptors are stored for each category. The second approach used for feature extraction was sliding window. During this step, the sliding window iteratively went through each image. An image pyramid with different scales were used to be able to gather features of different sizes. During the iteration, each window is inspected. The variance of the intensity values of the window is computed and if the value is higher than a given threshold, then the window is stored as an image patch as illustrated in Section 3.4.2. Features are extracted and separated into the corresponding classes according to the ground truth masks and then the descriptors are computed using HoG.

The last step of training the BoVW model is to generate a vocabulary by clustering similar descriptors from the training images into k number of clusters. The results from training is a vocabulary consisting of similar descriptors in each cluster. The cluster centres will inform the classifier to which class the testing samples belongs. These vocabularies would later be used during testing to classify image patches into ship or non-ship.

3.8 Testing of model

The testing approach is done for one test image at a time. After the vocabulary is generated, it is used to classify test image features. First, images and annotation files are retrieved from a folder on the computer containing test images. Then, images are re-scaled to the same size with the same decisions done during training. During testing, the ground truth box will later be used for performance evaluation. The ground truth is thus resized based on the test image and the (x, y, xmax, ymax) image coordinates are stored in array.

Testing has the same steps as training. The first step after pre-processing is feature detection. Then descriptors are computed for each image patch of the test image. After this, each descriptor is matched with both vocabularies to evaluate which category the descriptor is closest to. The distance between the test descriptor and each cluster centre is computed. As mentioned in Section 3.6, this is done with Nearest Neighbor algorithm with FLANN library to increase efficiency. The distance for the descriptor is computed for both vocabularies.

3.8.1 Classification of Image Features

The next step of the test approach is to classify each descriptor into one of the two categories. The distance between the descriptor to each cluster centre in ship vocabulary is compared with the distance to the cluster centres of non-ship vocabulary. Equation 3.1 show the computation of the distance between descriptor and cluster centres. $dist^{t}_{jk}$ ($dist^{b}_{jk}$) is the distance between the j-th descriptor and the k-th ship (non-ship) cluster centre, $desc_j(n)$ is the n-th element of the j-th descriptor, $cent_k(n)$ is the n-th element of the k-th cluster centre, and p is the size of the descriptor vector. t is ship and b is non-ship. The descriptor is classified as either ship or non-ship depending on which vocabulary it is closest to regarding distances.

$$dist^{t,b}_{jk} = \sum_{n=1}^{p} \left[desc_j(n) - cent^{t,b}_{k} \right]^2 \tag{3.1}$$

It is desirable to classify descriptors into one of the categories only if there is some certainty that the classification is correct. By this it is meant that the distances to the ship and non-ship vocabulary should be different from each other. If the distances are similar to each other, it is not obvious if the descriptor is a ship or a non-ship. This decision is shown in Equation 3.2.

$$\text{decision} = \begin{cases} \text{ship}, & \min[dist^{t}_{jk}] << \min[dist^{b}_{jk}] \\ \text{non-ship} & \min[dist^{t}_{jk}] >> \min[dist^{b}_{jk}] \\ \text{not classified} & \text{otherwise} \end{cases} \tag{3.2}$$

where $\min[dist^{t}_{jk}]$ ($\min[dist^{b}_{jk}]$) is the smallest distance value between the descriptor and the ship (non-ship) centroids. The much larger sign is defined by a threshold T. A descriptor belong to ship if the distance between descriptor and ship vocabulary is larger than distance to non-ship, and the difference between the distances are larger than T. Considering that the descriptor is a compact representation of an image patch, image patches are also classified. For each test image, the number of ship image patches and non-ship image patches was counted. The image is classified based on the number of patches in each category. If most of the patches in the image is classified as ship, then the object is classified as ship.

3.8.2 Detection of Ship

For ship detection it was first experimented with images consisting of one object and no background noise to evaluate if the model was able to distinguish between ship and non-ship objects. Then it was tested on real scenario images consisting of multiple objects and categories. During testing is was used 9 unseen test images to evaluate the performance of the model. The model was evaluated with different vocabularies consisting of SIFT and HoG descriptors.

For the single-object images, a predicted bounding box was generated by enclosing all the correctly classified patches. The predicted bounding box was considered as the minimum rectangle that encloses all classified patches. The detector was evaluated based on a confidence score and Intersection over Union (IoU). The confidence score reflects how likely the predicted bounding box contains an objects. The score was computed from the ratio between the total number of patches and the number of classified patches. So if the image consists of mostly ship classified patches, then the confidence score is defined as:

$$\text{confidence score} = \frac{\text{Number of classified patches}}{\text{Total number of patches}} \tag{3.3}$$

Then IoU was used to evaluate the localization of the detected object. The results are visualized on three different output images; (1) An image visualizing the original image with the extracted patches and their classified label. (2) An image showing only the patches classified as ship. (3) An image showing the input image with predicted bounding box and ground truth box.

These evaluation measurements are only implemented for single-object detection. When evaluating the multi-object images, it was only used output images to visualize the classification of the image patches. Thus, image (1) and (2) was used for visualization. It was not created a complete algorithm for detecting multiple objects in the image due to complications when trying to implement it. Based on the observed results during experimentation, the author did not have time or programming experience to implement it. Thus, this was set as future work.

Chapter 4

Results and Discussion

In this chapter the results gathered from conducting experiments on the BoVW approach is introduced. The results gathered from feature extraction and vocabulary size experimentation will first be presented. Then, results from experimenting with ship detection will be given. Object detection is divided into two approaches; single-object classification and multi-object classification.

4.1 Feature extraction approaches

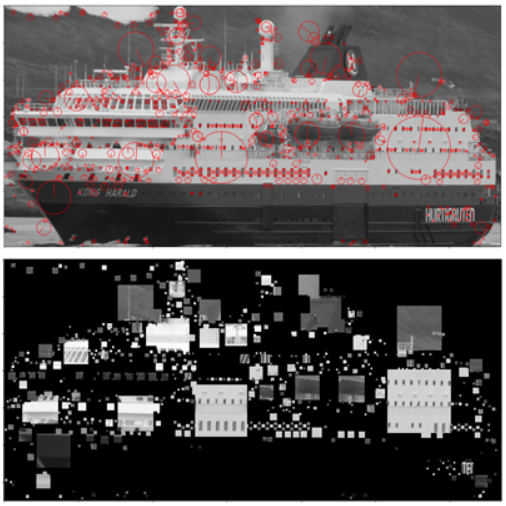
The best features from ship and non-ship categories were gathered through experiments. The aim of this section was to compare the two feature extraction methods to see which gave the best image features.

4.1.1 Interest point feature extraction with SIFT

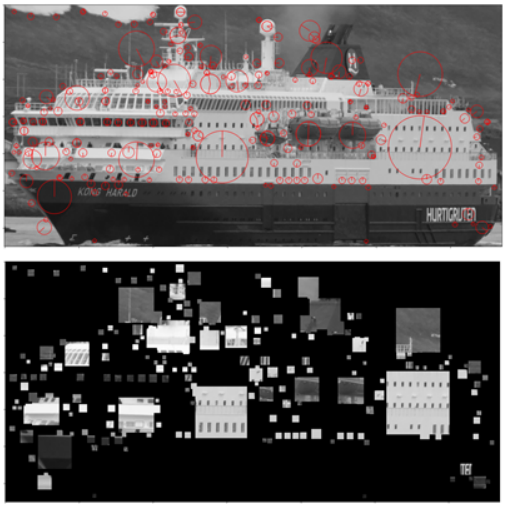
In the first approach, SIFT was used as feature extractor. Figure 4.1 illustrate where SIFT interest point are detected on different images. The interest points shown in the figure consists of a centre point and a red circle indicating its size. For this thesis it was of interest to extract parts of the ship that made sense for human assessment. This means that it should be possible to understand which part of the object the patch represent (e.g part of the mast or window). It was observed that many of the interest points represented only edges and corners, thus being too small in size.

Figure 4.1: Sample images from SIFT detector

To handle this the smallest interest points were removed by using a threshold on the size of the interest point. Figure 4.2 show the effect of using a threshold of 5 and 10. Utilizing this the interest points with a size smaller than the threshold were ignored. The bottom images of Figure 4.2a and 4.2b show only the extracted patches from the input image. By selecting a larger threshold, more interest point were removed which results in only the largest remaining. For the rest of the experiments, the threshold was set to 10.



(a) Interest points with a size larger than 5



(b) Interest points with a size larger than 10

Figure 4.2: Effect of filter out small image patches with threshold

SIFT interest points are detected at random and irrelevant locations which resulted in features that do not represent specific ship parts. In addition, only a few interest points were remaining after filtering out the smallest. This resulted in that not all features of the ship, or non-ship, is covered at all times. An illustration of extracted patches are shown in Figure 4.3.

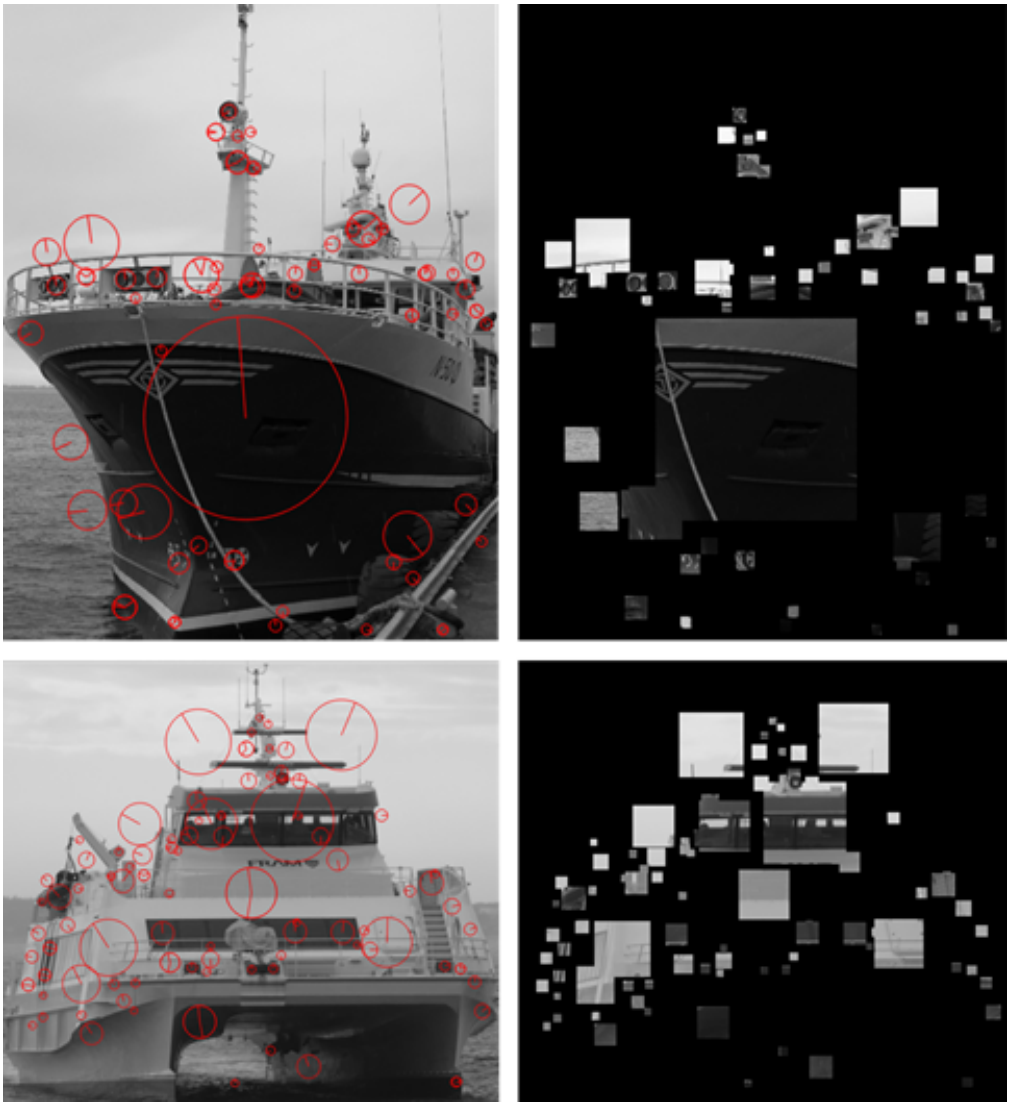


Figure 4.3: Sample images from SIFT detector. (Left) Detected interest points. (Right) Extracted patches from interest points

Based on the observations, it was concluded that SIFT as feature detector did not fit the dataset for this thesis. This was the main reason why sliding window technique was also implemented for feature extraction. Although, SIFT does not extract features that are of interest to the thesis, it computes a robust descriptor. The SIFT descriptor is the most used and most recommended descriptor for feature matching. It is invariant to image scale, rotation and change in illumi-

Table 4.1: Overview over features after feature extraction with interest point

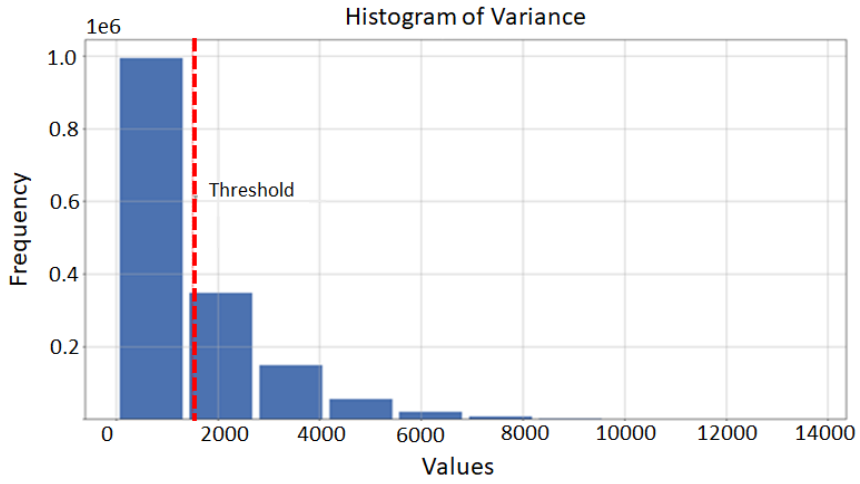
Feature extraction summary		
	Number of images	Number of features
Ship class	4146	206 755
Non-ship	2961	124 474

nation, noise, and minor changes in viewpoint. SIFT descriptor is also highly distinctive. The HoG descriptor is not invariant to these properties. Because of this, it was done some more tests with the interest point approach to see if the SIFT descriptor could perform better than sliding window with HoG.

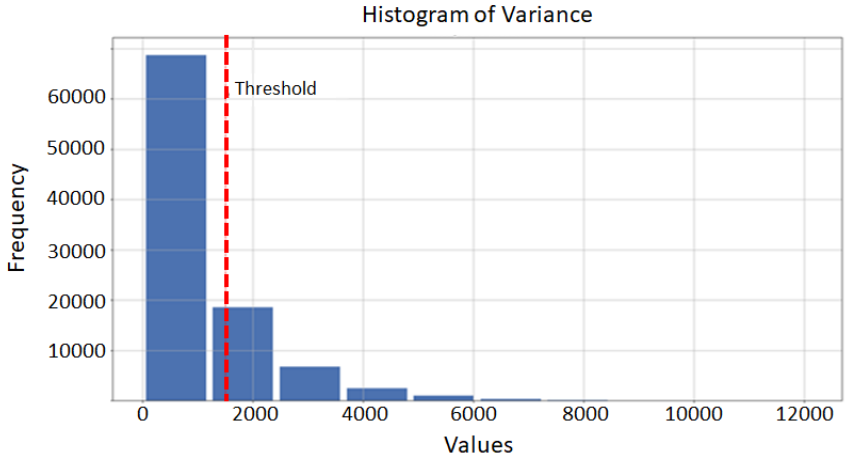
4.1.2 Feature Extraction using Sliding Window

In the next approach, feature extraction method using sliding window was used to slide through all regions of the image to extract potential features. A variance threshold was used to select image patches with interesting content.

As stated in Section 3.4.2, the variance threshold was empirically chosen based on two methods; a histogram of variance frequencies and images containing only the patches extracted with the threshold. The histograms from the ship and non-ship training set are shown in Figure 4.4, respectively. Based on the histogram, three different threshold values were tested; 1000, 1500 and 2000.



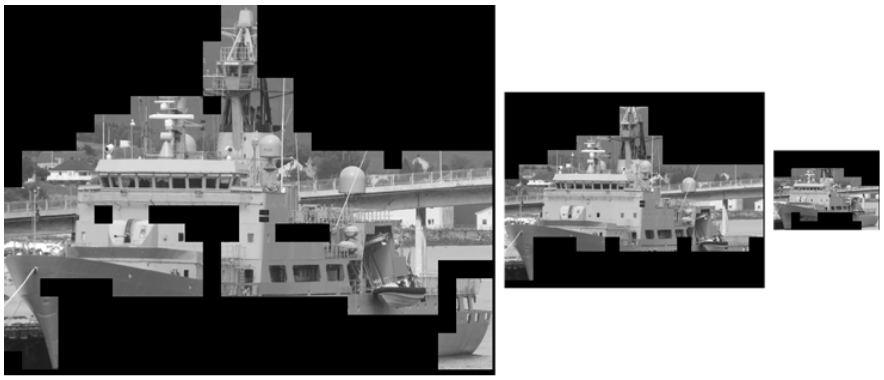
(a) Histogram of variances from ship training dataset



(b) Histogram of variances from non-ship training dataset

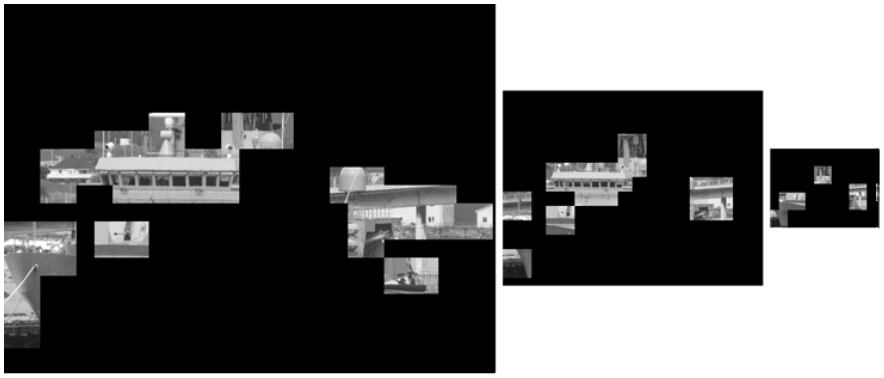
Figure 4.4: Histogram of variance frequencies. Used to determine variance threshold for the whole training dataset

Figure 4.5 illustrate the affect of selecting different threshold values. As expected, less patches were extracted with a high variance, while a lower variance extract more, but less interesting image patches. A threshold of 1500 was selected for both ship and non-ship image sets (shown as red, dotted line on histogram), since most of the target features were extracted without including too much from the background. A threshold of 1000 could be too low and resulted in more landscape elements being included in the training set, while 2000 could exclude too many of the target features. Wanted a threshold that would fit for the whole training set. Sample images are shown in Figure 4.6. The empirically chosen threshold also fits the results from the histogram since it removes the first bar with the lowest variances.



(a) Variance threshold of 1000

(b) Variance threshold of 1500



(c) Variance threshold of 2000

Figure 4.5: Effect of using different variance thresholds on a sample image. The original image is represented with different scales of the image pyramid.

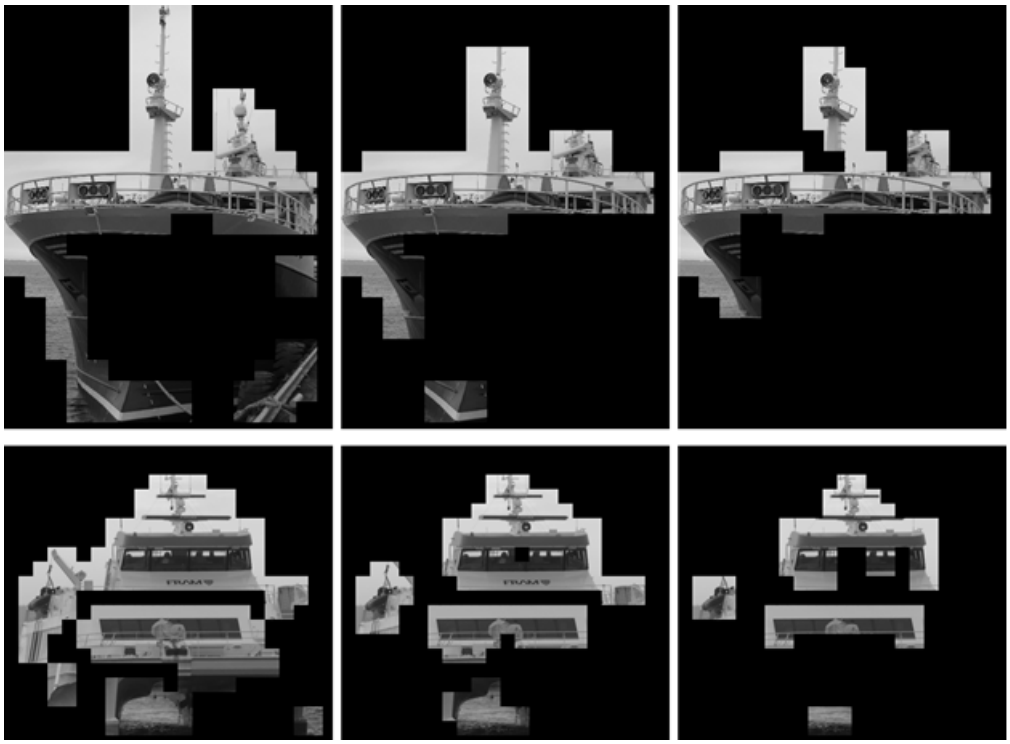
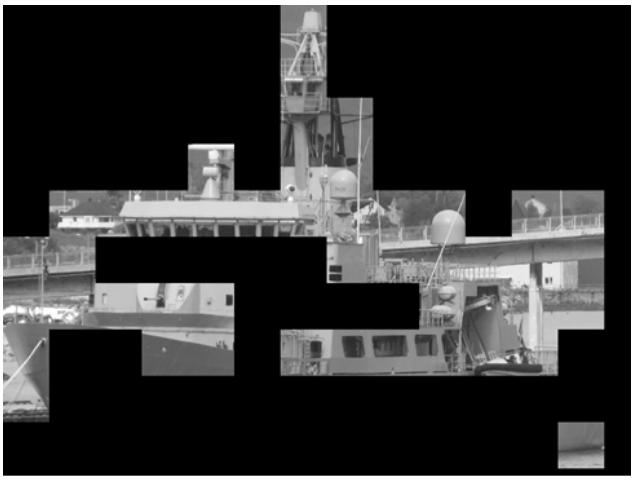
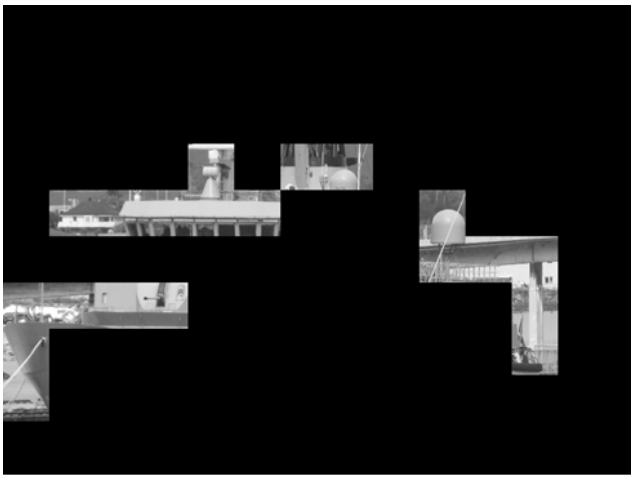


Figure 4.6: Sample image after different variance threshold extractions. From left: 1000, 1500, 2000.

During the feature extraction experimentation it was also tested with different step sizes of the sliding window. In the figures above it was used a step size of 1/2 of the window size. Figure 4.7 show how features are extracted using a step size as large as the window. Used to avoid overlapping image patches. However, based on the figure, less patches were extracted. A smaller step is recommended since more features of the image are analyzed. Because of this, the stride was selected to be 1/2 of the window size.



(a) Variance threshold of 1000



(b) Variance threshold of 1500

Figure 4.7: Effect of different variance thresholds on a sample image. Patches extracted with a step size equal to window size.

Therefore, it was decided to extract features using a sliding window with the following properties; window size of 64x64, step size of 1/2 of image size, and variance threshold of 1500. With these properties the desired patches were selected for training.

After the extraction of image patches was completed, a HoG descriptor was computed for each image patch. The window size resulted in a descriptor of length 1764. This is a large dimension which comes with a high computationally cost when computing descriptor for each feature. One way to deal with this is to use Principal Component Analysis (PCA) to reduce the dimension of the vector without loosing information. However, this was not implemented for this thesis.

Table 4.2 show an overview of the number of features gathered using the sliding window technique discussed above. The last column (right) show the number of features after removing

similar descriptors. Which in this thesis is referred to filtering of the descriptors. When using HoG it was observed that descriptors in the ship and non-ship vocabulary had similar characteristics in case of distance. Since the descriptor will later be used to generate a vocabulary it was desired to filter out the descriptors in the ship and non-ship feature set that had a small Euclidean distance. Figure 4.8 illustrate the difference in distance between similar and different descriptors.

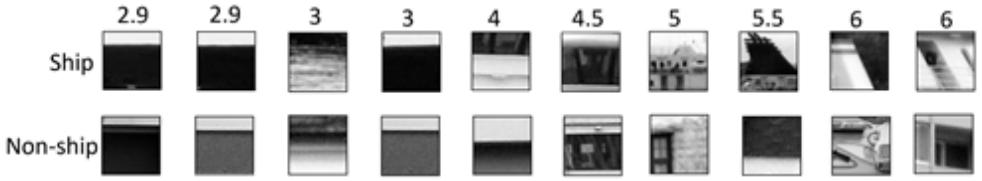


Figure 4.8: Distance between ship and non-ship patches. The numbers on top is the Euclidean distance between the shown image patches

Based on this figure it was determined to create a lower bound threshold to remove similar descriptors. The threshold was set to 4.5. This should develop training sets of unique ship and non-ship descriptors. By using this threshold, the number of features were reduced from 538 994 to 457 782. An overview is shown in Table 4.2.

Table 4.2: Overview over number of features after filtering out similar descriptors

Feature extraction summary			
	Number of images	Number of features	Number of features (after filtering)
Ship class	4146	538 994	457 782
Non-ship	2961	231 902	196 805

4.2 Selecting Vocabulary Size Experiments

Based on the two descriptor sets from computing HoG and SIFT on all image patches, a vocabulary is created from both descriptor types. In addition, a vocabulary is generated for both ship and non-ship descriptor lists. In this section, each vocabulary will be inspected.

4.2.1 Vocabulary Generation using SIFT descriptors

A more thorough experimentation has been done with the sliding window technique since SIFT did not gather desired ship and non-ship features. However, some tests were done to compare the constructed vocabularies.

Figure 4.9 show the vocabulary from clustering SIFT descriptors with 100 clusters. It can be shown that some of the clusters consists of similar features. Cluster number 2 to 4 from the top share a darker color. However, besides color there is not much similarities between the patches in each cluster. In addition, it is not possible to determine what kind of ship features are presented based on the content of the patches. Therefore, it was tested with a much larger vocabulary size for both ship and non-ship to see if it would generate better clusters. This is shown in Figure 4.10.

Figure 4.9: Vocabulary from ship descriptors. $k = 100$

(a) Vocabulary from ship descriptors, $k = 500$

(b) Vocabulary from non-ship descriptors, $k = 500$

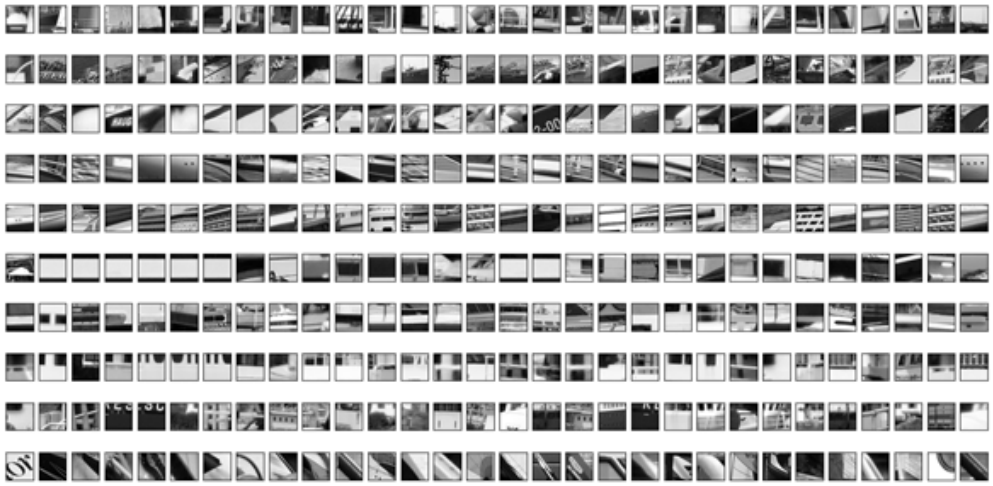
Figure 4.10: Vocabulary generated from SIFT descriptors

As observed, the clusters did not improve with a vocabulary of size $k = 500$. From the vocabularies it is observed that there is no similarities between the image patches. Neither of the ship or non-ship vocabulary give any information about the features that were clustered. The reason behind this could be that it was selected a k-value that were too large for the SIFT descriptors. The aim of the k-means cluster algorithm was to cluster descriptors that share similar properties. Unfortunately, this was not achieved with SIFT. Based on this, it was concluded that SIFT features did not fit for this project and thus no more experiments of this approach was conducted. However, it was done small tests of object detection to check the performance of the model with vocabularies of size $k = 100, 500, 5000$. This is later shown in the chapter.

Together with the feature extraction part, the creation of the vocabulary was the most time consuming step of the project regarding implementation. The author worked first on implementing feature extraction using interest point extractors. Unfortunately, this created a deviation in the progress since SIFT did not give desired results. It was used too much time on trying to implement and improve the vocabulary with SIFT descriptors. Therefore, it was necessary to start with a new approach using sliding window and experiment with different parameters of that approach. This made less time for the next step of the method which is implementing a test approach of object detection.

4.2.2 Vocabulary Generation using HoG descriptors

Figure 4.11 show vocabularies of size $k = 100$ from ship and non-ship. The following vocabularies were generated before filtering descriptors was implemented.



(a) Ship vocabulary, $k = 100$

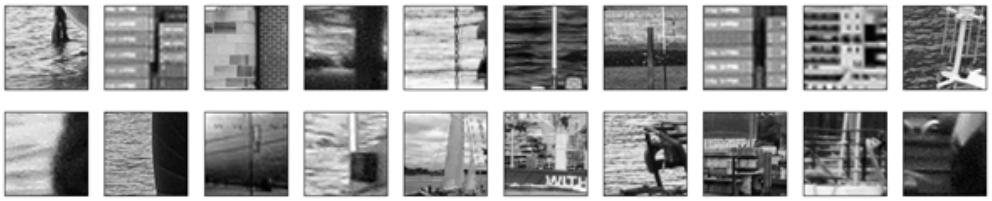
(b) Non-ship vocabulary, $k = 100$

Figure 4.11: Vocabulary generated from HoG descriptors

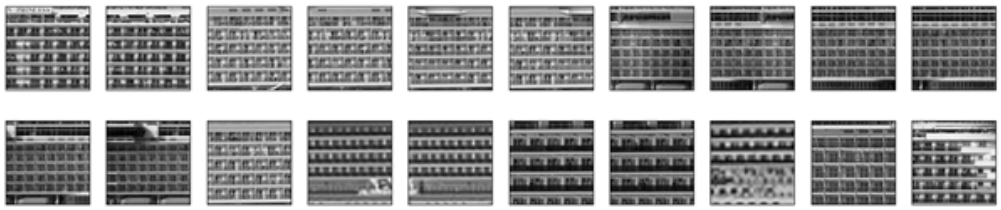
From these figures it was more obvious to see what kind of pattern each cluster represent. Figure 4.12 illustrate 10 clusters with similar image patches. This vocabulary indicate that k-means manage to group similar patches based on the relative distances of the defined features.

Figure 4.12: Top 10 best visual clusters from ship vocabulary, $k = 100$

Figure 4.13 show samples of clusters where patches are different and similar to each others, respectively.



(a) Samples of clusters with not so similar patches



(b) Samples of clusters with similar patches

Figure 4.13: Cluster samples from ship vocabulary, $k = 100$

Based on the figure above, the clustering achieved desirable results. Unfortunately, this was tested before object detection was implemented. Thus, object classification and localization is not tested with these vocabularies.

The final vocabularies, which are generated after filtering out similar descriptors, are shown in Figure 4.14 to Figure 4.16. It is observed that the vocabularies did not cluster similar patches like the previous vocabulary did. The reason for this change is that many of the patches in the

"good" clusters shown in 4.12 are similar to non-ship vocabulary. Patches in clusters 2, 3, 6, and 7 have similarities to the non-ship descriptors. This was illustrated in Figure 4.8 where it was shown the difference between distances. Therefore, there will not be as many similar patches as compared to before filtering. With these vocabularies the model performance was tested on unseen images later in the chapter.

(a) Ship vocabulary, $k = 100$

(b) Non-ship vocabulary, $k = 100$

Figure 4.14: Vocabulary using sliding window, $k = 100$

(a) Ship vocabulary, $k = 500$

(b) Non-ship vocabulary, $k = 500$

Figure 4.15: Vocabulary using sliding window, $k = 500$

(a) Ship vocabulary, $k = 1000$

(b) Non-ship vocabulary, $k = 1000$

Figure 4.16: Vocabulary using sliding window, $k = 1000$

4.3 Comparison between sliding window and interest point approaches

After implementing the different feature extraction methods some remarks are made. It was observed that sliding window gather more features from the target objects compared to SIFT feature extraction. Numerous features are desired since it gathers a larger training set. In addition, the features extracted with sliding window represented specific parts of the ship and non-ship images. The SIFT extractor extracted local features (i.e edges and corners) which were not ideal to use as training samples. The local features were extracted from random places of the

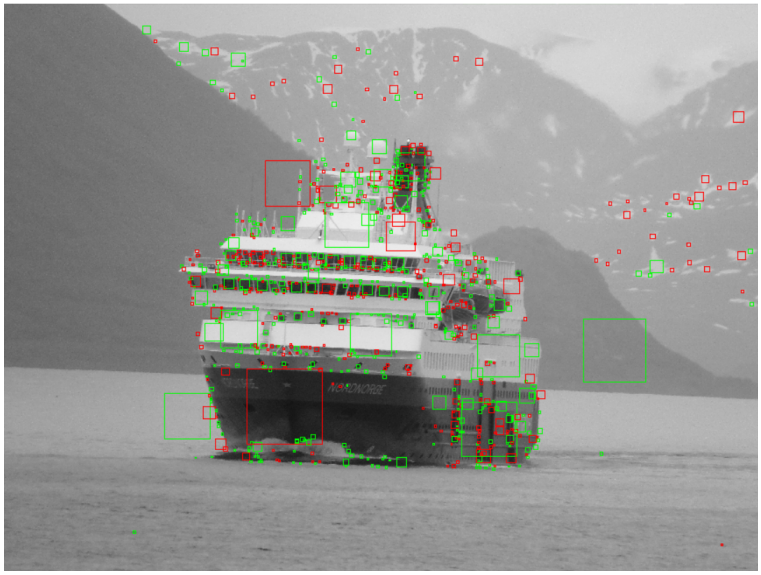
target object. Sliding window give the opportunity to decide where to extract patches since it iterates over all regions of the image and store image patches with a variance higher than a given threshold. Sliding window is the method of choice for feature extraction in the BoVW approach.

4.4 Ship Detection using Bag-of-Visual-Words

4.4.1 Object Detection with Interest Point approach

Figure 4.17 show samples images from testing dataset. The green and red rectangles indicate the extracted interest points detected by SIFT. Green rectangular boxes indicate that the interest point is classified as ship, while red indicate non-ship classification. Based on both figures it is observed that it is challenging to define if the given object was a ship or not. In Figure 4.17a it is not easy to determine if it is a building since too many green boxes are detected on the object. Figure 4.17b show that most of the interest points detected on the ship is classified as ship. And also, most of the interest points detected at the mountain is classified as non-ship. This indicate that the interest point approach for the most part can be able to detect ship targets, however more tests and better tuning is needed.

(a) $k = 100$



(b) $k = 100$

Figure 4.17: Interest point classification using vocabulary of size 100 on images consisting of only ship or non-ship

To make this happen, some changes was needed. It was implemented a threshold value to the classification decision. If a descriptor was similar to both ship and non-ship vocabulary then the descriptor would not be classified as either ship or non-ship. The patch would only be illustrated as a blue rectangular box. The Figures 4.18 to 4.19 are used to illustrate the results of classifying the descriptors with different vocabulary sizes.

Figure 4.18: Interest point classification on image consisting of multiple objects. $k = 5000$

(a) $k = 500$

(b) $k = 5000$

Figure 4.19: Comparison between interest point classification using different vocabulary sizes

Comparing Figure 4.19a with Figure 4.19b it is observed that there is some improvements when increasing the vocabulary size. However, more tests are needed to make a conclusion. This can be seen at the front of the ship. Other than that, the results are not optimal for ship detection. That is expected considering the lack of experimentation's done with this approach. More results from single-object classification are shown in Appendix A while multi-object are shown in Appendix B.

4.4.2 Binary classification experiment

In this section the results from single-object detection with sliding window is presented.

The algorithm was first tested with a sample image shown in Figure 4.20. It is observed that there are too many misclassifications. The model is struggling to distinguish between ship and non-ship. Especially on the buildings in the background. Based on this it was implemented a threshold like it was done with interest point approach.

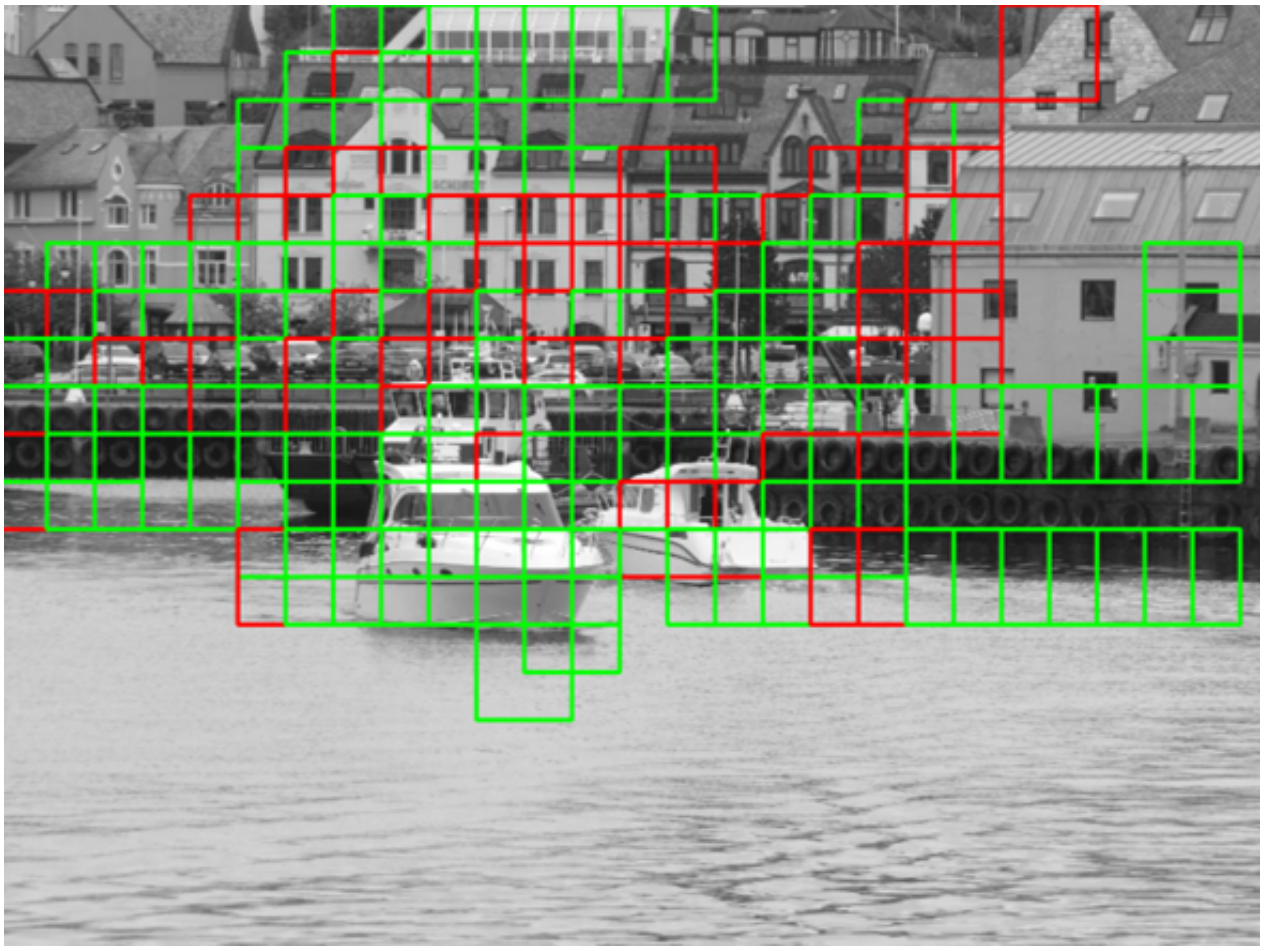


Figure 4.20: Sample image illustrating classified image patches without threshold. Illustrate the motivation between threshold

The model performance using a vocabulary of size 100 and with a threshold of 0.1 is illustrated in Figure 4.21 to 4.24. Each image contains a sub-image illustrating the classification of the extracted image patches, sub-image showing only patches classified as ship, and sub-image with ground truth and predicted bounding box. In the figures of non-ship it is observed that most of the patches are either classified as non-ship or not classified at all (blue rectangles). However, in Figure 4.22, too many patches are classified as ship as seen in the middle sub-image.

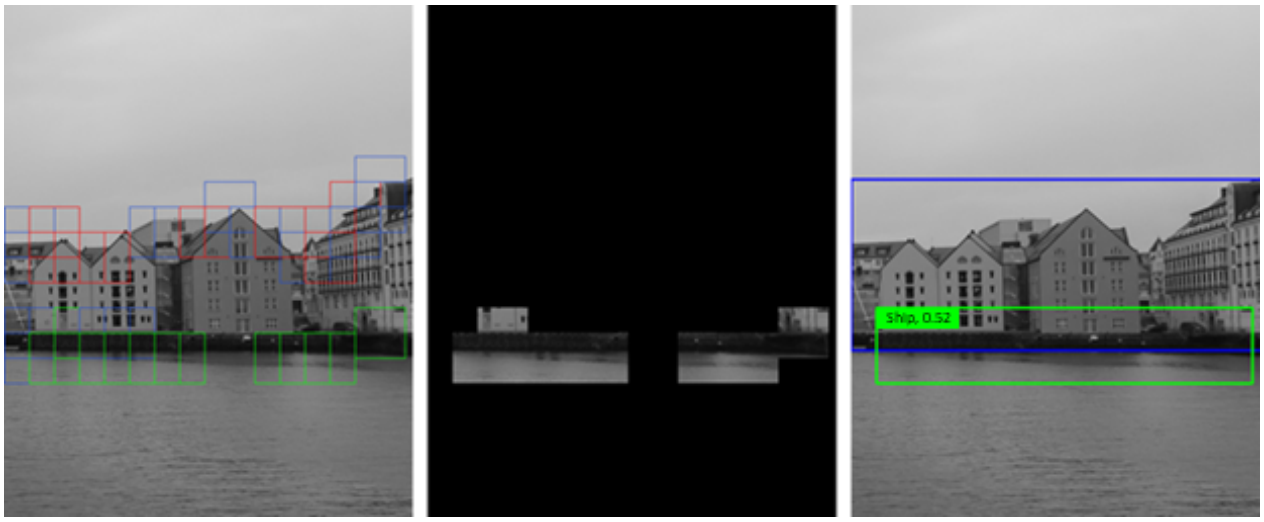


Figure 4.21: Non-ship classification, $k = 100$

Figure 4.22: Non- ship classification, $k = 100$

Figure 4.23 and 4.24 show two images consisting of a single vessel. For these images, most of the patches are classified as ship. In Figure 4.23, 40 patches are classified as ship, while 5 as non-ship. This results in a confidence score of 0.89 as seen in the right sub-image. A majority of the patches are extracted from the ship, resulting in a predicted bounding box with a IoU of 0.83. The left sub-image of Figure 4.24 consist of 28 ship patches and 5 non-ship patches giving a confidence score of 0.85. The placement of the classified patches give a IoU of 0.9.

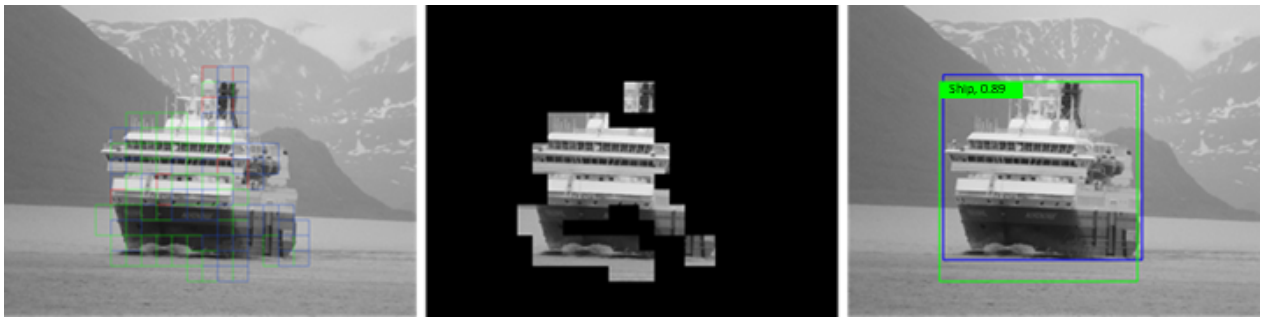


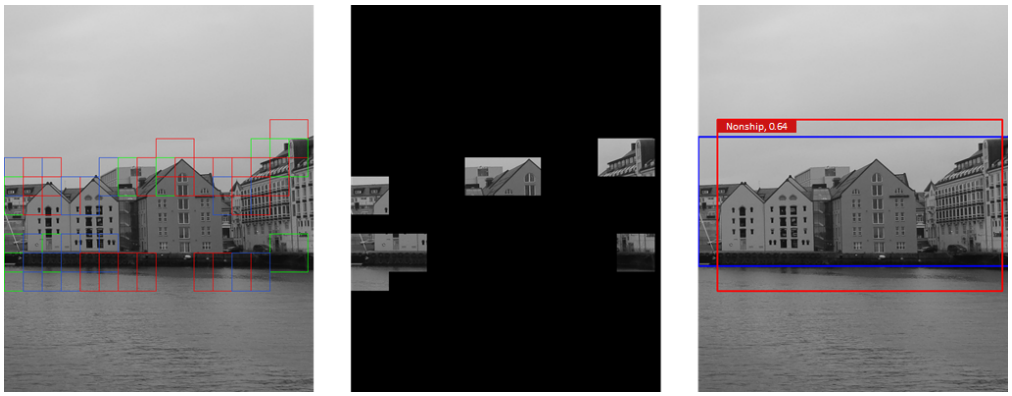
Figure 4.23: Ship classification, $k = 100$

Figure 4.24: Ship classification, $k = 100$

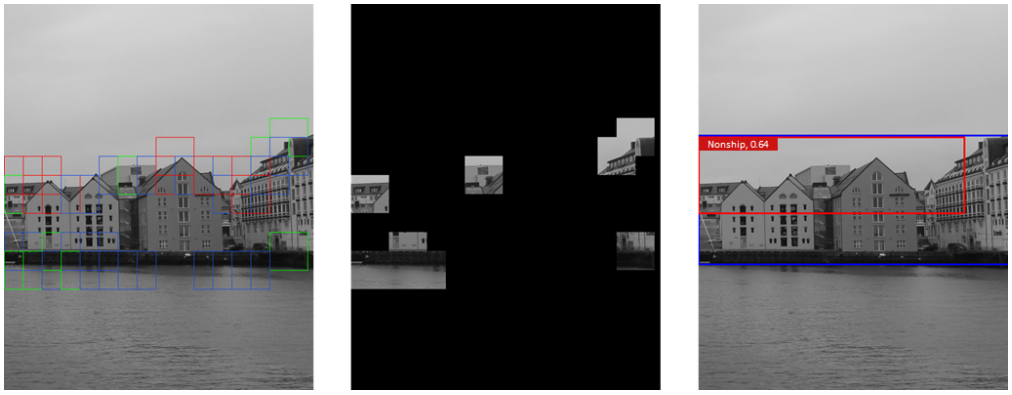
These images give promising results, however misclassification can occur according to Figure 4.25. For this image 20 patches were classified as ship, while 24 were classified as non-ship. This gives a confidence score of 0.55. In addition, the placement of patches generated a predicted bounding box with a IoU of 0.7. Fortunately, the confidence score was lower compared to the correctly classified images. This illustrate that the confidence score can be used to detect false positive detections by using a threshold on the score.

Figure 4.25: Sample of wrongly classified object, $k = 100$

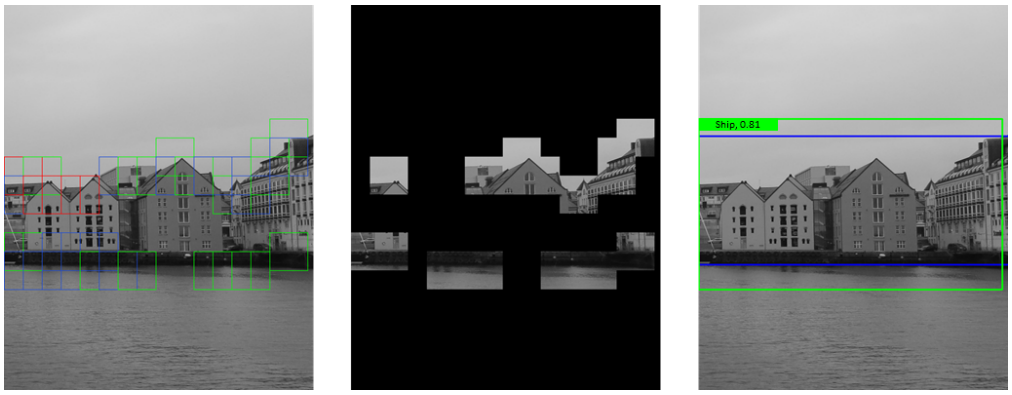
In the following Figures it is illustrated the effect of increasing the vocabulary size. In the above experiment it was used $k = 100$, in the next experiment it was tested with $k = 500, 1000, 5000$.



(a) $k = 500$



(b) $k = 1000$



(c) $k = 5000$

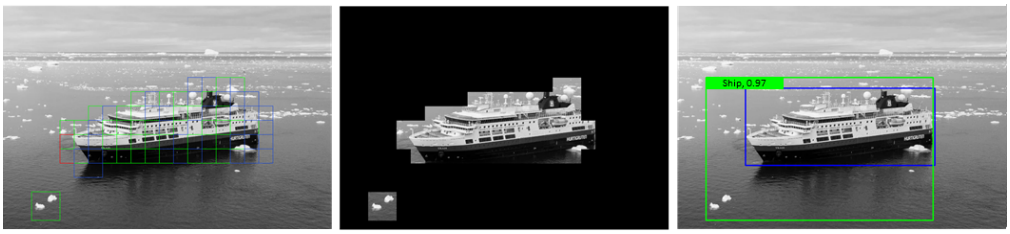
Figure 4.26: Comparison of non-ship detection when increasing vocabulary size

Table 4.3: Overview over classification for non-ship

Classification summary			
Vocabulary size	500	1000	5000
Number positive patches	9	9	17
Number negative patches	16	9	4
Confidence score	0.64	0.5	0.81
IoU	0.7	0.51	0.74

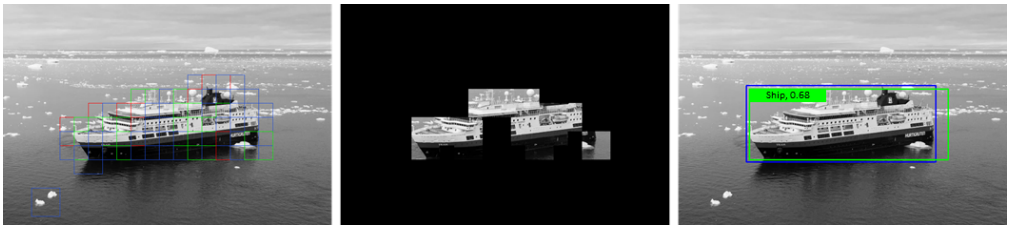
Increasing vocabulary size for non-ship does not necessary improve classification of the patches. Using a very large vocabulary size of 5000 decreased the performance in a way where more patches were wrongly classified as ship. This is illustrated in the middle sub-image of Figure 4.27c. The best result is shown in Figure 4.26a with a vocabulary size of 500. For this image, 9 patches were classified as ship while 16 as non-ship. Resulting in a confidence score of 0.64 and IoU of 0.7. The model have more challenges with classifying non-ship patches compared to ship. The model is more uncertain on these features giving a lower confidence score on the object classification. The reason for this could be that some patches of building is similar to those of ship. In addition, the non-ship category has less training samples. One way to deal with this is to gather more training samples of non-ship. This was set as future work. A large dataset is crucial in object detection tasks. Refer to Appendix A to see more results from the classification on different test images. Another way of handling the poor classification is to use Support Vector Machine (SVM) instead of Nearest Neighbor to distinguish between classes within the dataset. The affect of this need to be tested, and thus set as future work.

In the following figures, the same experiment is illustrated for an image containing a single ship object. In this case, increasing the vocabulary can improve the performance. Figure 4.27b gave almost perfect results considering the classification of each image patch. In the left sub-image it is shown that 27 patches were classified as ship while 5 were classified as non-ship, resulting in a confidence score of 0.84 and IoU of 0.90. The results from each image is shown in Table 4.4.



(a) $k = 500$

(b) $k = 1000$



(c) $k = 5000$

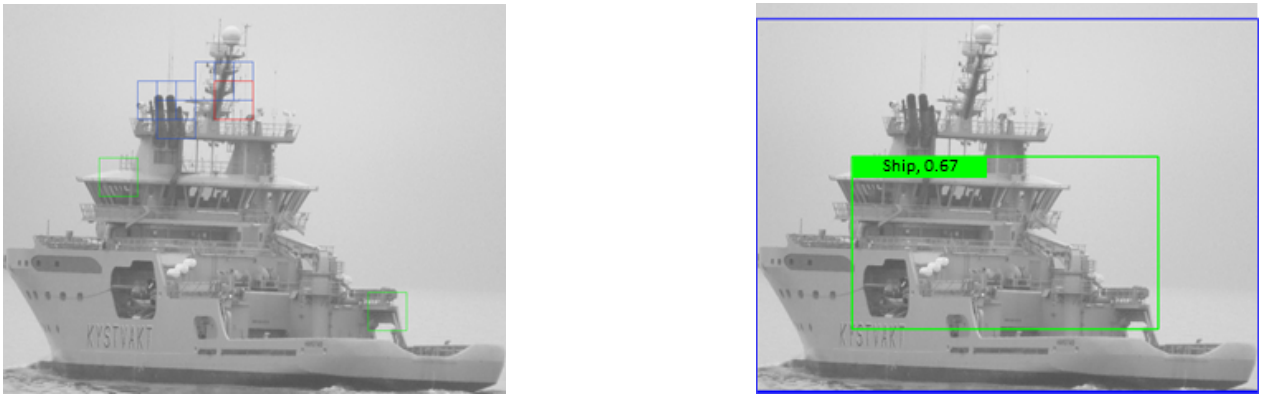
Figure 4.27: Comparison of ship detection when increasing vocabulary size

Table 4.4: Overview over classification for ship

Classification summary			
Vocabulary size	500	1000	5000
Number positive patches	30	27	17
Number negative patches	1	5	8
Confidence score	0.97	0.84	0.68
IoU	0.44	0.9	0.85

A disadvantage with how the predicted bounding boxes are generated is that the IoU can be inaccurate if not all the correctly classified patches are placed in the same region. Figure 4.27a illustrate how one misclassified image patch can decrease the localization accuracy of the detected object. It should have been implemented a method that detects false positive image patches and ignores them before localizing the object. Another disadvantage that was observed during testing with the sliding window approach is that in some cases, not enough patches are extracted from the image. Figure 4.28 show the affect of this. The same variance threshold was

used during testing. When using this threshold on the Figure shown below, it is impossible to both classify and localize the object since few patches has been extracted.



(a) Classification of patches (b) Predicted bounding box

Figure 4.28: Sample image where less patches are extracted

Based on the classification results, changing the size of vocabulary does not improve the performance. It works to an extend, but larger values of k does affect the performance in a negative way. This observation was also presented in Section 3.5. The best results was achieved with a ship vocabulary of size 1000 and non-ship of size 500. The difference in sizes makes sense since there is more features in ship compared to non-ship.

It was also observed that the model classified non-ship features poorly. The model was more unsure when classifying features from non-ship objects than ship features. This indicate that the model is not able to perfectly distinguish between ship and non-ship features, even when the target object is fairly easy to classify and locate. The reason that it was more challenging for the model to classify non-ship features could be a result of the difference between the number of features in the training set. Non-ship had less features compared to ship category. In addition, some non-ship features were similar to ship. This makes it challenging to differentiate between the two categories. The aim of the thesis is to detect vessel targets in cluttered environments. Therefore the model also needed to be tested on images consisting of noisy background and multiple objects. Given the results that have been presented in this Section, it is expected that the model will perform poor on images with multiple objects.

4.4.3 Multi-class Classification Experiment

For multi-object detection, it was mainly focused on evaluating the model performance based on visualization. Image showing the classified image patches and image showing only the cor-

rectly classified patches were used to analyze the performance. A precise multi-object detection model would return an output image consisting of only ship features.

Figure 4.29 show the object detection performance using a vocabulary of size 100 and a classification threshold of 0.1. It was tested with different threshold values, however a larger threshold would only result in less patches being classified. Therefore, loosing information about the target object. The top images show classified patches on the input image, while the bottom images show the correctly classified image patches. This output image can also be used to localize the ship target. On these images the ground truth boxes were added to show the correct location of each ship target. From these images it can be observed that the ship objects are localized and classified as ship, however a lot of background elements are classified too. This was expected based on the results shown in previous section. Based on the bottom images, it is challenging to locate the ship correctly in the image.

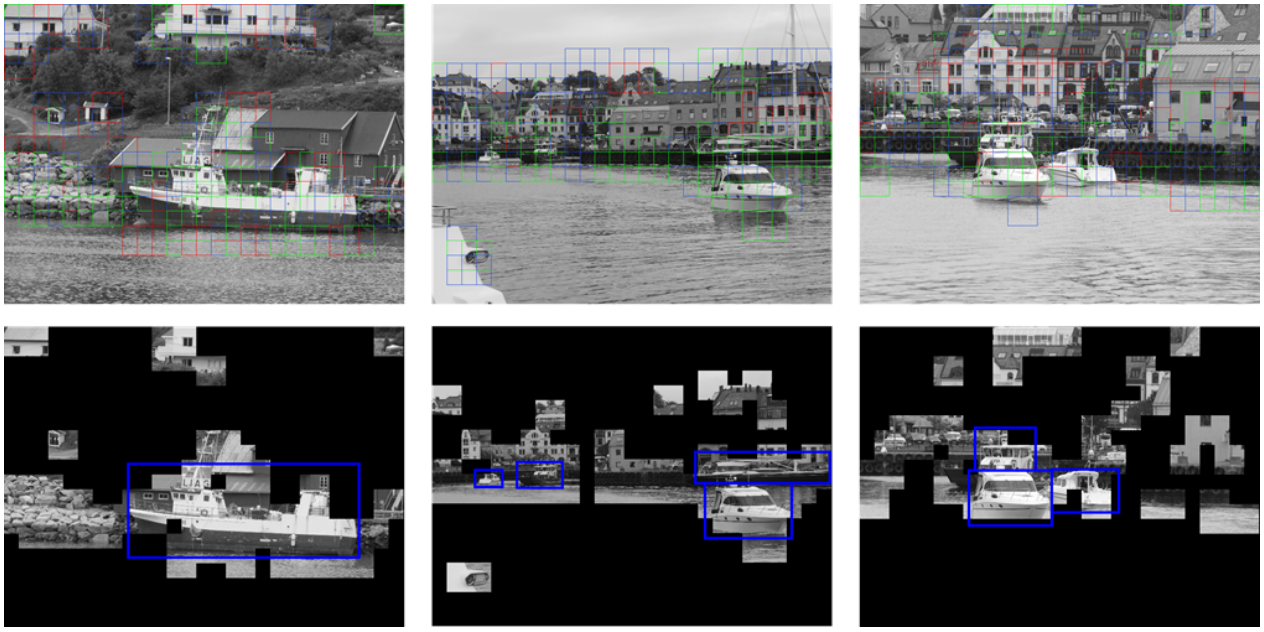


Figure 4.29: Sample images used to show multi-object detection. Ship and non-ship vocabulary of size 100, Ground truth boxes of ship is shown on output image to visualize the target location

It was then experimented with the vocabularies which gave the best results on the single-object test images. This is shown in Figure 4.30. A higher vocabulary does not show magnificent improvements. The results are similar, however more patches are wrongly classified when using a larger vocabulary.

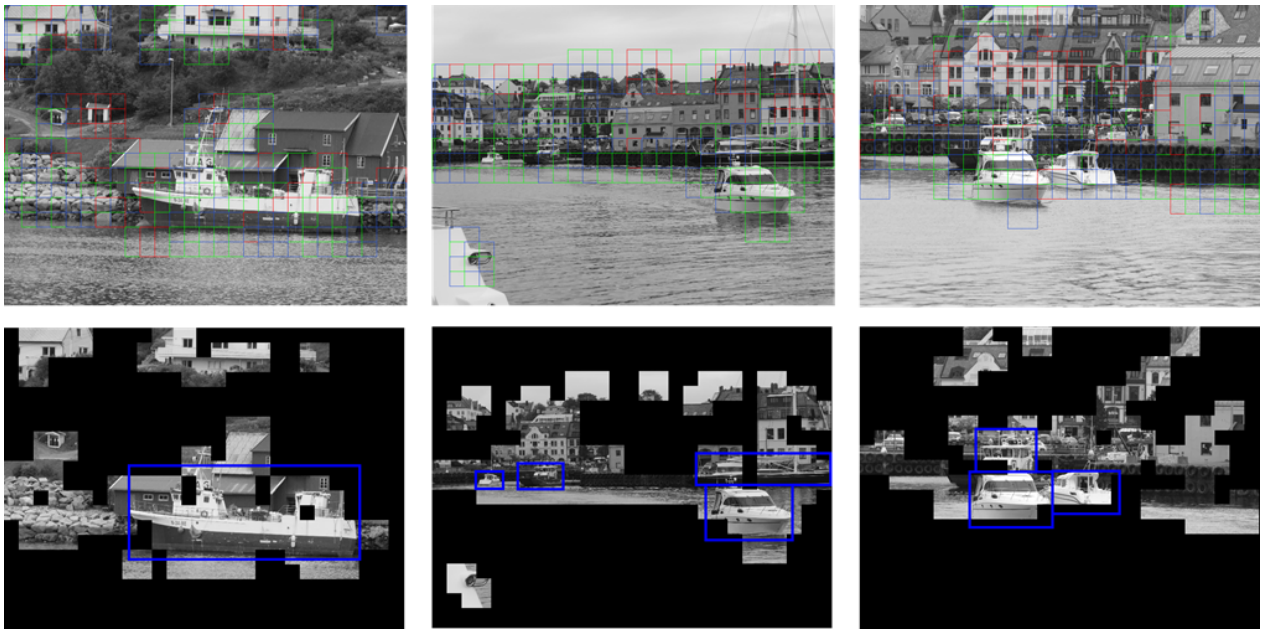


Figure 4.30: Sample images used to show multi-object detection. Ship vocabulary of size 1000 and non-ship of size 500

During experimentation of different vocabulary sizes and classification thresholds, it was not achieved a desirable detection result on the multi-object images. The model did not manage to differentiate between the two categories, thus making it challenging to locate the ship objects. The detection model can not be used to detect ships that are partly covered by other ships or objects. The method have good results when there the image is limited to only one ship in the image, and there is less background noise. When ships are clustered together, like in Figure 4.30, the detection algorithm have trouble separating them. This is not surprising as it is hard to make out the boats individually, also by human assessment. This is not desirable results since the aim of the thesis was to locate ship objects in such environments. The reason behind these results come from that the model is not able to differentiate between ship and non-ship as stated in Section 4.4.2. There is a need for a larger dataset, both from ship and non-ship. Another suggestion in addition to increasing the dataset, could be to train the model on different ship classes. Instead of only having two classes, it could be extended to *motorboat*, *sailboat* and *motorboat with priority*. The features of a sailboat will be different from a motorboat, so trying to classify all the ship features as the same object might confuse the detector. In addition, the similarities between the ship and non-ship features need to be improved such that more unique features from both classes are used for generating the vocabulary. A way of solving this could be to tune the HoG descriptor, because even after filtering out the most similar patches, there still is some similarities between the descriptors. It was also noticed during testing that the difference between the cluster centres in ship vocabulary is similar to the cluster centres in

non-ship vocabulary. This is not desired when the classification is dependent on the distance to the cluster centres in the vocabularies. If this is handled, then the difference between the categories becomes greater. Although, the classification of the image patches are not precise, the model still manage to detect the ship objects of the image. The detection is not precise, but it does not miss a ship object. This means that the feature extraction approach performs well and the variance threshold works. It can be observed in the left sub-images of Figure 4.30 that most of the features are extracted from the ship and non-ship objects only. The sliding window technique with variance threshold exclude patches with background elements i.e bushes and ocean.

As stated in Section 3.8.2, evaluation measurements were not implemented for these experiments due to the poor detection performance. Other methods would be necessary to implement in order to be able to create a complete detection algorithm for the results shown above. The approach that has been chosen in this thesis to detect the object has its weakness. As explained during the methodology chapter, the ship is detected based on the location of the correctly classified patches. This works well as long as the patches are classified precisely and if there is only one ship present. Therefore, the model was evaluated based on visual results only. More evaluation measurements are set as future work when the image patch classification become more precise. All results gathered from experiments with multi-object detection are shown in Appendix B.

Chapter 5

Conclusion and Future Work

A method for detecting maritime vessels in camera images using a Bag-of-Visual-Words (BoVW) approach has been presented. The method was based on feature extraction, vocabulary generation, and feature matching for completing the ship detection task. While the main method relied on sliding window for feature extraction, the thesis also explored this step using Scale-Invariant Feature Transform (SIFT), but with less success than the main method which was better at obtaining ship features. In this project thesis, the challenges of object detection in maritime environments have been analyzed. Evaluation of the object detection method was presented, and the results for single-object detection indicated that the method is promising. However, the method failed to handle the main task of this thesis, which were to implement an object detector on real-life, cluttered maritime environments. The main challenge was to achieve better and more precise classification of image patches. Unfortunately, the model had trouble to distinguish between ship and non-ship features.

Even though there are some challenges that need to be addressed within the implementation, the BoVW method still proved to be noteworthy and can be implemented in the future. The approach has potential to contribute to the improvement of vessel detection in maritime harbour environments. However, this is something that needs to be investigated further. The BoVW method may not reach up to its competitors within deep learning, but can be used together with one of the deep architectures to tackle the problems within the maritime environments.

5.1 Future work

Based on the reported results and discussion in Chapter 4, there are several suggestions for future work.

- A larger dataset would be beneficial for generalizing the detector to both ship and non-ship objects. The data should be gathered in relevant environments, containing different boats, buildings and locations.
- Improved feature detection and descriptor methods are considered essential for further continuation of using BoVW approach for object detection. It is needed to conduct more tests to detect better features with sliding window technique. In addition, improving the HoG descriptor or finding another option to represent the extracted image patches.
- Do more experiments with the vocabularies to see if it is possible to get better cluster centres. Good cluster centres is crucial when classifying descriptors with nearest neighbor algorithm.
- Implementing and testing with SVM to distinguish between classes in the dataset.
- Implementing object localization algorithm for multi-object images. This can be done by computing the distribution of correctly classified patches in certain areas and using this to create predicted bounding boxes. In addition, implementing evaluation measurements like precision and recall to evaluate the overall performance of the method.
- There has not been implemented an orientation algorithm in this project. The labeling of the ship parts are completed, but an algorithm for detection of these ship parts are missing.
- Implement and test object detection algorithm on multiple ship classes (i.e *motorboat*, *motorboat with priority*, *sailboat*)

Bibliography

[1] Chris Albon. Svc parameters when using rbf kernel, December 2018. URL https://chrisalbon.com/code/machine_learning/support_vector_machines/svc_parameters_using_rbf_kernel/. [Accessed: 12.06.2022].

[2] Nursabillilah Mohd Ali, Soon Wei Jun, Mohd Safirin Karis, Mariam Md Ghazaly, and Mohd Shahrieel Mohd Aras. Object classification and recognition using bag-of-words (bow) model. March 2016.

[3] David Arthur and Sergei Vassilvitskii. K-means++: The advantages of careful seeding. volume 8, pages 1027–1035, January 2007.

[4] *Marvel: A Large-Scale Image Dataset for Maritime Vessels*, 2016. Asian Conference on Computer Vision (ACCV).

[5] Alexey Bochkovskiy, Chien-Yao Wang, and Hong-Yuan Mark Liao. Yolov4: Optimal speed and accuracy of object detection. *CoRR*, 2020. URL https://arxiv.org/abs/2004.10934.

[6] Tolga Can, A. Onur Karali, and Tayfun Aytaç. Detection and tracking of sea-surface targets in infrared and visual band videos using the bag-of-features technique with scale-invariant feature transform. *Applied optics*, 2011.

[7] Dawood Al Chanti and Alice Caplier. Improving bag-of-visual-words towards effective facial expressive image classification. *CoRR*, abs/1810.00360, 2018. URL http://arxiv.org/abs/1810.00360.

[8] Zhijun Chen, Yishi Zhang, Chaozhong Wu, and Bin Ran. Understanding individualization driving states via latent dirichlet allocation model. *IEEE Intelligent Transportation Systems Magazine*, 11(2):41–53, 2019.

[9] Gabriella Csurka, Christopher R. Dance, Lixin Fan, Jutta Willamowsk, and Cédric Bray. Visual categorization with bags of keypoints. *Work Stat Learn Comput Vision, ECCV*, Vol. 1, January 2004.

[10] Carlos Miguel Correia da Costa. Multiview object recognition using bag of words approach, February 2014.

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

[12] Marcin Gabryel. The bag-of-words method with different types of image features and dictionary analysis. *Journal of Universal Computer Science*, 24:357–371, January 2018.

[13] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *2014 IEEE Conference on Computer Vision and Pattern Recognition*, pages 580–587, 2014. doi: 10.1109/CVPR.2014.81.

[14] Rafael C Gonzales and Richard E Woods. *Digital Image Processing*. Pearson, 4 edition, 2018.

[15] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. The MIT Press, 2017.

[16] Chirag Goyal. Multiclass classification using svm, May 2021. URL https://www.analyticsvidhya.com/blog/2021/05/multiclass-classification-using-svm/. [Accessed: 12.06.2022].

[17] Simen Viken Grini. *Object Detection in Maritime Environments*. NTNU, 2019.

[18] Jian Hou, Jianxin Kang, and Naiming Qi. On vocabulary size in bag-of-visual-words representation. pages 414–424, September 2010.

[19] Mingyuan Jiu, Christian Wolf, Christophe Garcia, and Atilla Baskurt. Supervised learning and codebook optimization for bag-of-words models. *Cognitive Computation*, 4, December 2012.

[20] Amalie Kolsgaard. *Improved maritime vessel detection in camera images using Bag-of-Visual-Words*. Specialization project. NTNU, 2021.

[21] Marthe Lauvnes. *Feature-based pose estimation of ships in monocular camera images*. NTNU, 2020.

[22] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott E. Reed, Cheng-Yang Fu, and Alexander C. Berg. SSD: single shot multibox detector. *CoRR*, abs/1512.02325, 2015. URL http://arxiv.org/abs/1512.02325.

[23] David G. Lowe. Distinctive image features from scale-invariant keypoints, November 2004. URL https://www.cs.ubc.ca/~lowe/papers/ijcv04.pdf. [Accessed: 12.05.2022].

[24] Satya Mallick. Histogram of oriented gradients explained using opencv, December 2016. URL https://learnopencv.com/histogram-of-oriented-gradients/. [Accessed: 06.05.2022].

[25] Krystian Mikolajczyk and Cordelia Schmid. A performance evaluation of local descriptors. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(10):1615–1630, 2005.

[26] Sebastian Moosbauer, Daniel König, Jens Jäkel, and Michael Teutsch. A benchmark for deep learning based object detection in maritime environments. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 916–925, 2019.

[27] Marius Muja and David Lowe. Flann - fast library for approximate nearest neighbors, November 2009. URL https://www.fit.vutbr.cz/~ibarina/pub/VGE/reading/flann_manual-1.6.pdf. [Accessed: 26.05.2022].

[28] Marius Muja and David G. Lowe. Scalable nearest neighbor algorithms for high dimensional data. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 36(11):2227–2240, 2014. doi: 10.1109/TPAMI.2014.2321376.

[29] Abraham Noel, Shreyanka K, and Kaja Gowtham Satya Kumar. Autonomous ship navigation methods: A review. November 2019. doi: 10.24868/icmet.oman.2019.028.

[30] Edgar Osuna, Robert Freund, and Federico Girosi. Training support vector machines: an application to face detection, June 1997. URL https://mitsloan.mit.edu/shared/ods/documents?DocumentID=2508. [Accessed: 12.06.2022].

[31] Rafael Padilla, Sergio L. Netto, and Eduardo A. B. da Silva. A survey on performance metrics for object-detection algorithms. In *2020 International Conference on Systems, Signals and Image Processing (IWSSIP)*, pages 237–242, 2020.

[32] Pornntiwa Pawara, Emmanuel Okafor, Olarik Surinta, and Lambert Schomaker. Comparing local descriptors and bags of visual words to deep convolutional neural networks for plant recognition. February 2017.

[33] Xiaojiang Peng, Limin Wang, Xingxing Wang, and Yu Qiao. Bag of visual words and fusion methods for action recognition: Comprehensive study and good practice. *Computer Vision and Image Understanding*, 150:109–125, 2016. ISSN 1077-3142. URL https://www.sciencedirect.com/science/article/pii/S1077314216300091.

[34] Dawid Polap and Marta Wlodarczyk-Sielicka. Classification of non-conventional ships using a neural bag-of-words mechanism. *Sensors*, 20:1608, March 2020.

[35] Dilip K. Prasad, Deepu Rajan, Lily Rachmawati, Eshan Rajabally, and Chai Quek. Video processing from electro-optical sensors for object detection and tracking in a maritime environment: A survey. *IEEE Transactions on Intelligent Transportation Systems*, 18(8):1993–2016, 2017.

[36] Wisam Qader, Musa M. Ameen, and Bilal I. Ahmed. An overview of bag of words;importance, implementation, applications, and challenges. In *2019 International Engineering Conference (IEC)*, pages 200–204, 2019.

[37] Joseph Redmon and Ali Farhadi. YOLO9000: better, faster, stronger. *CoRR*, abs/1612.08242, 2016. URL http://arxiv.org/abs/1612.08242.

[38] Joseph Redmon and Ali Farhadi. Yolov3: An incremental improvement. *CoRR*, abs/1804.02767, 2018. URL http://arxiv.org/abs/1804.02767.

[39] Joseph Redmon, Santosh Kumar Divvala, Ross B. Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. *CoRR*, abs/1506.02640, 2015. URL http://arxiv.org/abs/1506.02640.

[40] Shaoqing Ren, Kaiming He, Ross B. Girshick, and Jian Sun. Faster R-CNN: towards real-time object detection with region proposal networks. *CoRR*, 2015. URL http://arxiv.org/abs/1506.01497.

[41] Epsen Johansen Tangstad. *Visual Detection of Maritime Vessels*. NTNU, 2017.

[42] Chih-Fong Tsai. Bag-of-words representation in image annotation: A review. *ISRN Artificial Intelligence*, 2012, November 2012.

[43] Mrinal Tyagi. Hog (histogram of oriented gradients): An overview, July 2021. URL https://towardsdatascience.com/hog-histogram-of-oriented-gradients-67ecd887675f. [Accessed: 06.05.2022].

[44] Hariri Walid, Hedi Tabia, Nadir Farah, David Declercq, and Bernouareth Abdallah. Geometrical and visual feature quantization for 3d face recognition. 04 2017. doi: 10.5220/0006101701870193.

[45] Liqian Wang, Shuzhen Fan, Yunxia Liu, and Li Yongfu. A review of methods for ship detection with electro-optical images in marine environments. *Journal of Marine Science and Engineering*, 9:1408, December 2021.

[46] Jian Wu, Zhiming Cui, Victor S. Sheng, Pengpeng Zhao, Dongliang Su, and Shengrong Gong. A comparative study of sift and its variants. *Measurement Science Review*, 13, June 2013.

[47] Jun Yang, Yu-GangJiang, Alexander Hauptmann, and Chong-Wah Ngo. Evaluating bag-of-visual-words representations in scene classification. pages 197–206, January 2007.

[48] Qiqi Zhu, Yanfei Zhong, Bei Zhao, Gui-Song Xia, and Liangpei Zhang. Bag-of-visual-words scene classifier with local and global features for high spatial resolution remote sensing imagery. *IEEE Geoscience and Remote Sensing Letters*, 13(6):747–751, 2016.

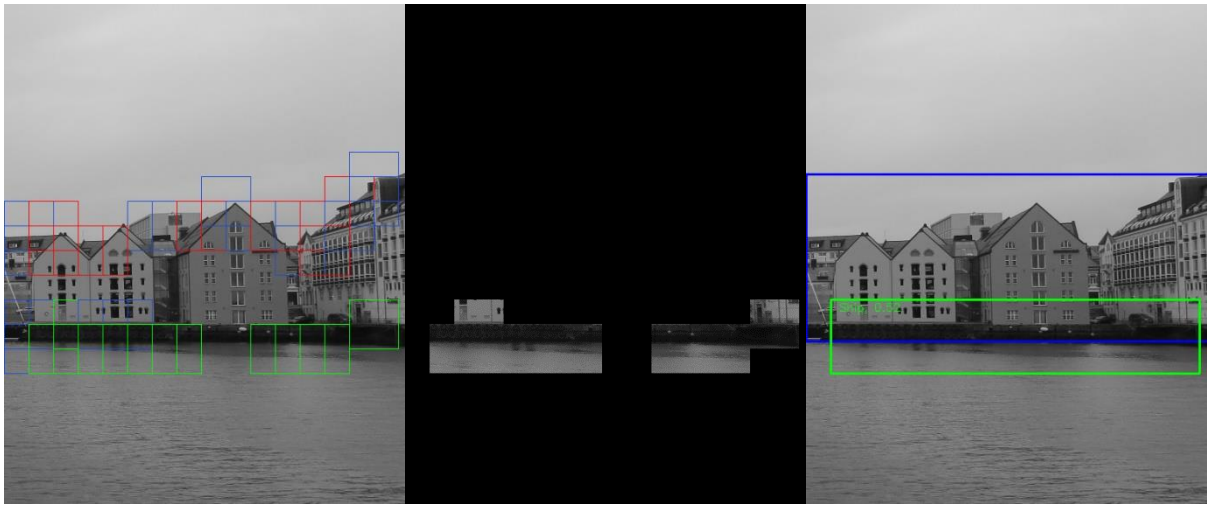
Appendices

Appendix A

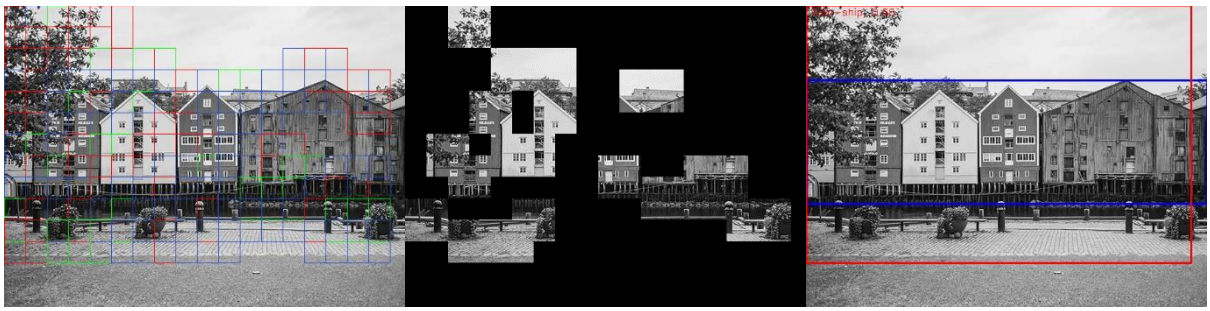
Results: Single-object Detection

Specification

variance_threshold_train	1500
hog_param	[9, (8,8), (2,2)]
diff_threshold	5
N_clusters_ship	100
N_clusters_nonship	100
init	“k-means++”
variance_threshold_test	1500
T_classification	0.1

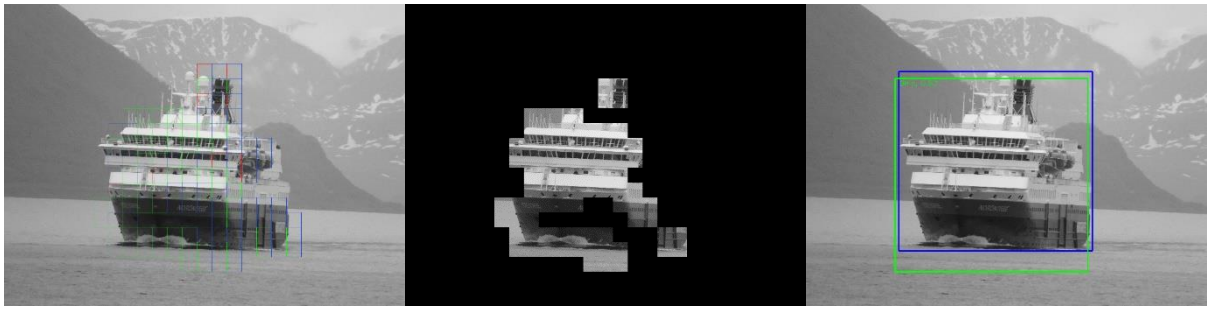


Total number of positive patches: 11
Total number of negative patches: 10
IoU ship: 0.19678801493669756

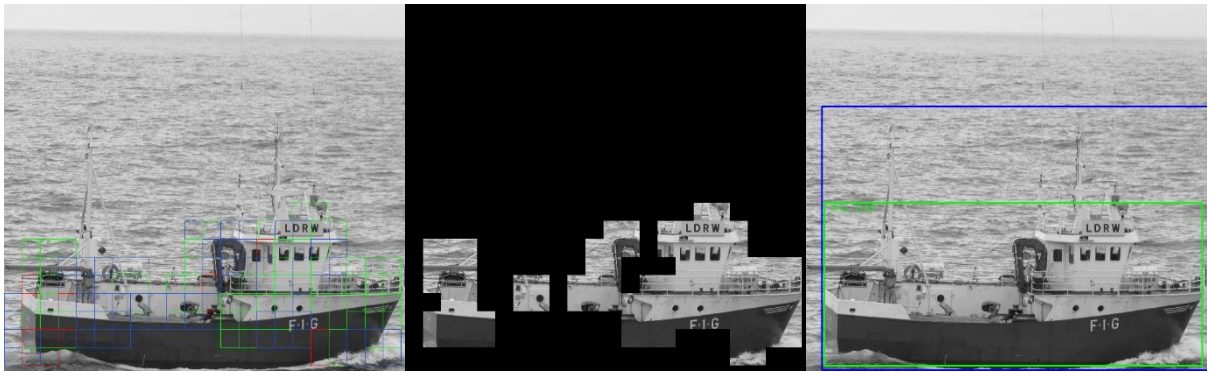


Total number of positive patches: 24
Total number of negative patches: 47
IoU ship: 0.4702870887304985

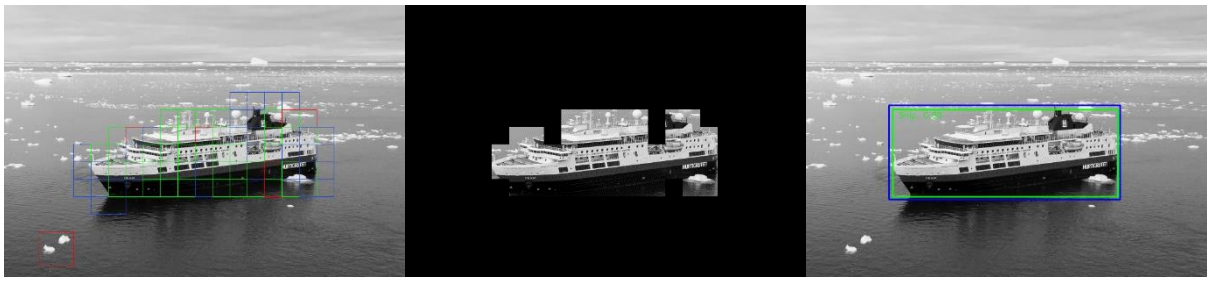
Total number of positive patches: 27
Total number of negative patches: 25
IoU ship: 0.9264117827998625



Total number of positive patches: 40
Total number of negative patches: 5
IoU ship: 0.8273679402264348

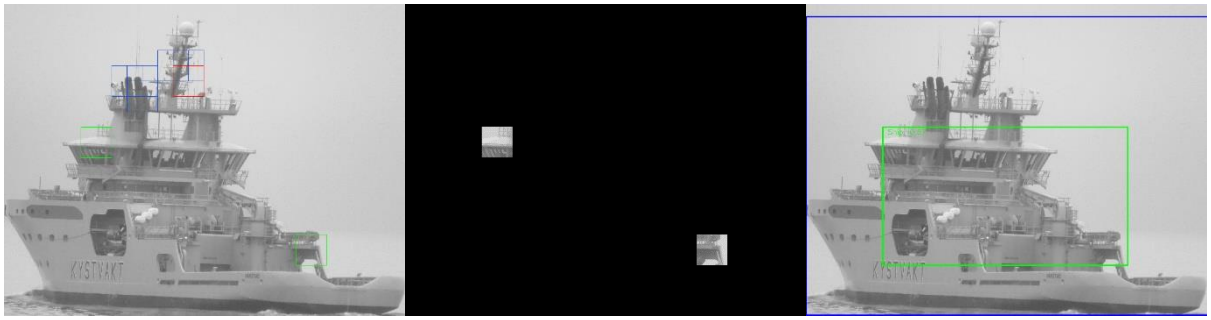


Total number of positive patches: 39
Total number of negative patches: 7
IoU ship: 0.6049372348498986

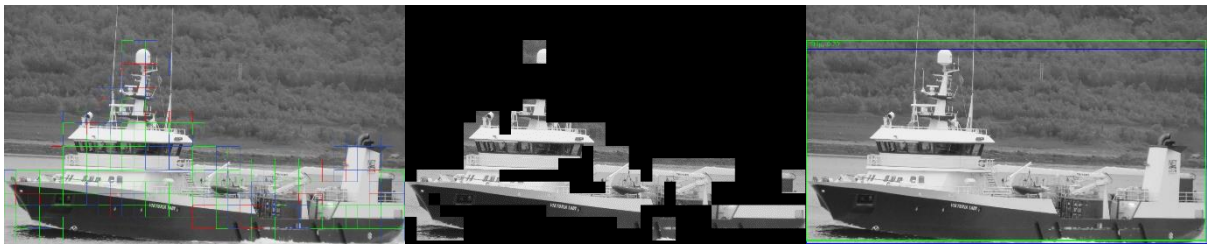


Total number of positive patches: 28
Total number of negative patches: 5
IoU ship: 0.9015066065098292

Total number of positive patches: 20
Total number of negative patches: 24
IoU ship: 0.7130730050933786



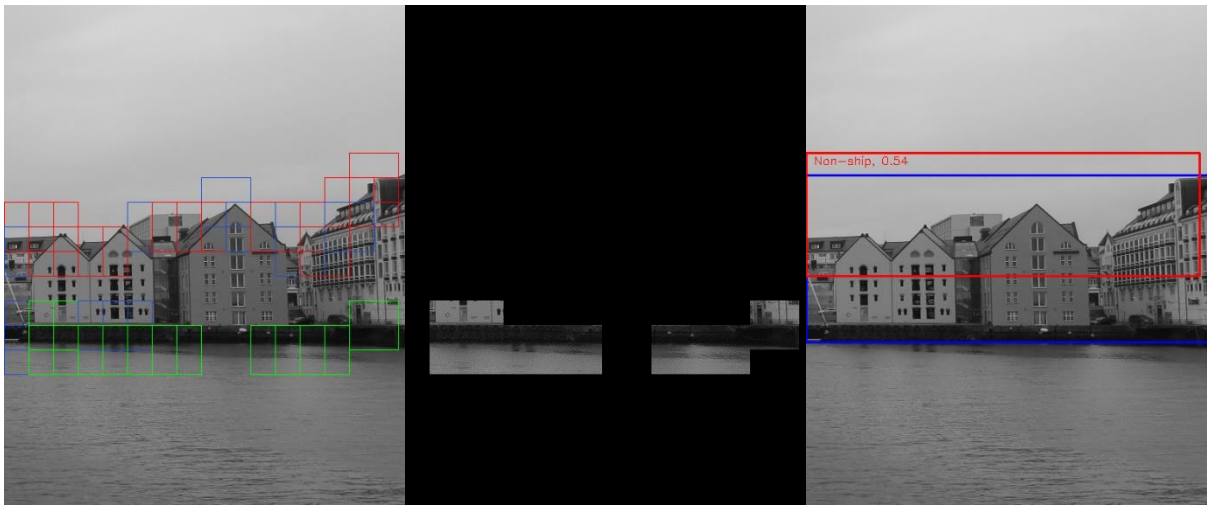
Total number of positive patches: 2
Total number of negative patches: 1
IoU ship: 0.28330084842925934



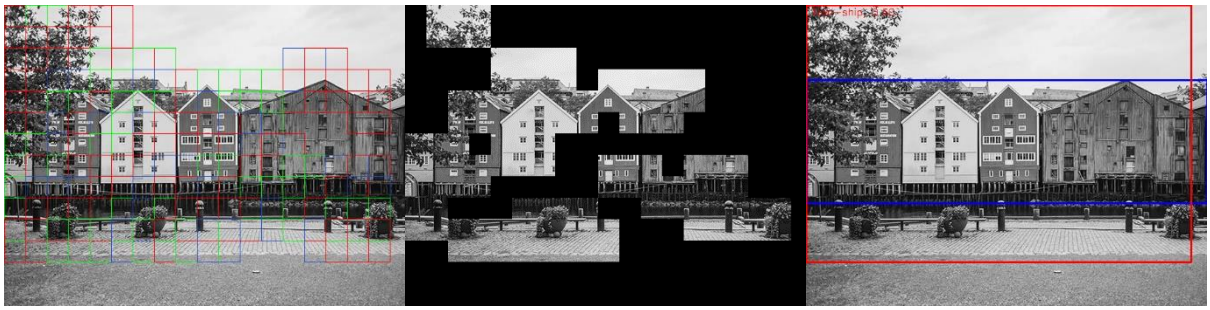
Total number of positive patches: 77
Total number of negative patches: 20
IoU ship: 0.9362934968790106

Specification

variance_threshold_train	1500
hog_param	[9, (8,8), (2,2)]
diff_threshold	5
N_clusters_ship	100
N_clusters_nonship	100
init	“k-means++”
variance_threshold_test	1500
T_classification	0.05

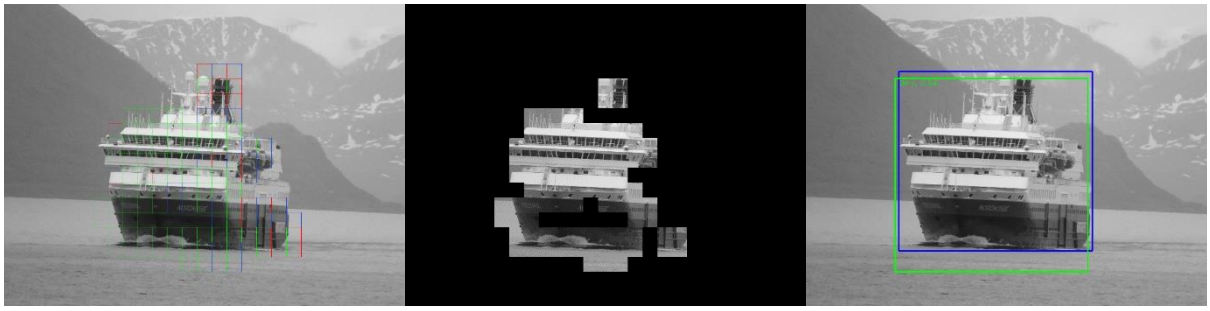


Total number of positive patches: 12
Total number of negative patches: 14
IoU ship: 0.5244870613202798

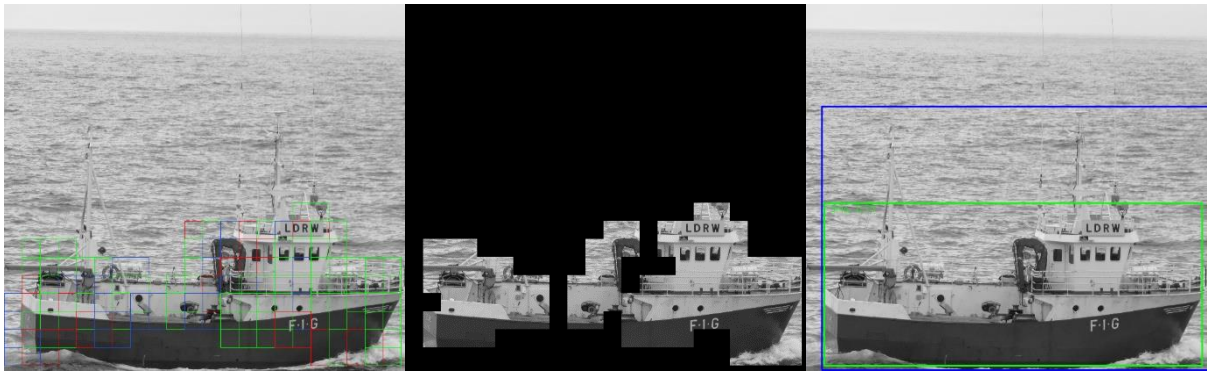


Total number of positive patches: 44
Total number of negative patches: 66
IoU ship: 0.4702870887304985

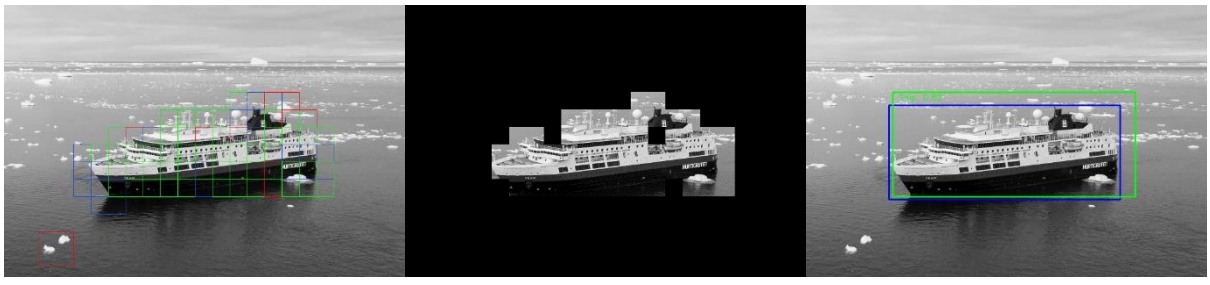
Total number of positive patches: 37
Total number of negative patches: 38
IoU ship: 0.9264117827998625



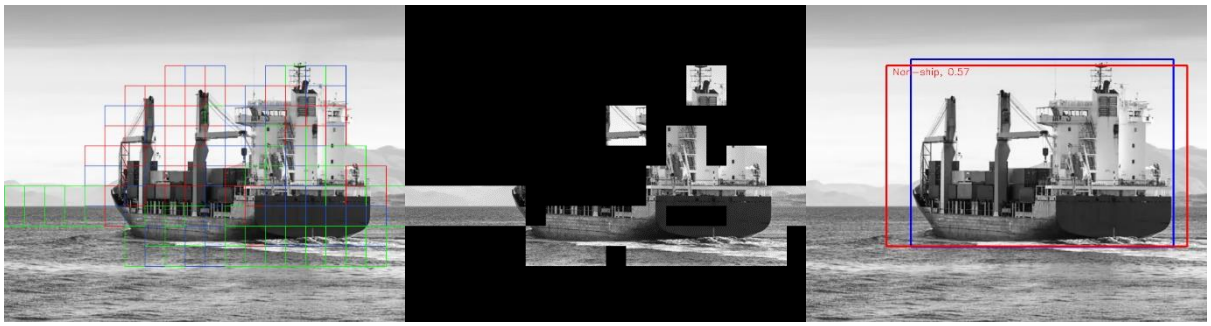
Total number of positive patches: 51
Total number of negative patches: 10
IoU ship: 0.8273679402264348



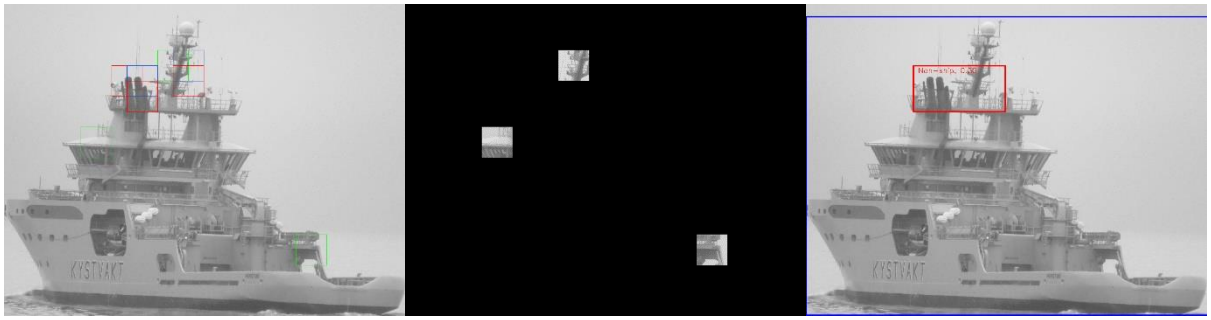
Total number of positive patches: 48
Total number of negative patches: 16
IoU ship: 0.6049372348498986



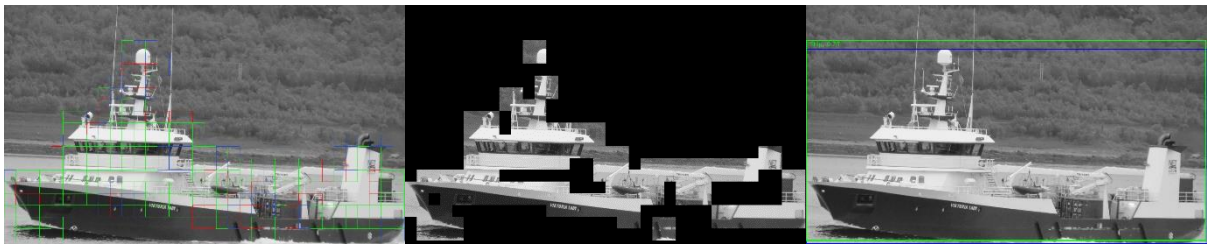
Total number of positive patches: 33
Total number of negative patches: 7
IoU ship: 0.7907201600355634



Total number of positive patches: 29
Total number of negative patches: 39
IoU ship: 0.8450939573484464



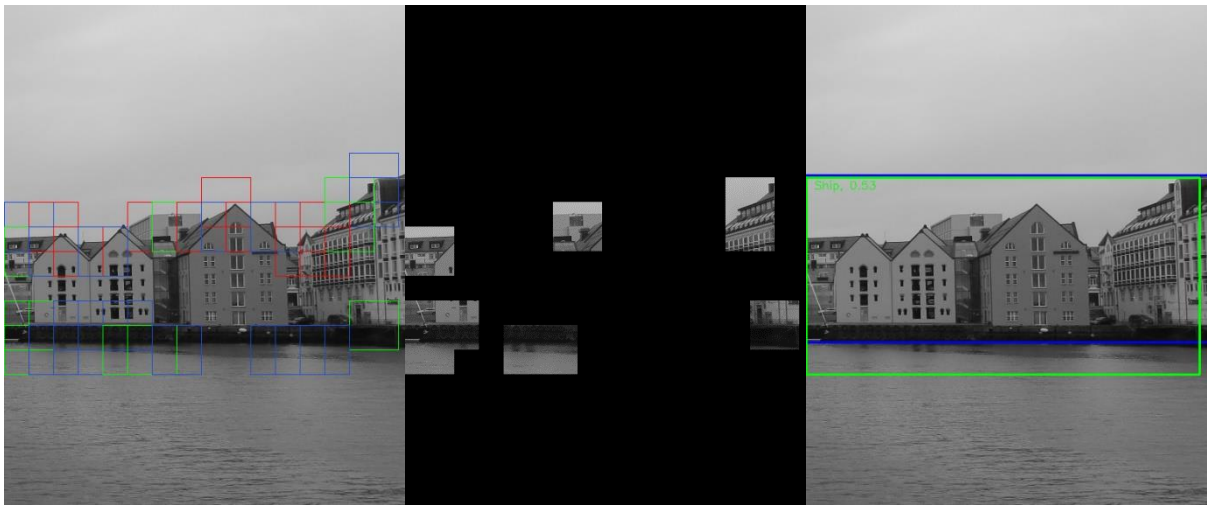
Total number of positive patches: 3
Total number of negative patches: 3
IoU ship: 0.03577352289230299



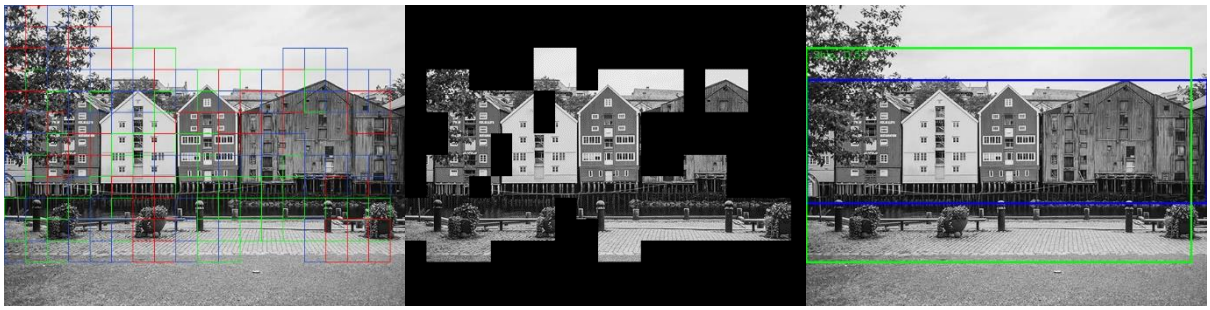
Total number of positive patches: 93
Total number of negative patches: 26
IoU ship: 0.9362934968790106

Specification

variance_threshold_train	1500
hog_param	[9, (8,8), (2,2)]
diff_threshold	5
N_clusters_ship	500
N_clusters_nonship	500
init	“k-means++”
variance_threshold_test	1500
T_classification	0.1

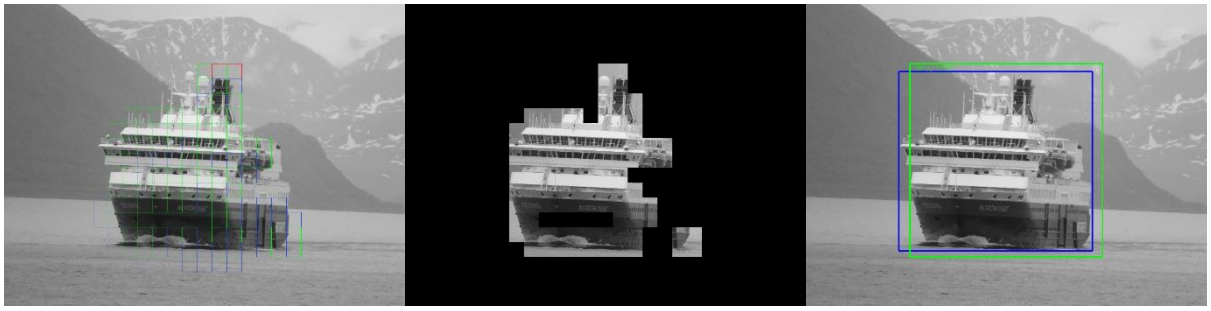


Total number of positive patches: 10
Total number of negative patches: 9
IoU ship: 0.8123186377763703

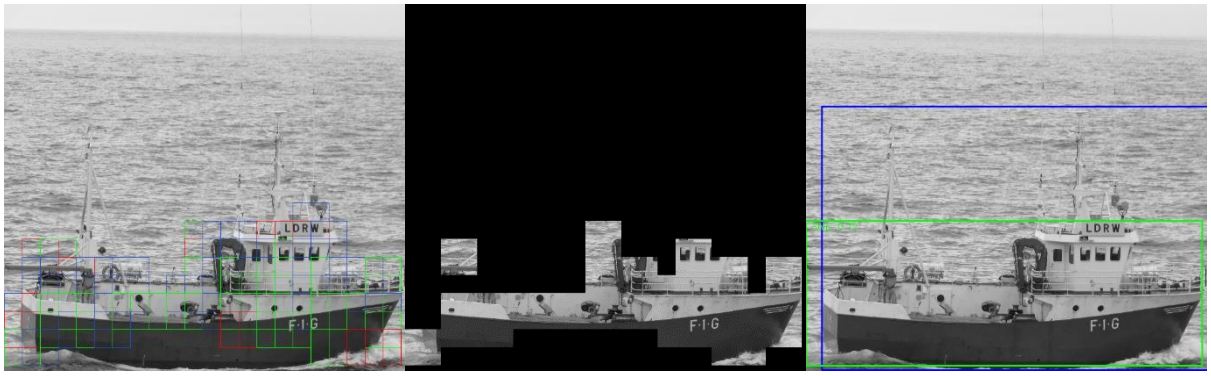


Total number of positive patches: 43
Total number of negative patches: 39
IoU ship: 0.5618564039291986

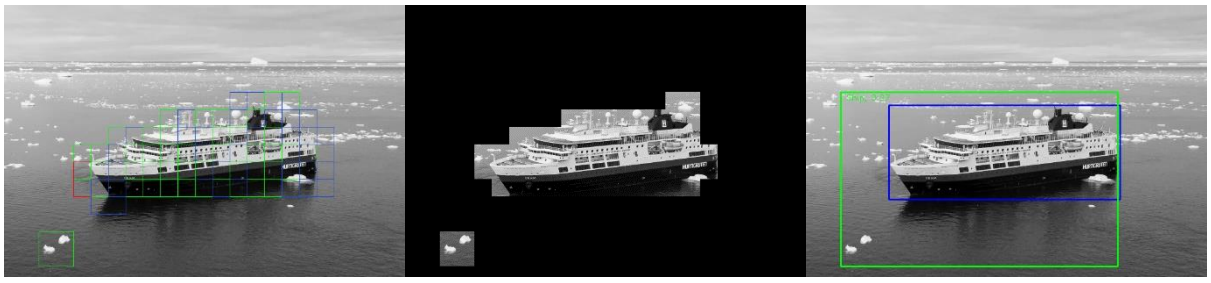
Total number of positive patches: 39
Total number of negative patches: 21
IoU ship: 0.9264117827998625



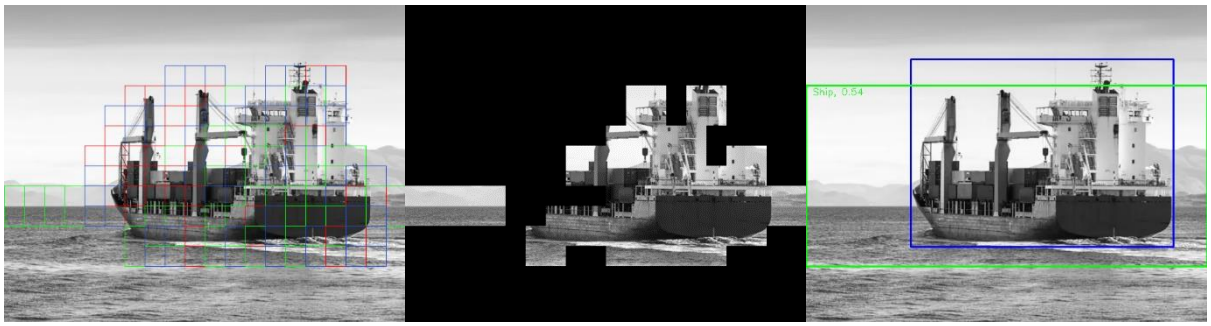
Total number of positive patches: 52
Total number of negative patches: 1
IoU ship: 0.8345207094777183



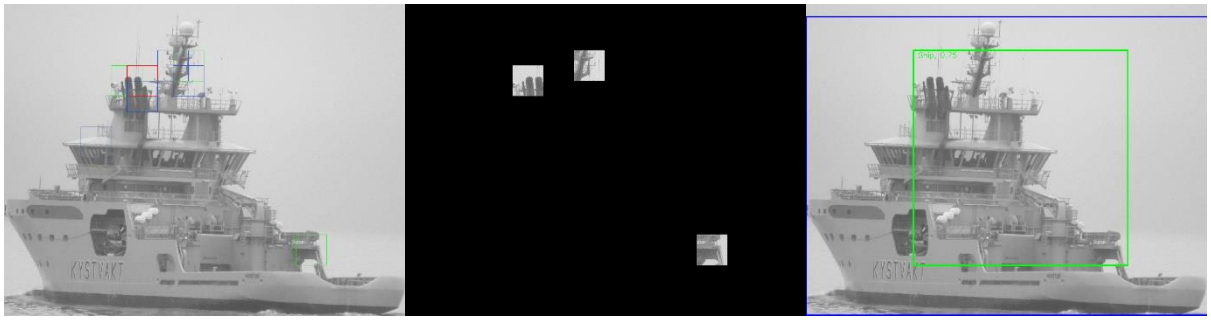
Total number of positive patches: 34
Total number of negative patches: 13
IoU ship: 0.5293052885200419



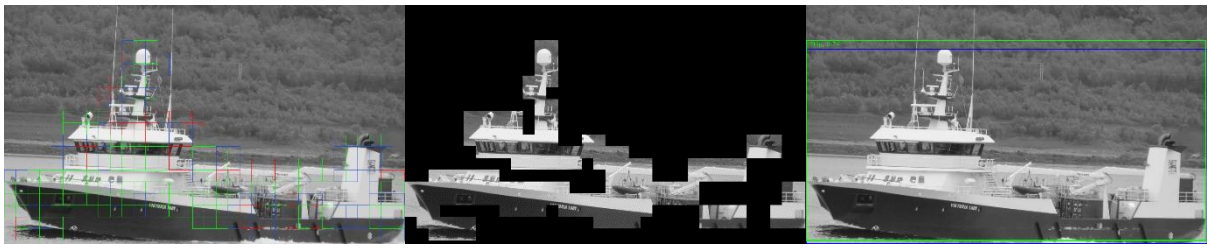
Total number of positive patches: 30
Total number of negative patches: 1
IoU ship: 0.44612956479146637



Total number of positive patches: 34
Total number of negative patches: 29
IoU ship: 0.5340851401505257



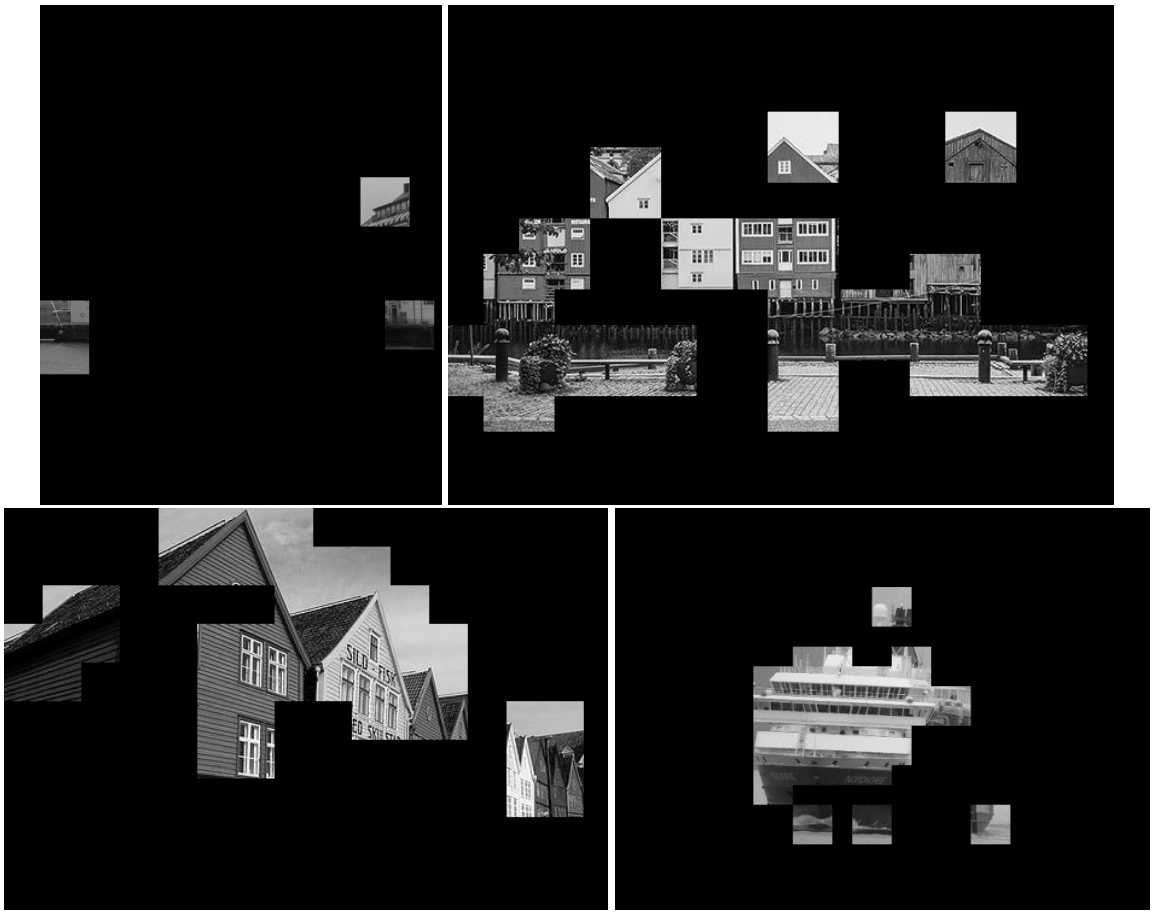
Total number of positive patches: 3
Total number of negative patches: 1
IoU ship: 0.3852346556600168

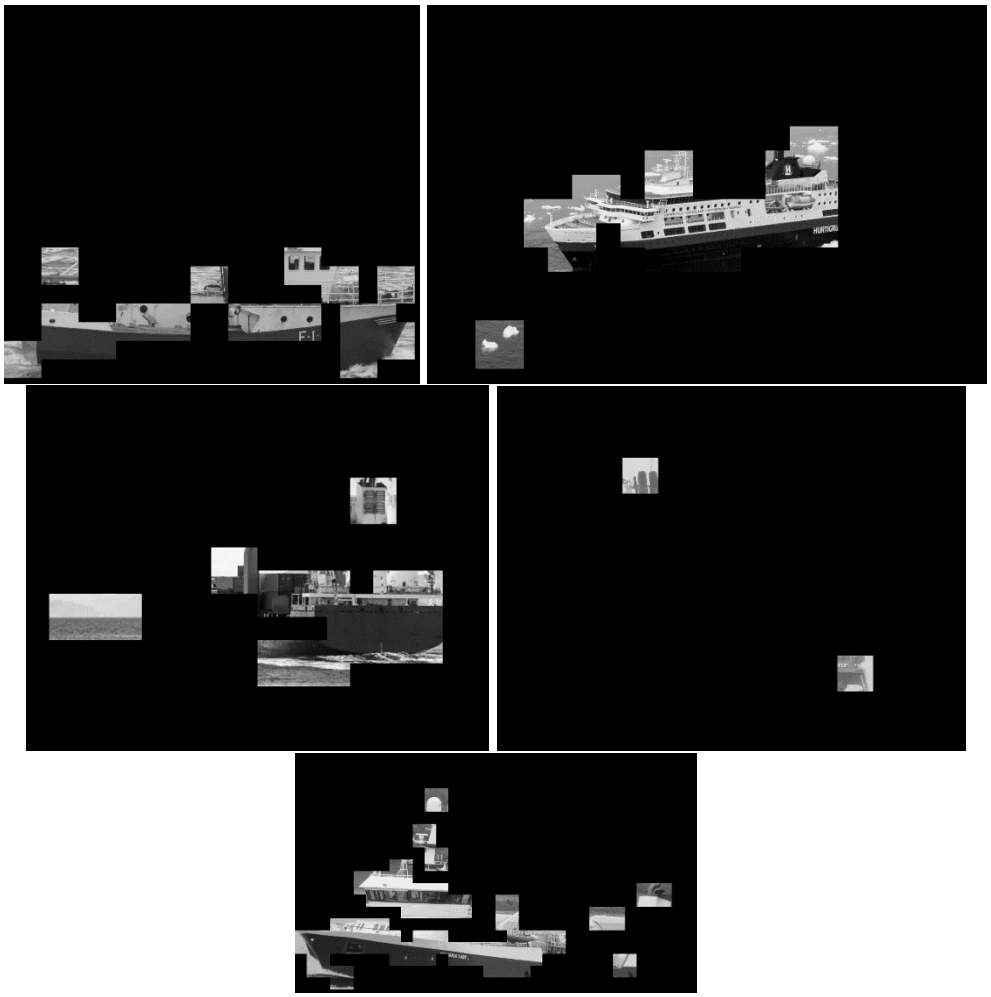


Total number of positive patches: 81
Total number of negative patches: 28
IoU ship: 0.9362934968790106

Specification

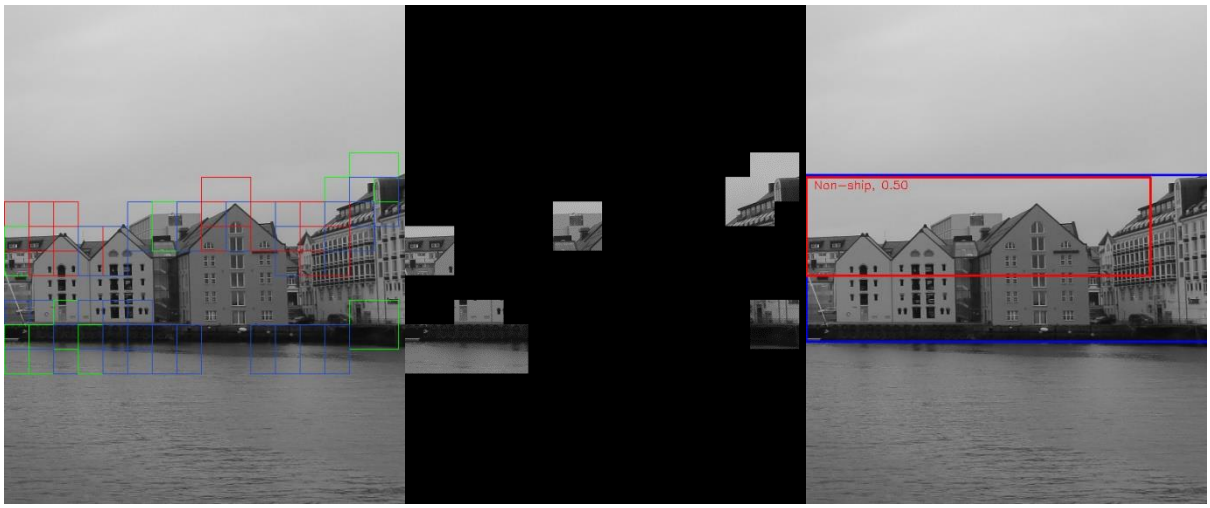
variance_threshold_train	1500
hog_param	[9, (8,8), (2,2)]
diff_threshold	5
N_clusters_ship	500
N_clusters_nonship	500
init	“k-means++”
variance_threshold_test	1500
T_classification	0.2



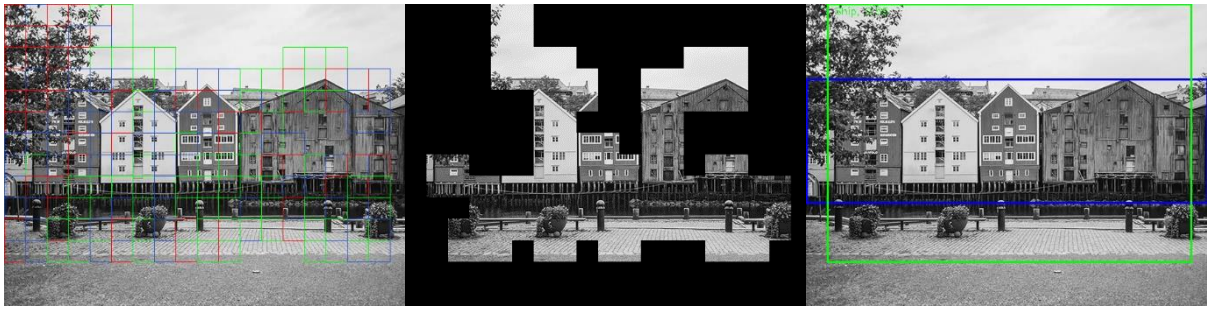


Specification

variance_threshold_train	1500
hog_param	[9, (8,8), (2,2)]
diff_threshold	5
N_clusters_ship	1000
N_clusters_nonship	1000
init	“k-means++”
variance_threshold_test	1500
T_classification	0.1

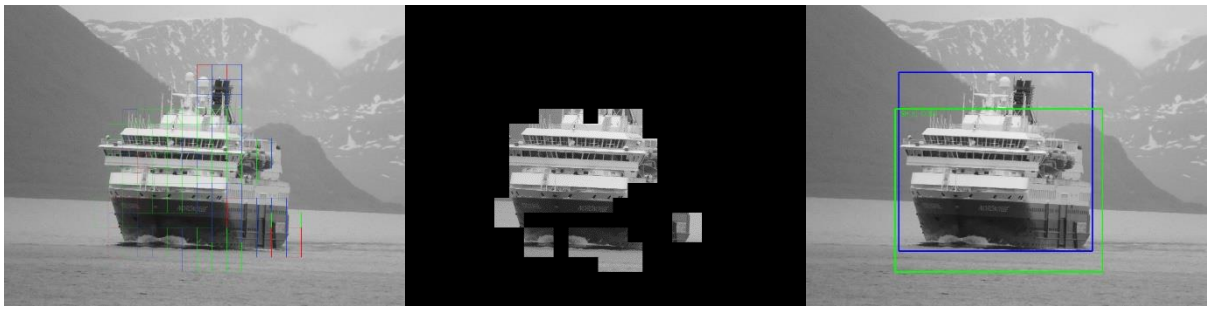


Total number of positive patches: 9
Total number of negative patches: 9
IoU ship: 0.507047062119196

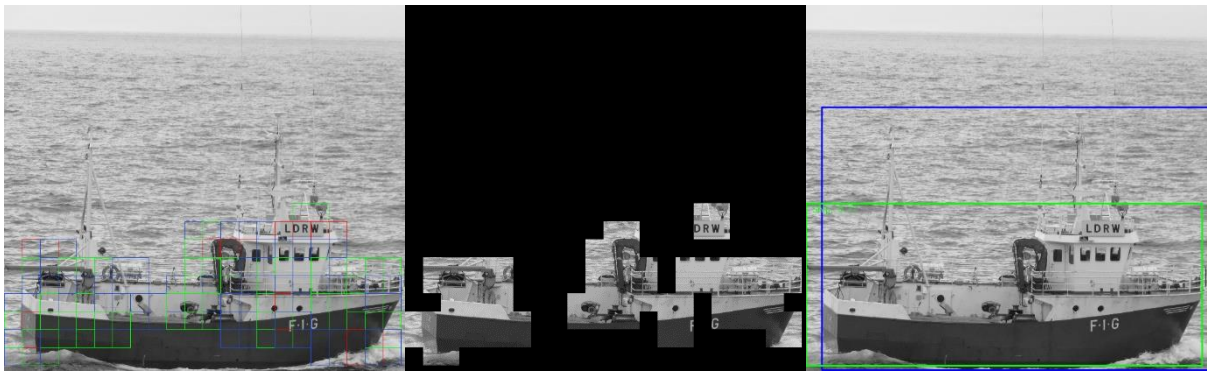


Total number of positive patches: 47
Total number of negative patches: 38
IoU ship: 0.45829545454545456

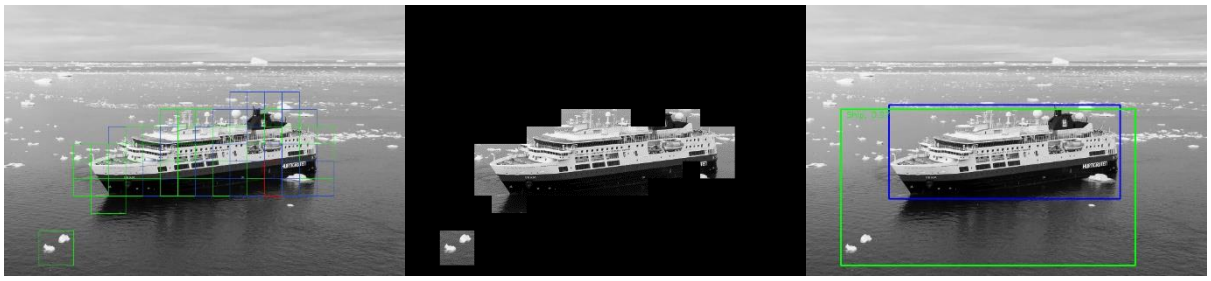
Total number of positive patches: 33
Total number of negative patches: 27
IoU ship: 0.9264117827998625



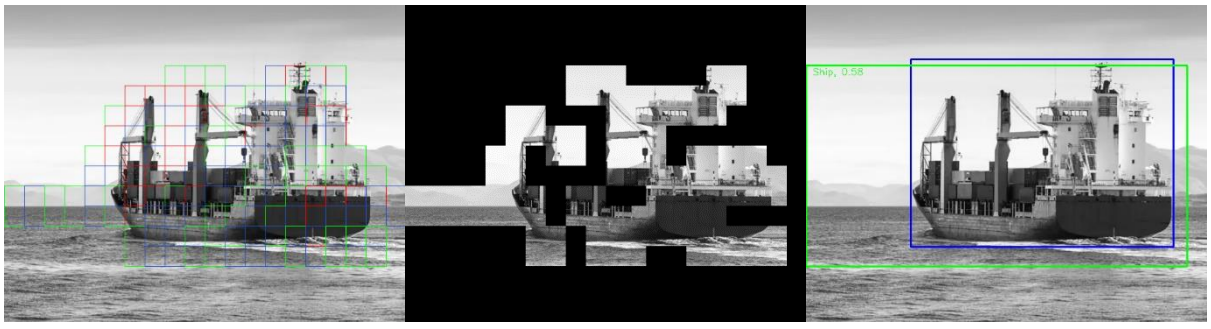
Total number of positive patches: 42
Total number of negative patches: 7
IoU ship: 0.6735560913997014



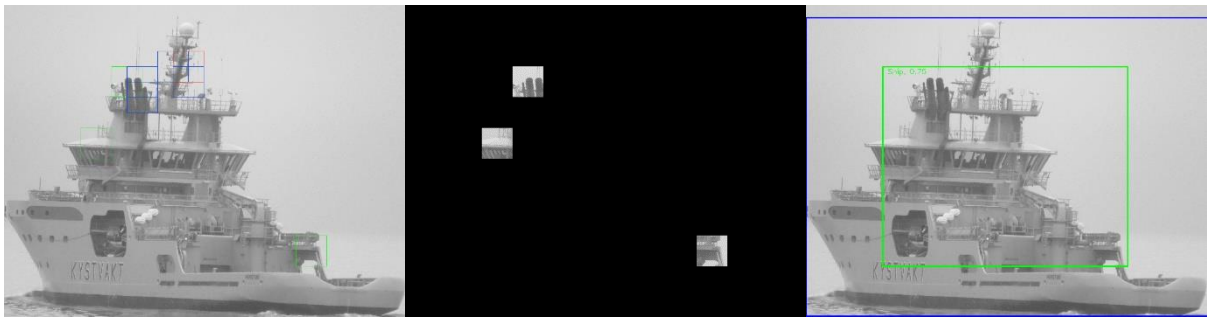
Total number of positive patches: 33
Total number of negative patches: 10
IoU ship: 0.5935929953156477



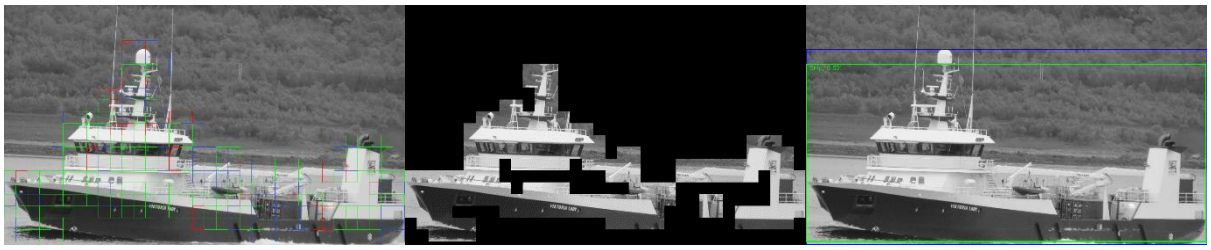
Total number of positive patches: 30
Total number of negative patches: 1
IoU ship: 0.44148661832236574



Total number of positive patches: 36
Total number of negative patches: 26
IoU ship: 0.6099484698706489



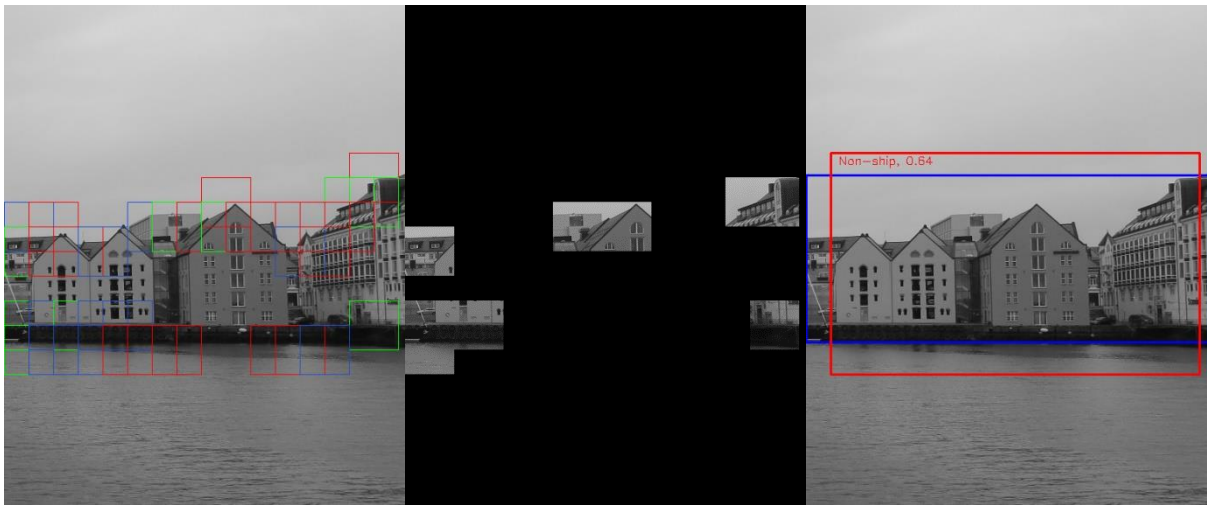
Total number of positive patches: 3
Total number of negative patches: 1
IoU ship: 0.40877665673010777



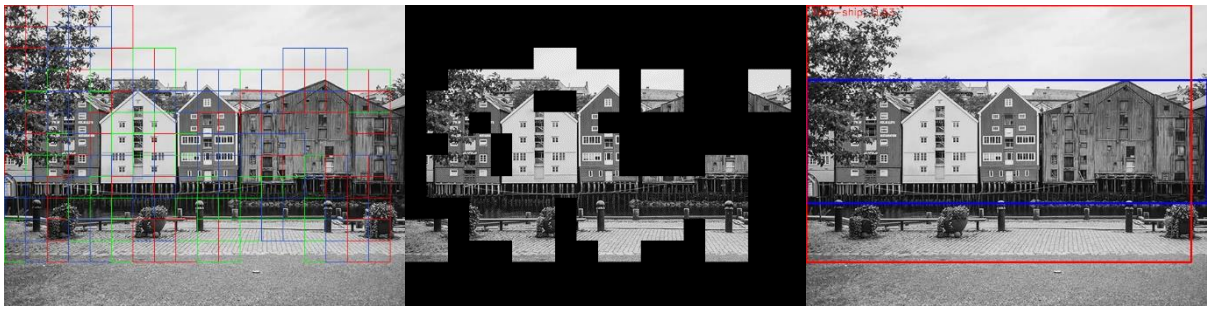
Total number of positive patches: 93
Total number of negative patches: 23
IoU ship: 0.9034719572682183

Specification

variance_threshold_train	1500
hog_param	[9, (8,8), (2,2)]
diff_threshold	5
N_clusters_ship	1000
N_clusters_nonship	500
init	“k-means++”
variance_threshold_test	1500
T_classification	0.1

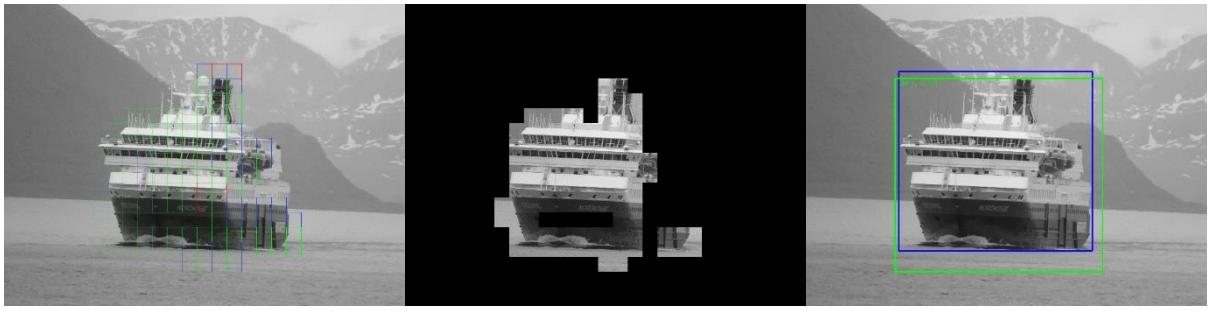


Total number of positive patches: 9
Total number of negative patches: 16
IoU ship: 0.7066712494018856

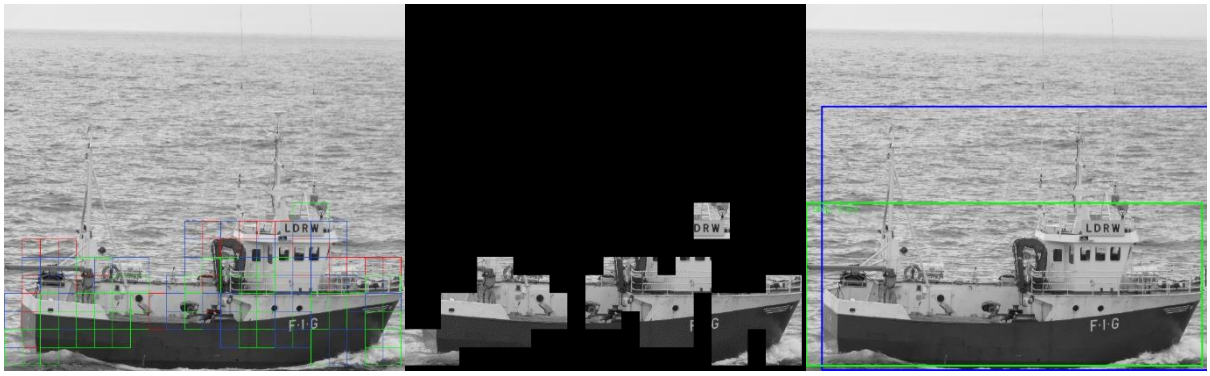


Total number of positive patches: 37
Total number of negative patches: 42
IoU ship: 0.4702870887304985

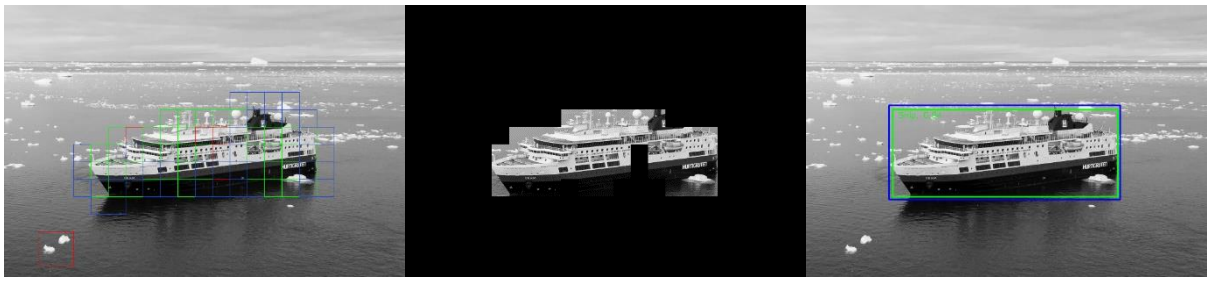
Total number of positive patches: 44
Total number of negative patches: 23
IoU ship: 0.9264117827998625



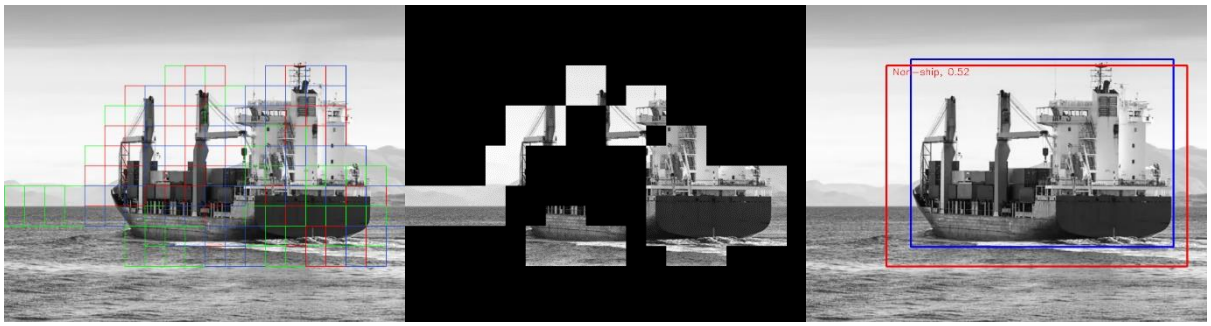
Total number of positive patches: 42
Total number of negative patches: 7
IoU ship: 0.6735560913997014



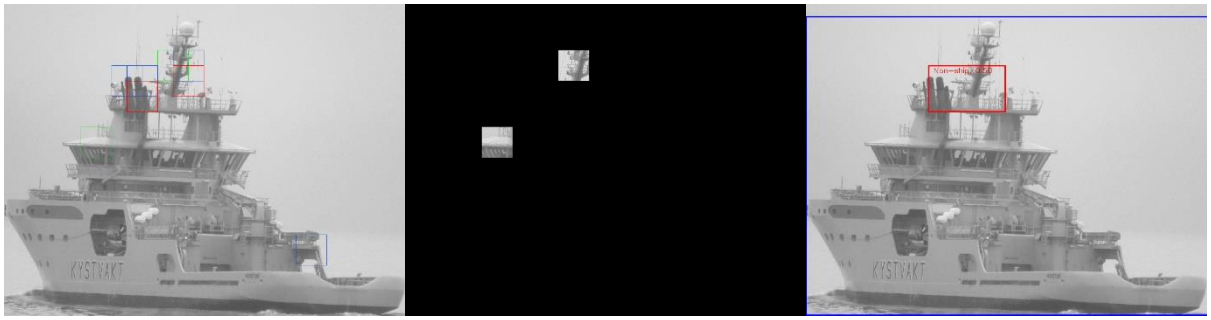
Total number of positive patches: 32
Total number of negative patches: 19
IoU ship: 0.5935929953156477



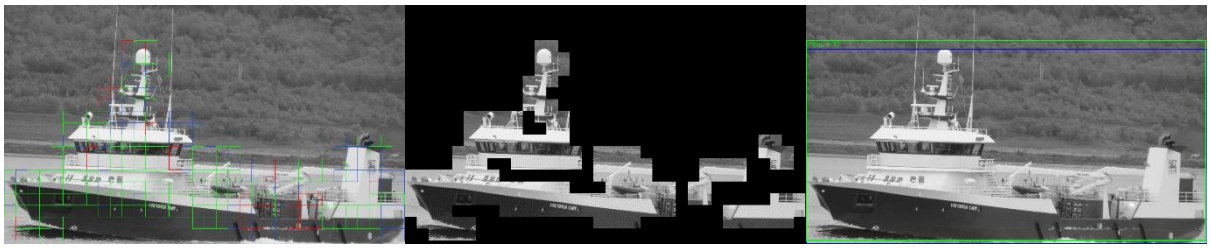
Total number of positive patches: 27
Total number of negative patches: 5
IoU ship: 0.9015066065098292



Total number of positive patches: 30
Total number of negative patches: 33
IoU ship: 0.7679648930334614



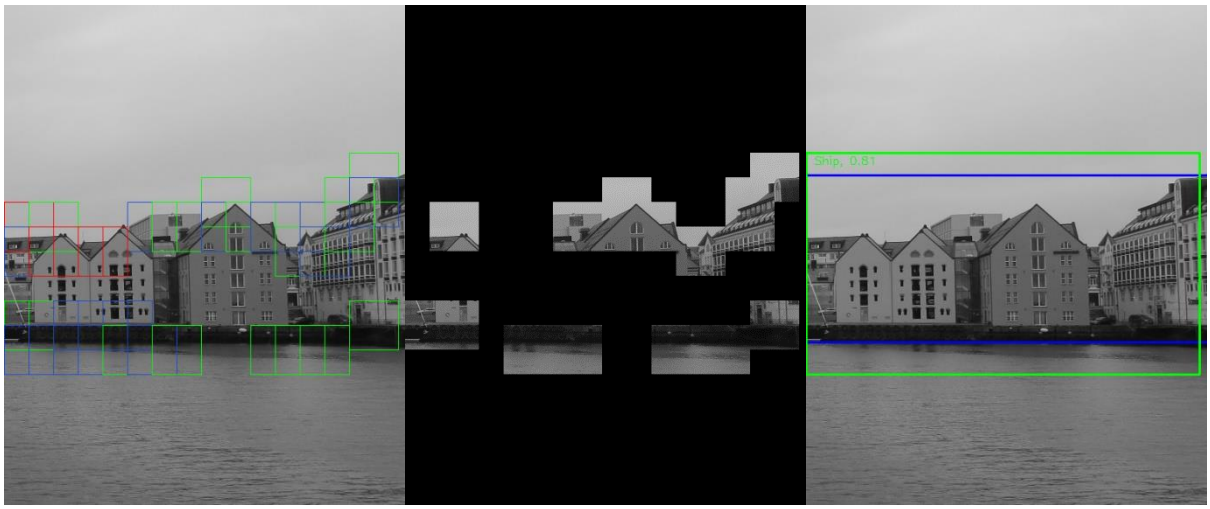
Total number of positive patches: 2
Total number of negative patches: 2
IoU ship: 0.02984216158373462



Total number of positive patches: 93
Total number of negative patches: 23
IoU ship: 0.9362934968790106

Specification

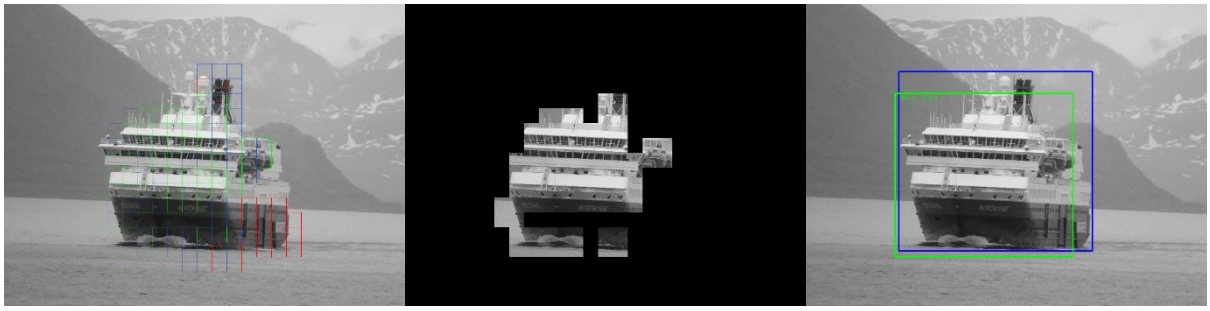
variance_threshold_train	1500
hog_param	[9, (8,8), (2,2)]
diff_threshold	5
N_clusters_ship	5000
N_clusters_nonship	5000
init	“k-means++”
variance_threshold_test	1500
T_classification	0.1



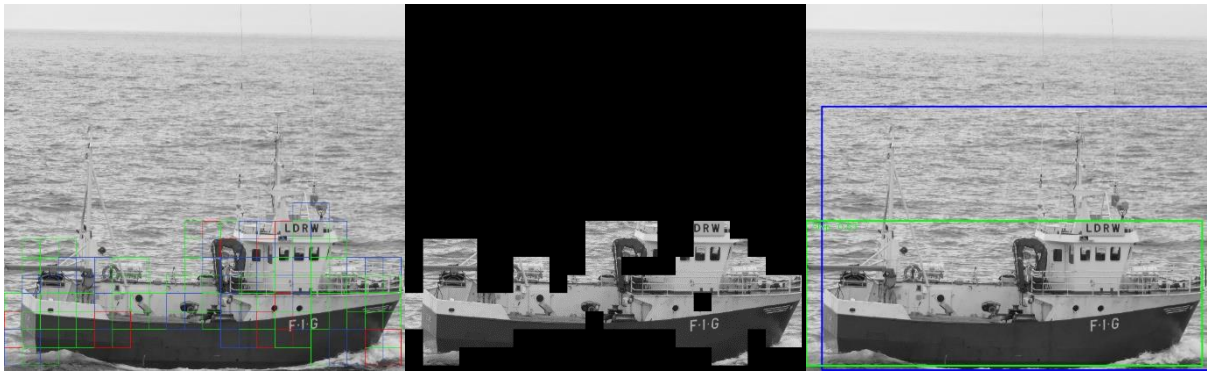
Total number of positive patches: 17
Total number of negative patches: 4
IoU ship: 0.7423185423650062

Total number of positive patches: 44
Total number of negative patches: 41
IoU ship: 0.4570701909757704

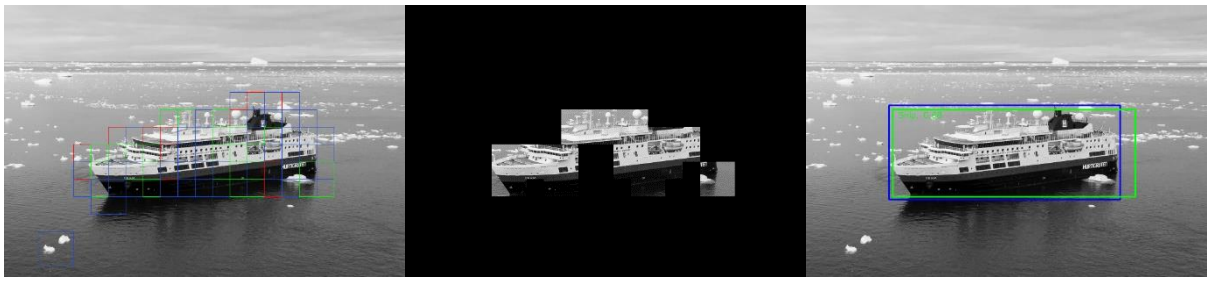
Total number of positive patches: 34
Total number of negative patches: 21
IoU ship: 0.9264117827998625



Total number of positive patches: 39
Total number of negative patches: 7
IoU ship: 0.754541494484277

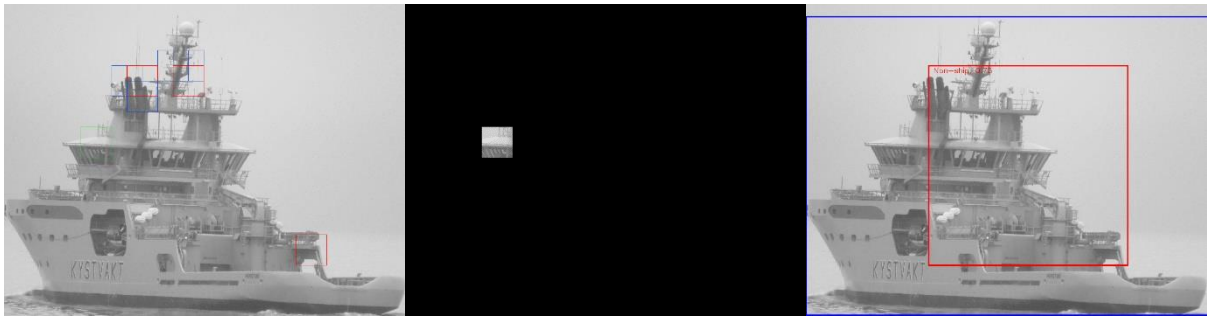


Total number of positive patches: 44
Total number of negative patches: 9
IoU ship: 0.5293052885200419

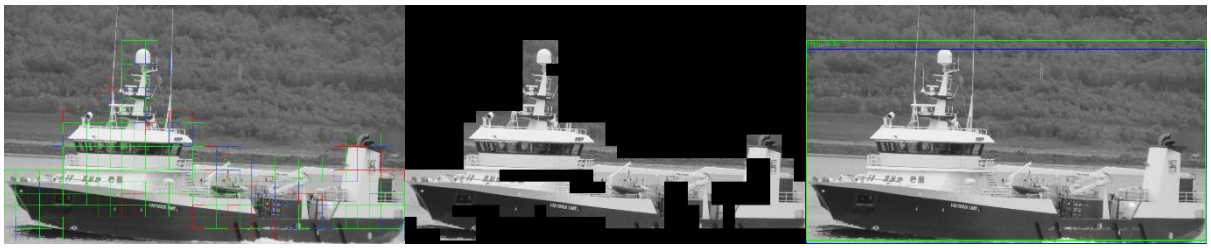


Total number of positive patches: 17
Total number of negative patches: 8
IoU ship: 0.8582046087617118

Total number of positive patches: 45
Total number of negative patches: 20
IoU ship: 0.6099484698706489



Total number of positive patches: 1
Total number of negative patches: 3
IoU ship: 0.332280440265994



Total number of positive patches: 107
Total number of negative patches: 21
IoU ship: 0.9362934968790106

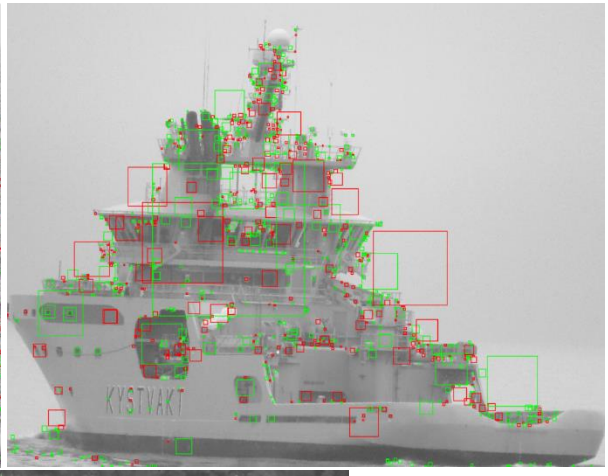
Specification:

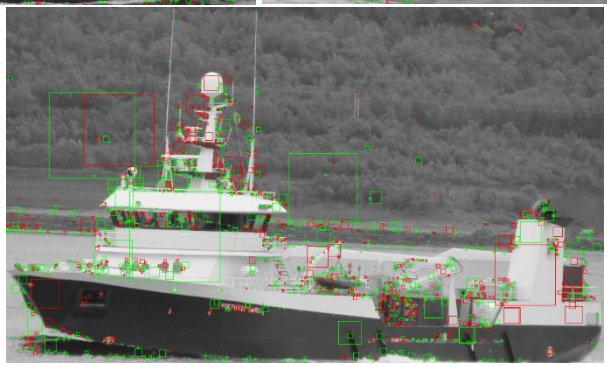
SIFT_param	Default (contrastThreshold = 0.06)
N_clusters_ship	100
N_clusters_nonship	100
init	“k-means++”





LDRW


KYSTVAKT

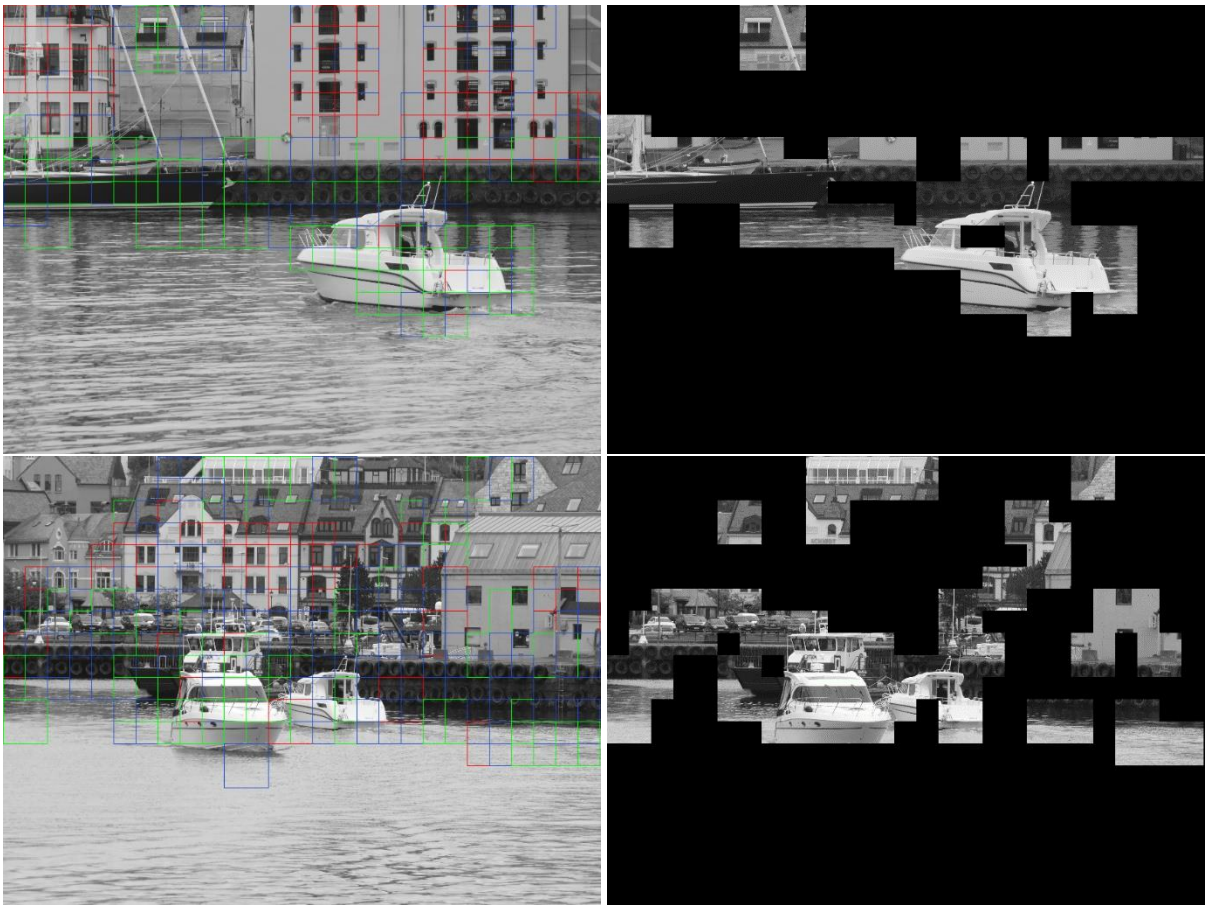


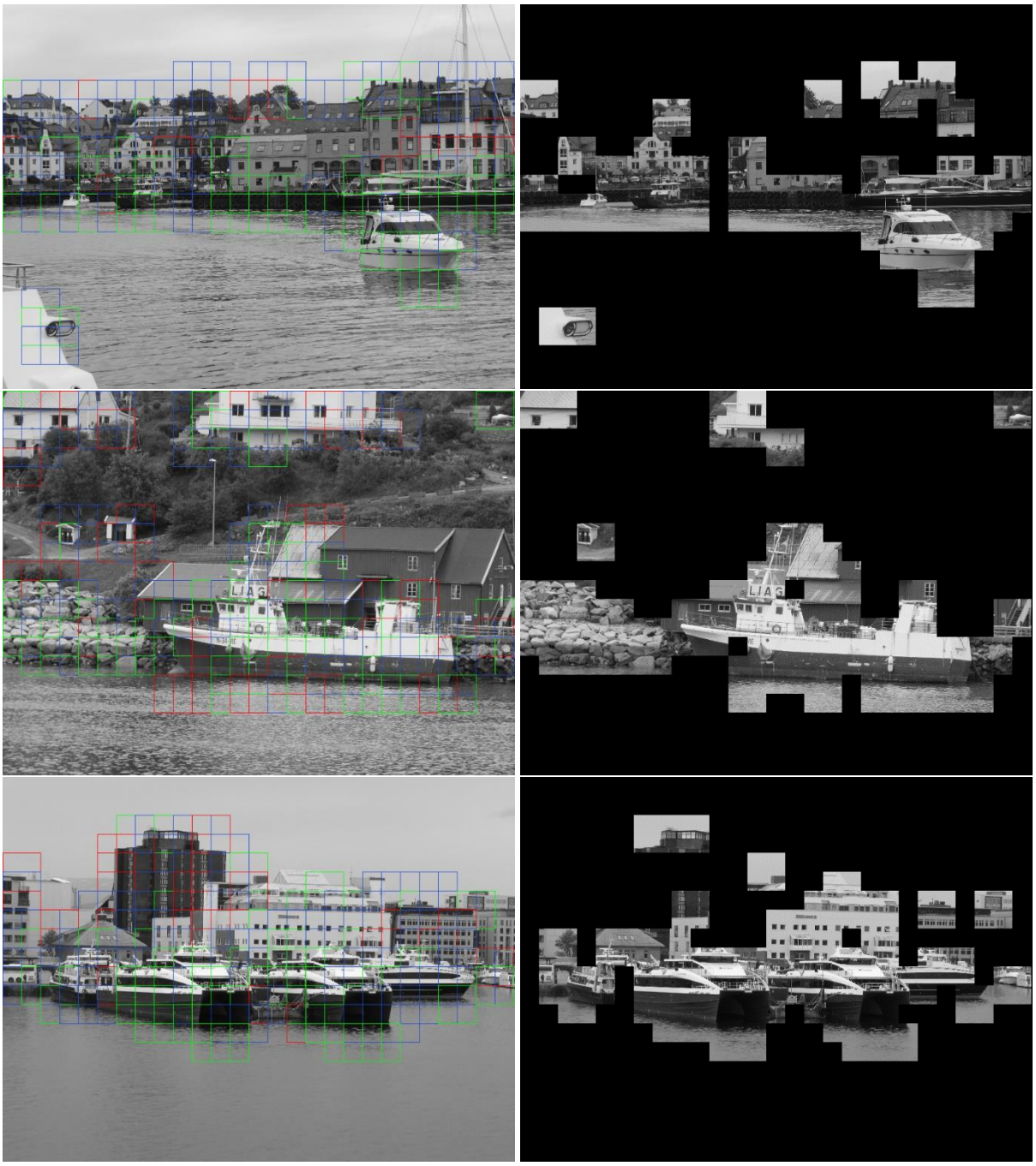
Appendix B

Results: Multi-object Detection

Specification

variance_threshold_train	1500
hog_param	[9, (8,8), (2,2)]
diff_threshold	5
N_clusters_ship	100
N_clusters_nonship	100
init	“k-means++”
variance_threshold_test	1500
T_classification	0.1



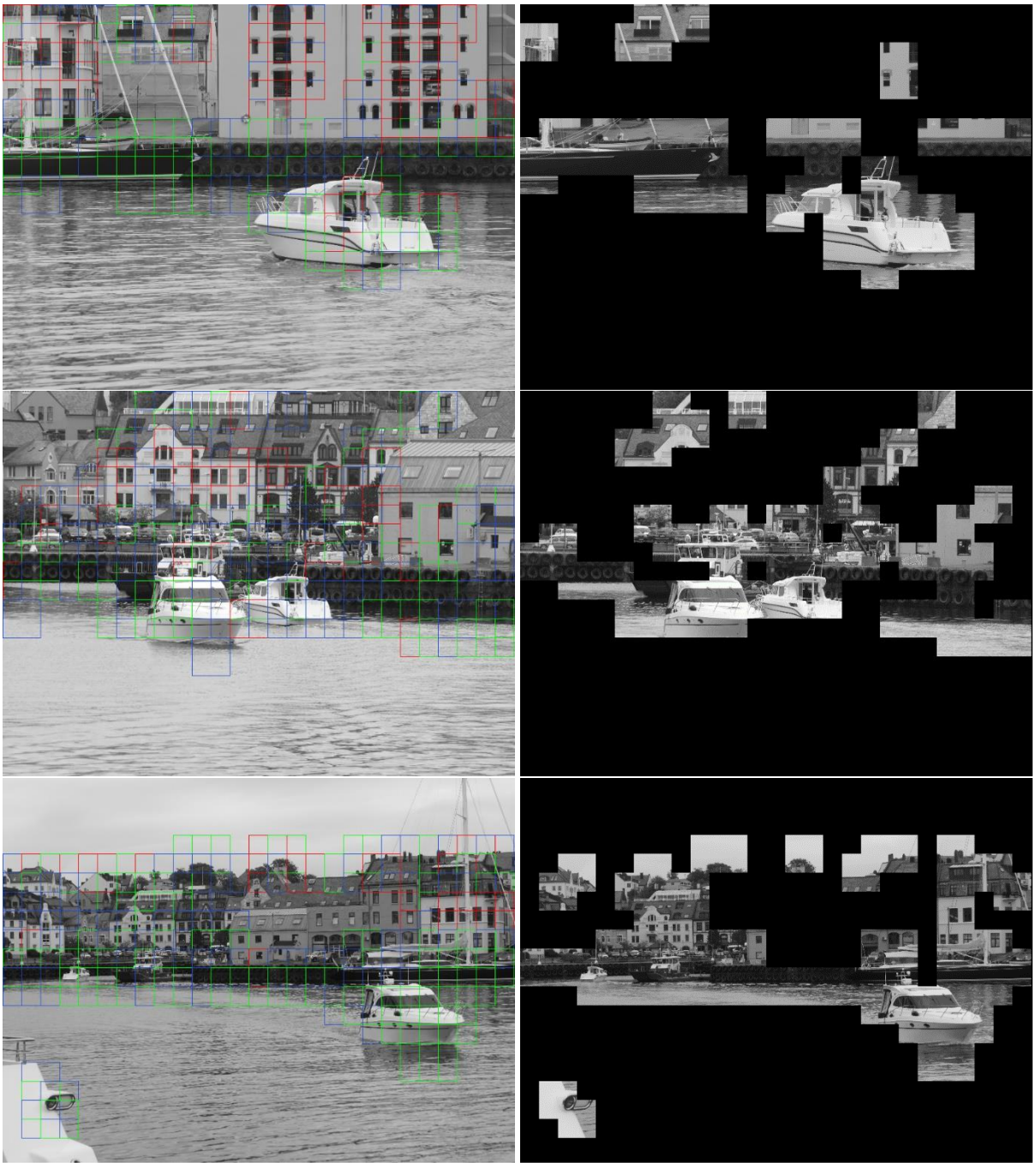


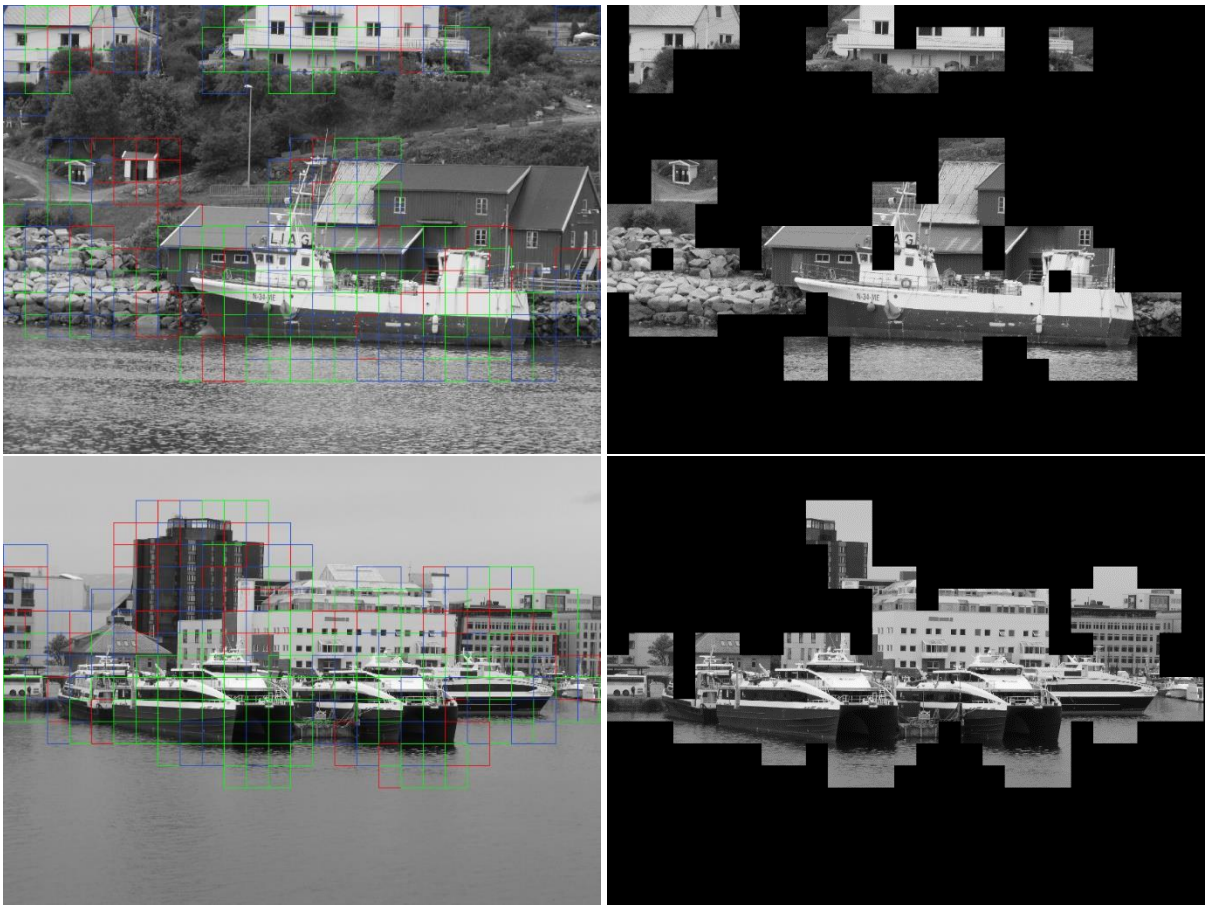
Specification

variance_threshold_train	1500
hog_param	[9, (8,8), (2,2)]
diff_threshold	5
N_clusters_ship	500
N_clusters_nonship	500
init	“k-means++”
variance_threshold_test	1500
T_classification	0.1

Specification

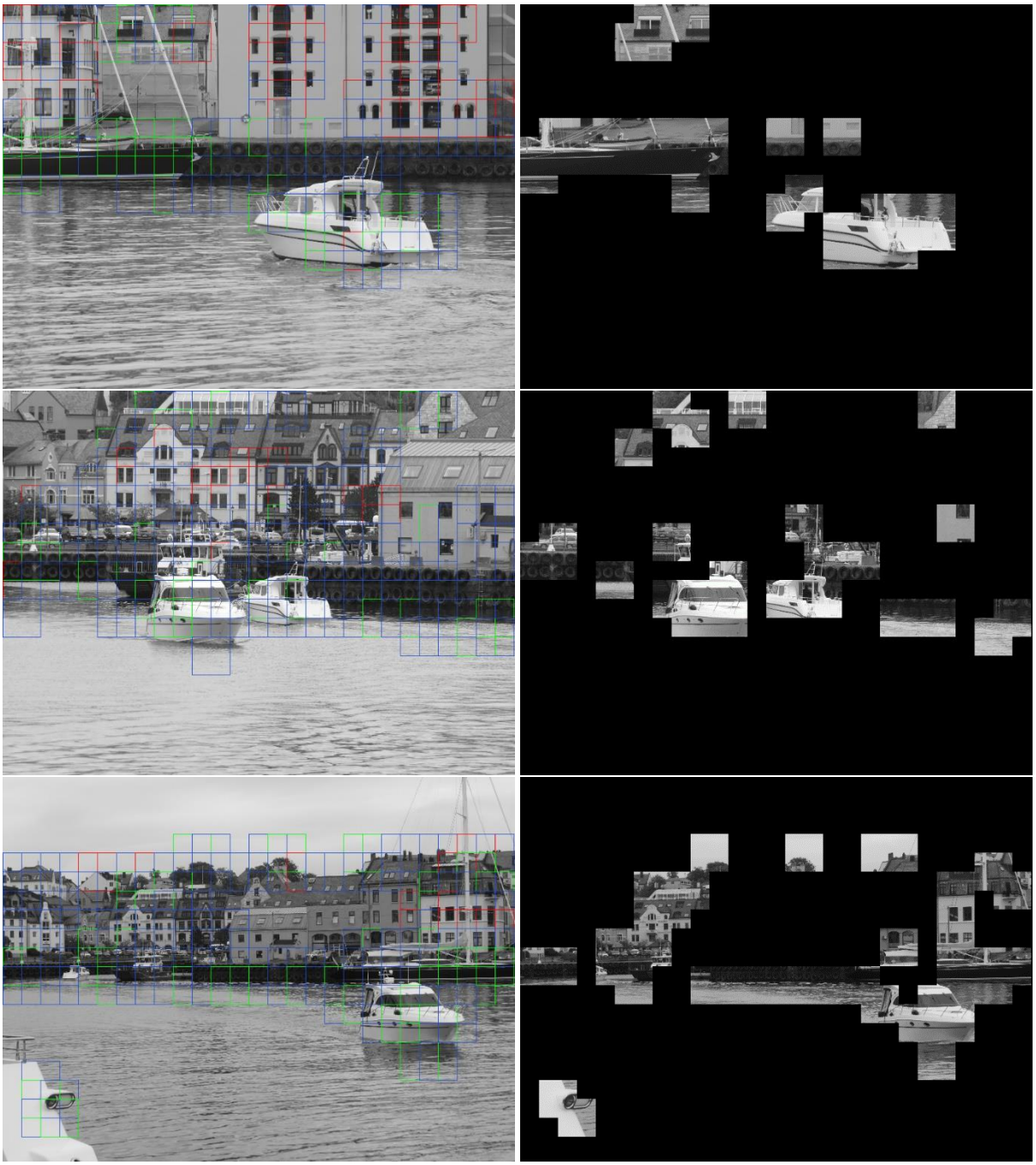
variance_threshold_train	1500
hog_param	[9, (8,8), (2,2)]
diff_threshold	5
N_clusters_ship	1000
N_clusters_nonship	1000
init	“k-means++”
variance_threshold_test	1500
T_classification	0.1

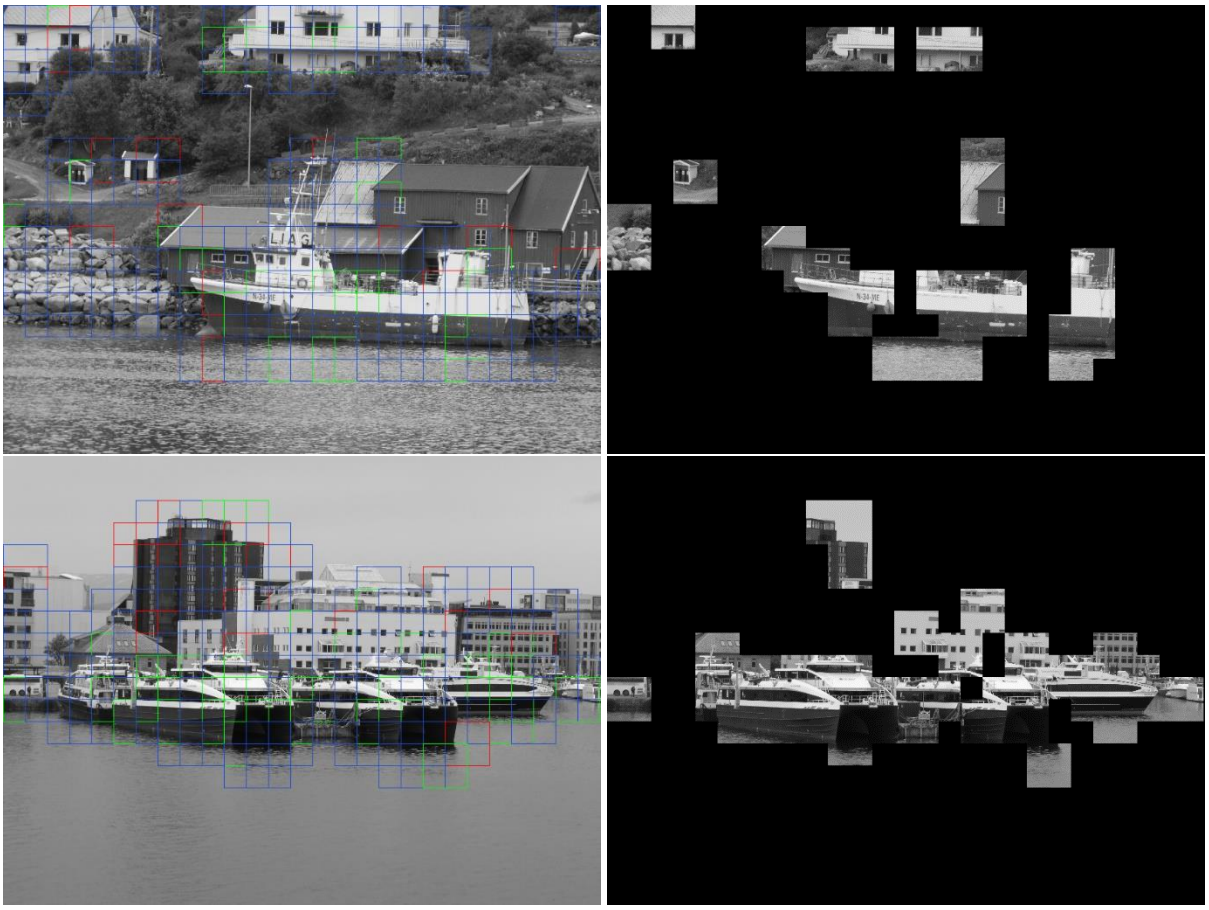




Specification

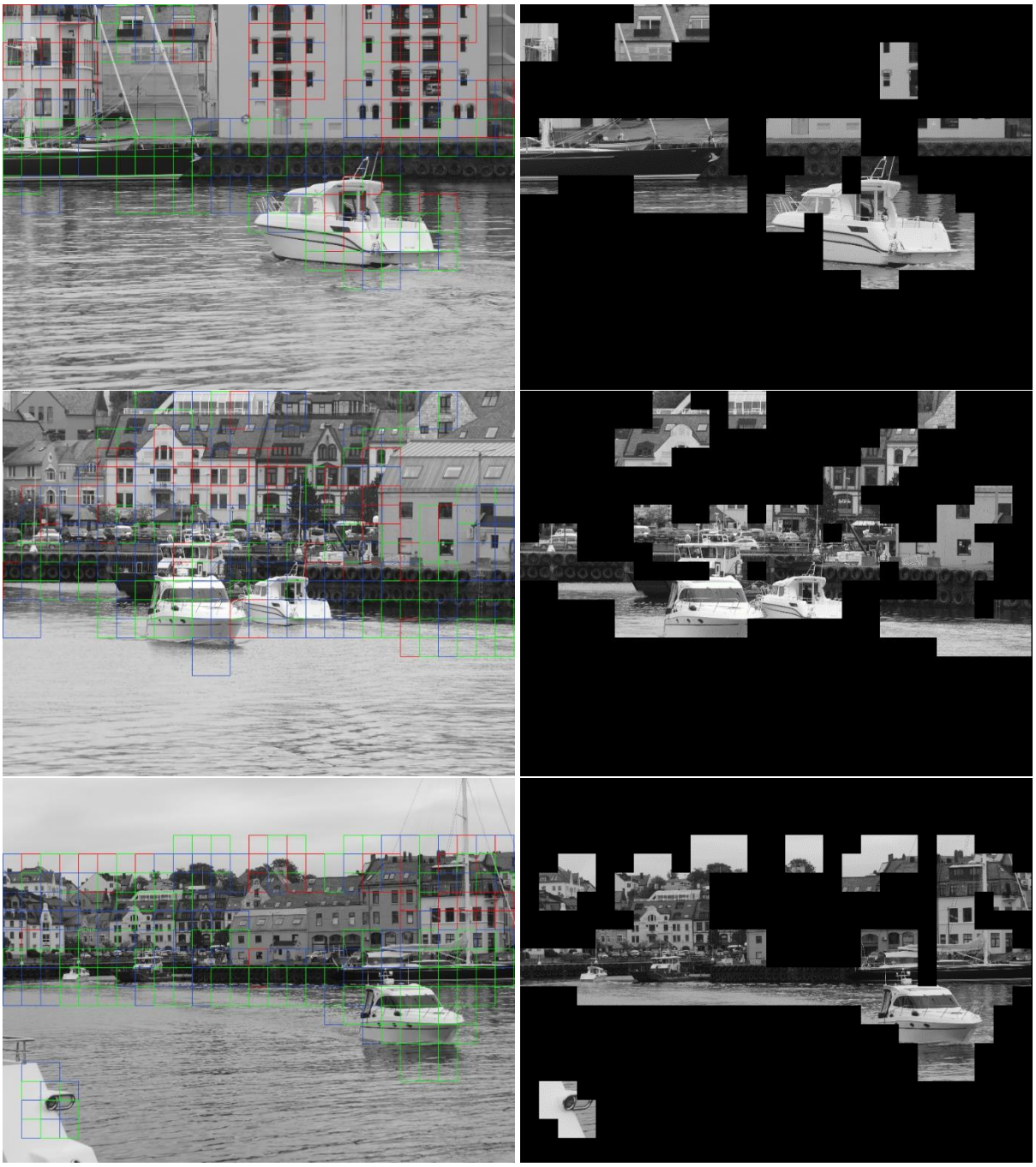
variance_threshold_train	1500
hog_param	[9, (8,8), (2,2)]
diff_threshold	5
N_clusters_ship	1000
N_clusters_nonship	1000
init	“k-means++”
variance_threshold_test	1500
T_classification	0.2

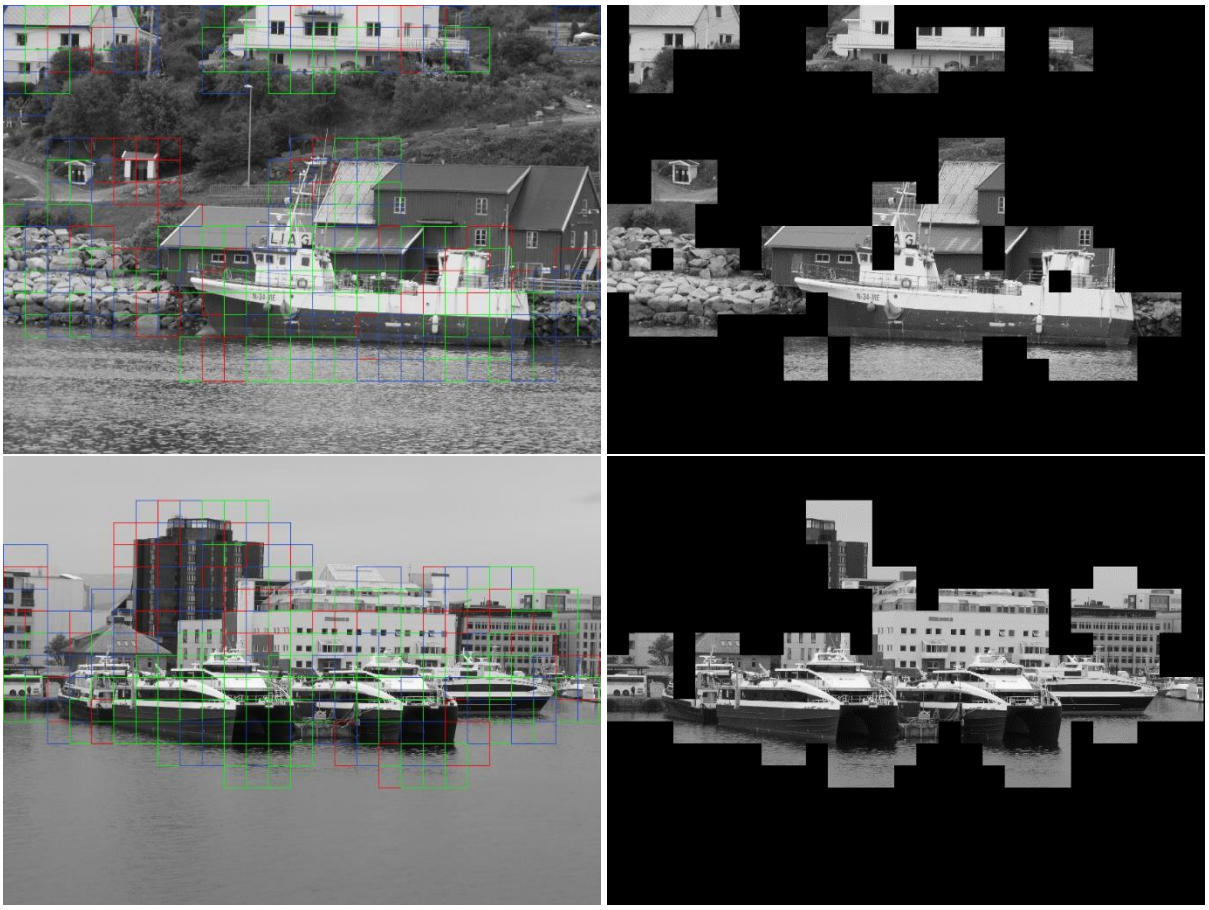




Specification

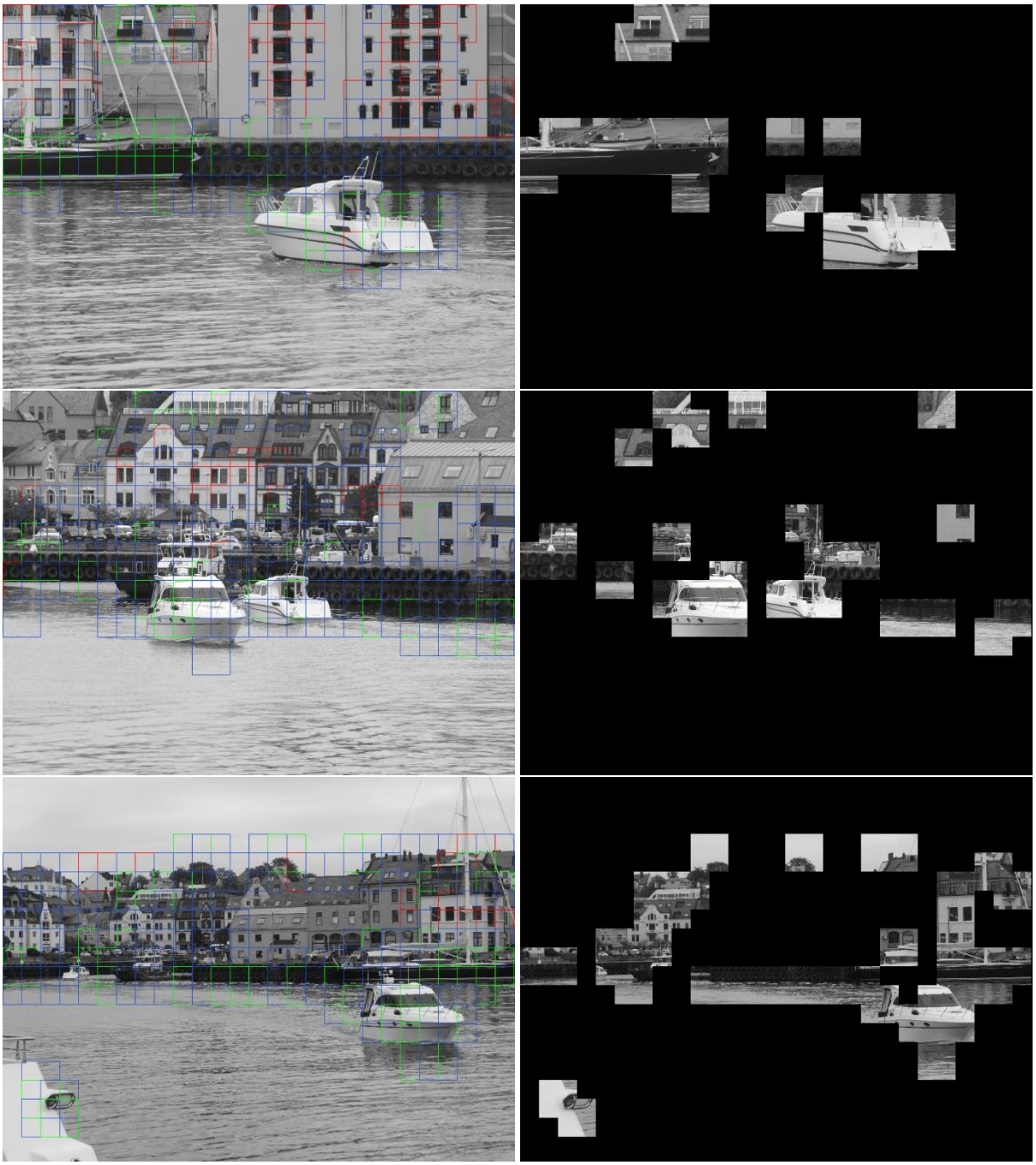
variance_threshold_train	1500
hog_param	[9, (8,8), (2,2)]
diff_threshold	5
N_clusters_ship	1000
N_clusters_nonship	500
init	“k-means++”
variance_threshold_test	1500
T_classification	0.1

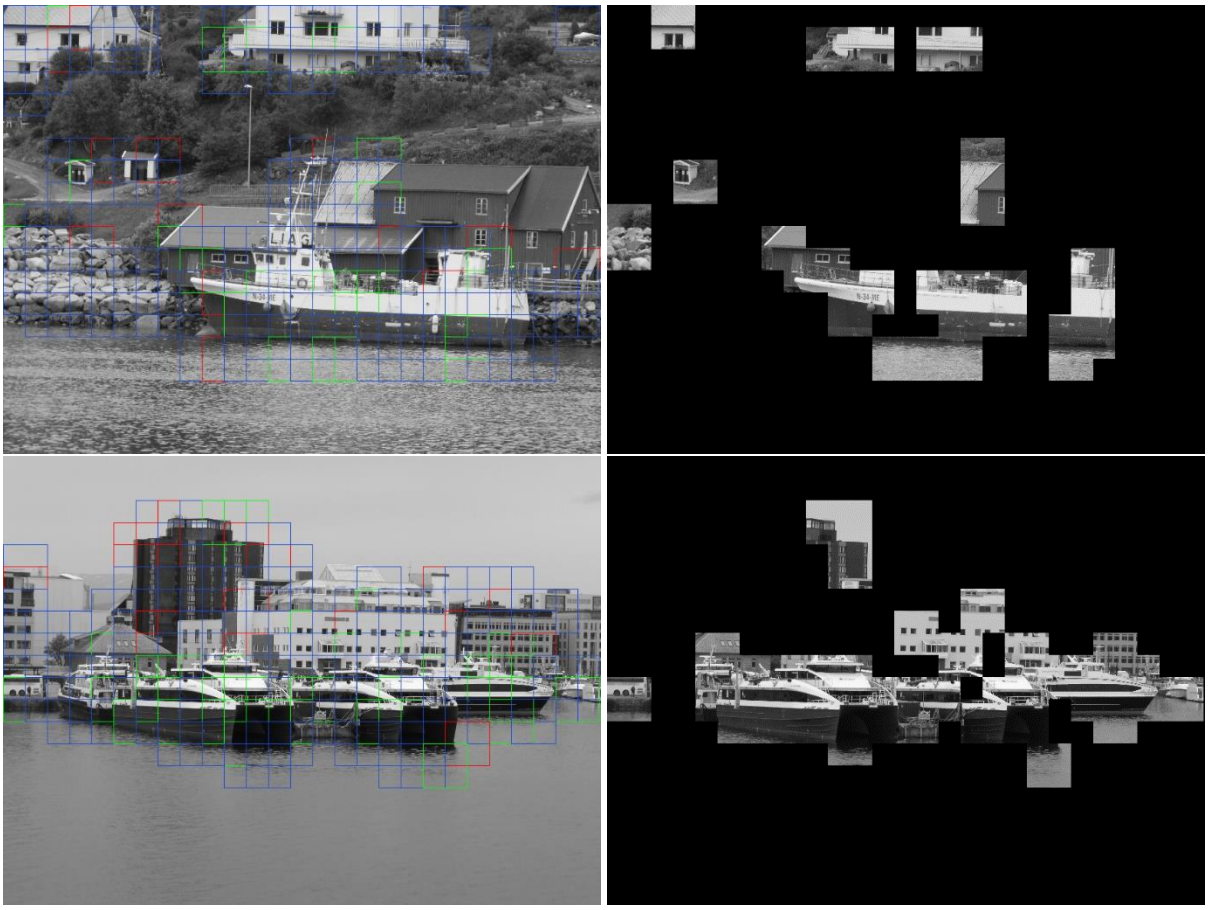




Specification

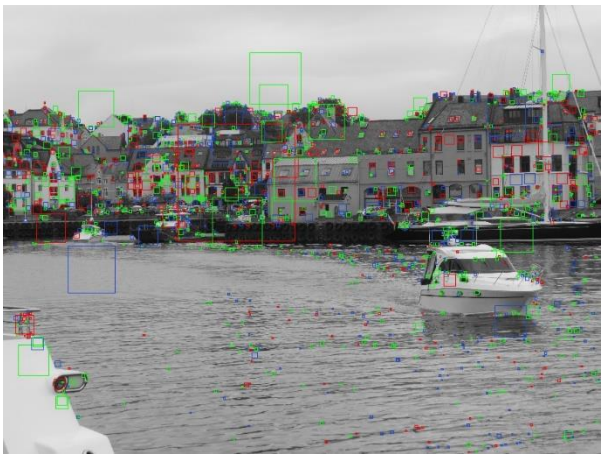
variance_threshold_train	1500
hog_param	[9, (8,8), (2,2)]
diff_threshold	5
N_clusters_ship	1000
N_clusters_nonship	500
init	“k-means++”
variance_threshold_test	1500
T_classification	0.2

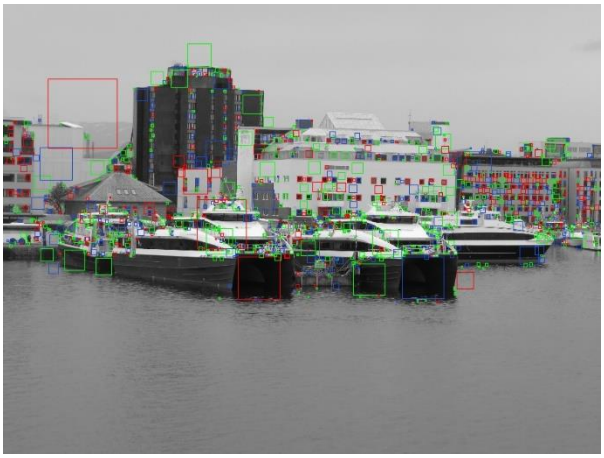


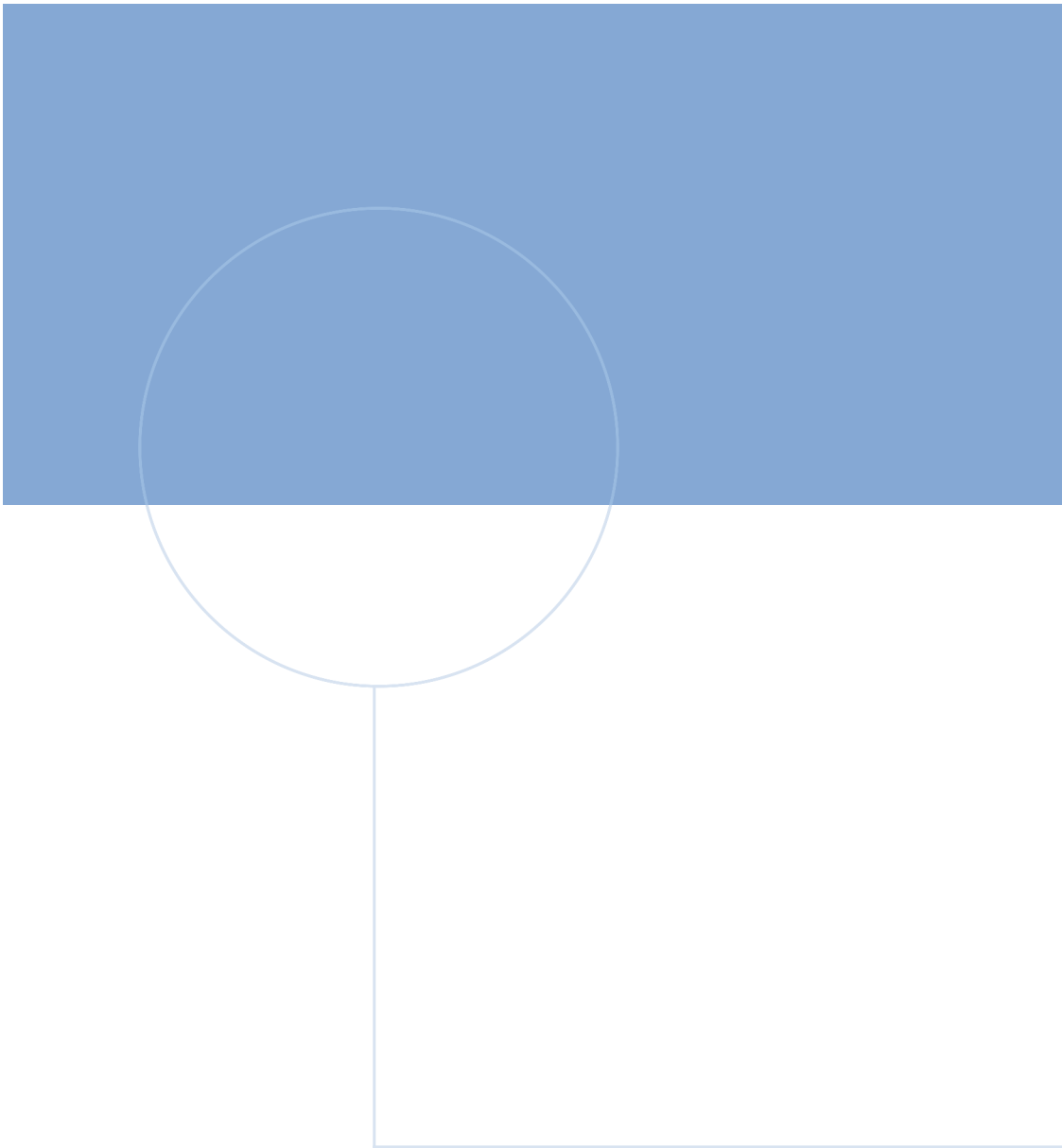


Specification:

SIFT_param	Default (contrastThreshold = 0.06)
N_clusters_ship	5000
N_clusters_nonship	5000
init	“k-means++”





NTNU

Kunnskap for en bedre verden