3-29-2012

Algorithms for Visual Maritime Surveillance with Rapidly Moving Camera

Sergiy Fefilatyev
University of South Florida, sfefilatyev@gmail.com

Follow this and additional works at: https://digitalcommons.usf.edu/etd

Part of the American Studies Commons, Computer Engineering Commons, and the Computer Sciences Commons

Scholar Commons Citation

This Dissertation is brought to you for free and open access by the USF Graduate Theses and Dissertations at Digital Commons @ University of South Florida. It has been accepted for inclusion in USF Tampa Graduate Theses and Dissertations by an authorized administrator of Digital Commons @ University of South Florida. For more information, please contact digitalcommons@usf.edu.
Algorithms for Visual Maritime Surveillance with Rapidly Moving Camera

by

Sergiy Fefilatyev

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
Department of Computer Science and Engineering
College of Engineering
University of South Florida

Major Professor: Dmitry B. Goldgof, Ph.D.
Lawrence O. Hall, Ph.D.
Rangachar Kasturi, Ph.D.
Lawrence Langebrake, M.S.

Date of Approval:
March 29, 2012

Keywords: Non-Stationary Camera, Ship Detection, Horizon Detection, Vanishing Line, Stochastic Texture, SIFT Keypoints

Copyright © 2012, Sergiy Fefilatyev
DEDICATION

This dissertation is dedicated to my parents, Larisa and Nikolay Fefilatyev, who encouraged and supported me during the academic years.
ACKNOWLEDGEMENTS

I would like to express words of sincere gratitude to my major professor Dr. Dmitry Goldgof for giving me opportunity to work in the field of computer vision under his supervision. Thank you, Dr. Lawrence Hall and Dr. Rangachar Kasturi, for assisting me during the years of graduate school and providing constant scientific input and feedback. I would also like to extend my gratitude to Mr. Larry Langebrake from SRI International and Mr. Chad Lembke from the Center of Ocean Technology for giving the idea for the project, consulting in the field of marine science and providing image and video data for experimental work.
# TABLE OF CONTENTS

LIST OF TABLES iii  
LIST OF FIGURES iv  
ABSTRACT vi  

CHAPTER 1 INTRODUCTION 1  
1.1 Motivation 1  
1.2 Related Work 3  
  1.2.1 Visual Surveillance in Maritime Domain 3  
  1.2.2 Horizon Detection and Localization 5  
1.3 Contributions 7  
1.4 Organization of Dissertation 8  

CHAPTER 2 BACKGROUND 10  
2.1 Area-Based Image Registration 10  
2.2 Linear Regression with Total Least Squares 14  
2.3 Kalman Filter 17  
2.4 Scale Invariant Feature Transform 19  
  2.4.1 Scale-Space Extrema Detection 20  
  2.4.2 Keypoint Localization 23  
  2.4.3 Orientation Assignment 25  
  2.4.4 Keypoint Descriptor 26  

CHAPTER 3 ALGORITHMS FOR DETECTION AND TRACKING OF SHIPS IN OPEN SEA 28  
3.1 Overview of Algorithms 30  
3.2 Horizon Detection 32  
3.3 Image Registration 38  
3.4 Segmentation 42  
3.5 Tracking 44  

CHAPTER 4 SHIP DETECTION EVALUATION METRICS 48  
4.1 Performance of Horizon Presence Detection Algorithm 48  
4.2 Performance of Horizon Localization Algorithm 49  
4.3 Performance of Ship Detection and Localization 50
## LIST OF TABLES

<table>
<thead>
<tr>
<th>Table</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.1</td>
<td>Distance measures for estimation of presence/absence of horizon line in an image.</td>
<td>37</td>
</tr>
<tr>
<td>5.1</td>
<td>Datasets used to train and evaluate the algorithms.</td>
<td>55</td>
</tr>
<tr>
<td>5.2</td>
<td>Values of parameters used during performance evaluation.</td>
<td>55</td>
</tr>
<tr>
<td>5.3</td>
<td>Results of evaluation of horizon presence detection algorithm.</td>
<td>56</td>
</tr>
<tr>
<td>5.4</td>
<td>Value of precision and recall of detection on the sequence level according to various values of the evaluation threshold ET3.</td>
<td>59</td>
</tr>
<tr>
<td>5.5</td>
<td>Value of precision and recall of detection on the sequence level according to various values of minimal height of objects.</td>
<td>60</td>
</tr>
<tr>
<td>6.1</td>
<td>Values of parameters in the algorithm for vanishing line detection.</td>
<td>75</td>
</tr>
<tr>
<td>6.2</td>
<td>Performance of vanishing line detection according to two setups.</td>
<td>76</td>
</tr>
</tbody>
</table>
# LIST OF FIGURES

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Figure 1.1</td>
<td>Concept of a buoy-based maritime surveillance system.</td>
<td>2</td>
</tr>
<tr>
<td>Figure 2.1</td>
<td>Example of image registration.</td>
<td>11</td>
</tr>
<tr>
<td>Figure 2.2</td>
<td>Principle of area-based image registration.</td>
<td>13</td>
</tr>
<tr>
<td>Figure 2.3</td>
<td>Geometrical interpretation of simple and total least squares.</td>
<td>16</td>
</tr>
<tr>
<td>Figure 2.4</td>
<td>Signal-flow graph representation of a linear discrete-time dynamical system.</td>
<td>18</td>
</tr>
<tr>
<td>Figure 2.5</td>
<td>Selecting minima/maxima in a scale space.</td>
<td>21</td>
</tr>
<tr>
<td>Figure 2.6</td>
<td>Obtaining scale space with difference of Gaussians.</td>
<td>22</td>
</tr>
<tr>
<td>Figure 2.7</td>
<td>Interpretation of SIFT descriptor.</td>
<td>27</td>
</tr>
<tr>
<td>Figure 3.1</td>
<td>Categories of horizon images.</td>
<td>29</td>
</tr>
<tr>
<td>Figure 3.2</td>
<td>Outline of the detection and tracking algorithm.</td>
<td>30</td>
</tr>
<tr>
<td>Figure 3.3</td>
<td>Outline of the horizon detection algorithm.</td>
<td>33</td>
</tr>
<tr>
<td>Figure 3.4</td>
<td>Registration of frames in global coordinate system.</td>
<td>39</td>
</tr>
<tr>
<td>Figure 3.5</td>
<td>Steps of the segmentation algorithm.</td>
<td>43</td>
</tr>
<tr>
<td>Figure 3.6</td>
<td>Outline of the tracking algorithm.</td>
<td>47</td>
</tr>
<tr>
<td>Figure 5.1</td>
<td>Examples of tracking multiple targets from a camera installed on a buoy.</td>
<td>54</td>
</tr>
<tr>
<td>Figure 5.2</td>
<td>Accuracy of horizon localization.</td>
<td>57</td>
</tr>
<tr>
<td>Figure 5.3</td>
<td>Precision and recall of detection on the sequence level according to various values of threshold ET2.</td>
<td>60</td>
</tr>
<tr>
<td>Figure 5.4</td>
<td>Precision and recall of detection on the sequence level according to various minimum heights of objects.</td>
<td>61</td>
</tr>
<tr>
<td>Figure 6.1</td>
<td>Example of detecting vanishing line by matching SIFT keypoints.</td>
<td>64</td>
</tr>
</tbody>
</table>
Figure 6.2  Example of supporting vectors and vanishing points generated by pairs of matched SIFT keypoints.  67
Figure 6.3  Geometry of vanishing point triangulation. 71
Figure 6.4  Estimating direction of the vanishing line from an ensemble of vanishing points.  72
Figure 6.5  Examples of vanishing line detection on real images of water surface.  78
ABSTRACT

Visual surveillance in the maritime domain has been explored for more than a decade. Although it has produced a number of working systems and resulted in a mature technology, surveillance has been restricted to the port facilities or areas close to the coastline assuming a fixed-camera scenario. This dissertation presents several contributions in the domain of maritime surveillance. First, a novel algorithm for open-sea visual maritime surveillance is introduced. We explore a challenging situation with a camera mounted on a buoy or other floating platform. The developed algorithm detects, localizes, and tracks ships in the field of view of the camera. Specifically, our method is uniquely designed to handle a rapidly moving camera. Its performance is robust in the presence of a random relatively-large camera motion. In the context of ship detection, a new horizon detection scheme for a complex maritime domain is also developed. Second, the performance of the ship detection algorithm is evaluated on a dataset of 55,000 images. Accuracy of detection of up to 88% of ships is achieved. Lastly, we consider the topic of detection of the vanishing line of the ocean surface plane as a way to estimate the horizon in difficult situations. This allows extension of the ship-detection algorithm to beyond open-sea scenarios.
CHAPTER 1
INTRODUCTION

1.1 Motivation

The maritime transportation system is important to the world’s economy for accommodation of freight traffic, recreational travel, commercial fishing, and offshore-resources management [1]. In addition to its economic significance, the marine transportation systems are also vital to national security [2]. Thus, adequate management and monitoring of the littoral and maritime areas are key elements of economic prosperity, national security, environmental, and recreational improvement [3].

Surveillance systems play an important role in this process by providing tools for situational awareness, threat assessment, and decision making. Various sources of surveillance, monitoring, maritime safety information are available. These include the Automatic Identification System (AIS) [4], Vessel Monitoring Systems (VMS) [5], air- and space-borne SAR systems [6], ship- and land-based radars, air- and space-borne optical sensors, harbor-based visual surveillance. During the last decade technology has been moving towards integration of several information sources, of which one essential component is vision. Fusion of vision with other information sources allows more accurate and descriptive monitoring of coastal areas, maritime borders, and offshore assets. Examples of this trend include systems that integrate AIS/VMS with SAR-imagery [7], radar- and visual-based surveillance for ports [8, 9], land-radars with visual information from air-borne platforms [10], and ship-based systems that integrate radar with visual information [11].
Figure 1.1. Concept of a buoy-based maritime surveillance system. Surveillance is limited by visual distance to targets. Data from a forward looking camera results in targets located on the horizon line (sub-picture in the bottom right corner).

A new concept [12, 13] integrates information from radars/AIS with vision sensors placed on autonomous buoys. Such an autonomous unmanned buoy-system is an attractive choice for integrated surveillance for reasons of cost, persistent ocean presence, form-factor, and flexibility. A network of vision sensors with on-board processing and bi-directional communication can provide a real-time intelligence for critical maritime areas. It can detect, classify, identify and track small vessels at sea often associated with issues of illegal immigration, smuggling, and poaching. Examples of the use of vision information from such sources include verification of vessel’s class, intent, and behavior, as well as description of state and status of the vessels. A passive vision sensor is also more desirable in military applications because it does not reveal the location of the surveillance system.

This work focuses on detection of marine vehicles using imaging sensors such as digital cameras or camcorders. The scope of surface area for monitoring is limited by visual distance from some point in the ocean or shore. This may be appropriate for some tasks such as port security or sanctuary protection. In essence, the limitation and effectiveness of the approach are similar to the periscope of a submarine: marine
vehicles are visually located in image data obtained from a camera sticking out of the water surface. An ocean buoy is considered a primary platform for such a system.

1.2 Related Work

We relate our contribution to other developments in the area with respect to several different aspects. First, in Section 1.2.1 we review the existing applications in visual maritime surveillance mostly related to automated target detection and to surveillance for port security. Second, in Section 1.2.2 we will review algorithms for precise detection and localization of horizon line which is going to be used in the ship detection algorithm.

1.2.1 Visual Surveillance in Maritime Domain

Automated visual surveillance in maritime domain has developed from two significant trends in security-based applications: Automated Target Recognition (ATR) systems and harbor surveillance. These two trends are similar in some of their functionality, such as detection, localization, tracking, and recognition of targets, but differ in terms of the environment they are used in and the sensors they use for data acquisition.

Most ATR systems are developed for the military and are to designed to be air or sea-borne. They rely on forward-looking infrared (FLIR) cameras as a primary source of the imagery because infrared sensors are less-sensitive to variations in illumination and appearance changes, which are common in maritime scenes. A notable disadvantage of FLIR systems is the low resolution of images and significant power consumption which makes these systems less capable for autonomous operation. Literature related to ATR systems focus mostly on correct classification of the detected target rather than detection [14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25]. Some
notable works in segmentation for FLIR imagery include [26, 27, 28] which are interesting because they assume non-stationary FLIR cameras. However, FLIR imagery is very different in nature from images from visible spectrum cameras and most of these methods for segmentation are not suitable for the situation with a non-stationary visible-spectrum camera. A silhouettes-based approach for classification so common for FLIR is not applicable for the visible spectrum as contours of the ships binarized from the visible spectrum vary significantly due to higher variance in illumination and appearance.

Harbor surveillance is the other significant application extensively studied [9, 29, 30, 31, 32, 8]. In general, all the existing systems for port surveillance assume that the camera is in a fixed position, and the water surface is fixed in terms of a camera view. With this set of assumptions, algorithms for background subtraction or background suppression are mostly used. Visual frameworks for those stationary conditions and background/foreground modeling in the context of harbor surveillance have been explored for more than a decade [14, 15]. Many of the mentioned papers describe already existing systems [9, 30, 8], signifying matureness of the topic and the spread of the commercial applications. Reported functionality of such systems include vessel detection, discrimination between vessel classes, comparison of the detections with the notifications of arrival received by the port, and geo-registration of ships that navigate in port waters [33, 34].

The most significant applications not related to ATR or harbor surveillance are classification (in the optical spectrum and in the non-harbor environment) [35, 36], image stabilization [37, 38], and detection and tracking in the non-stationary maritime environment [11]. Stabilization and horizon detection are two important parts of our algorithm and related work on these topics is reviewed separately in Chapter 3.
The system described in [11] is the closest match to the solution of a buoy-based surveillance problem. The authors bring up many issues that arise in an intelligent video surveillance system for ships such as segmentation of objects in complex maritime scenes, multi-target tracking, target classification, decision support, and activity recognition. However, performance of the described detection and tracking algorithm may be limited in a buoy-based scenario. This conclusion is based on the fact that the most important components related to segmentation, and tracking rely on two assumptions that are not valid for the case of a buoy-based non-stationary camera. First, the background-subtraction algorithms in general and the specific algorithm proposed in [11] are sensitive to the dynamics of the scene. Our implementation of the background subtraction algorithm in [11] does not show robustness on the data obtained from a buoy-based camera that experiences high-magnitude rapid motion in pitch, yaw, and bank dimensions. Second, the assumption that the ships in the frame of view are located in water regions below the horizon is not valid in our case. We expect the camera in a possible buoy-application to be parallel to the ocean’s surface and be located at relatively low heights. Also, the authors of [11] do not provide a quantitative evaluation of the detection, localization and tracking approaches used in their method. Our work aims to cover the issues of a very dynamic scenes with targets on-or-above the horizon line, and also provide performance evaluation of the proposed approach.

1.2.2 Horizon Detection and Localization

Because of the significance of the horizon detection in our surveillance scheme it is also important to briefly review the existing approaches to horizon detection and analyze how they are relevant to our situation.
The two main application domains for horizon estimation are unmanned aerial vehicle (UAV) navigation and vanishing line estimation. In the case of UAV navigation, the horizon line is used as an alternative to inertial sensors [39, 40, 41, 42, 43, 44]. The horizon line is a valuable attitude reference and is used to estimate the pitch and roll of the UAV with a forward looking camera. Applications not included in the UAV or camera calibration categories are given in [45, 11].

The vanishing line approaches [46, 33, 47, 48, 49] are aimed at locating a vanishing line of the horizon which may or may not be the actual visible horizon. All the methods from this category of approaches are related to a fixed camera and, thus, are not applicable to our case. The only method that could be used in a non-stationary environment is the approach utilizing the nature of surface’s texture [48]. Such a geometry-based solution does not provide sufficient precision for horizon localization compared to the direct methods of horizon detection in open sea. However, such solution may be integrated in the horizon detection scheme for more difficult scenarios such as images of the sea with the coast visible. In Chapter 6 a new method for vanishing line detection is described which has the potential for such integration.

The category of UAV-derived algorithms is more appropriate to the situation with a non-stationary camera. There are several approaches in this category to find the horizon line. In [42, 43] the horizon line is found by analysis of the projection profiles of the edge map of an image. The algorithm works well with those images that contain the horizon, however, it does not address the situation when the horizon line is not present in the image. The algorithm in [44] detects several candidate horizon lines by using morphological operations and the Hough transform [50]. The final horizon line is selected using optical flow applied to each of the candidate lines. However, the use of optical flow is inappropriate for the buoy-based situation, as we expect a large interframe motion from the camera. Optical flow is also less desirable for autonomous
systems because of the computational load it involves. The Hough transform as a basic approach to detect the horizon line is also used in [11]. The horizon is assumed to be always present in the image. The horizon detection algorithm described in [41] addresses both situations: when the horizon line is visible and when it is absent. The approach classifies the pixels into sky/ground regions and finds the boundary between them. Since the classification power is incorporated in the beginning of such a scheme, it easily detects non-horizon images. A representative algorithm developed in [39, 40] relies on a color-based statistical model of the sky and ground (sea in our case). The algorithm searches for the straight line which maximizes the sky-ground separation criterion. The authors also offer a solution for extreme attitude case - when only sky or ground regions are visible by validating the current results of the horizon detection with the accumulated statistics of the sky/ground appearance. The main difference between the last two approaches is that the training in [41] is done in a supervised manner, while in [39, 40] it is semi-supervised: the algorithm detects the horizon line in the first batch of images assuming that the horizon is present and then, using the found horizon, creates a statistical model of the background for the sky and ground. Other approaches in horizon detection [51, 52] use different image acquisition techniques and, thus, are not appropriate for our case.

1.3 Contributions

The contributions of this dissertation are as follows.

First, we present an algorithm for the on-board processing of image and video-data obtained by a buoy-based maritime surveillance system in open sea. This envisions a system that operates autonomously and intelligently in a maritime area away from the visible coast-line. The algorithm discussed here includes topics of detection, localization, and tracking of ships or other marine vehicles of interest in imagery
taken by a camera that is mounted on a buoy and is subject to a rapid random motion associated with the buoy’s flotation. Common problems related to buoy-based maritime imagery such as low-contrast profiles of targets, large inter-frame motion, and compression artifacts are addressed. Good experimental results of the proposed methods are shown on a set of 55,000 frames of video obtained from the camera installed on an untethered buoy. The performance of the proposed scheme is evaluated for the most important characteristics: detection, localization, and tracking accuracy.

Second, we present a novel approach from Shape From Texture [53] category which will further improve the system’s performance and address more complex maritime scenery: sea surface with coastal objects or artificial objects present. For that, we investigate estimation of the vanishing line of a plane using the gradient of perspective distortion of the sea-surface texture. The approach is based on detection of individual vanishing points through matching SIFT features [54] of plane texture and finding a representative set of such vanishing points that support a hypothesized vanishing line. The performance of the method is evaluated on a separate dataset of images of ocean surface.

1.4 Organization of Dissertation

The rest of the dissertation is structured as follows. Chapter 2 introduces important concepts and framework used in algorithms for open sea ship detection, horizon detection, and detection of the vanishing line of a plane under perspective projection. Chapter 3 presents the algorithms for frame stabilization, segmentation, detection, and tracking of ships in the open sea. The datasets and description of the metrics used to evaluate the performance of the algorithms are addressed in Chapter 4. Chapter 5 reports experimental evaluations on data obtained from a prototype of
a buoy-based surveillance system. Chapter 6 presents theoretical contribution for
detection of the horizon through the vanishing line using texture. Conclusions are
drawn in the last chapter.
CHAPTER 2
BACKGROUND

2.1 Area-Based Image Registration

Image registration methods transform two or more sets of image data into one coordinate system in order to align images of the same scene [55, 56, 57]. For example, several photographs taken at different times need to be aligned in order to detect or compare changes in environment, medical condition, attitude of the camera (see Figure 2.1). In some applications images may differ in resolution and spectrum and be produced by different instruments. Registration in this case is often performed to integrate such disparate images into a single, enhanced visualization of the scene.

The process of registration requires compensating for geometric aberrations, examples of which include differences in camera orientation and distance from the imaged object, sensor resolution and dynamic range, and other factors. One of the images in this process is referred to as the reference or source and the second and other images are referred to as the target(s).

Image registration algorithms can be classified into feature-based and area-based and generally consist of the following steps:

- Detecting Features. Features can be points, lines, corners, or patches that can potentially be detected and identified in the aligned set of images.

- Matching corresponding features. In other words, for each feature in one image a corresponding feature in the other image(s) needs to be determined.
Figure 2.1. Example of image registration. Three images of parts of the Florida peninsula are stitched together to obtain a single integrated map of the area.
• Obtaining a geometric transformation function that maps features in one image onto the locations of the matching features in the other image(s). Usually a particular parametric transformation model is chosen, based on image capture geometry.

• Transforming image data using the transformation function to align one image with the other.

Transformation functions are generally divided into two categories: global and local. The global transformation is characterized by the same function everywhere in the image. Examples of such transformation include affine, projective, polynomial, etc. For many image registration problems, the geometric correspondence between features in the two images is too complex to be characterized by a single transformation function that applies everywhere. For such problems, a local transformation function with locally varying parameters may be used.

The feature-based registration establishes correspondence between a number of points in images. Having the correspondence established, a transformation function is then determined to map the target image to the reference images. In the area-based registration the transformation function is only described by a global affine transformation. In order to find such a transformation the target image, called the template image, is shifted to cover each location in the reference image (see Figure 2.2). At each location, an area-based similarity metric is computed. A distinct peak in the similarity metric found at a certain position suggests a match of the alignment between the images at that position.

Several area-based similarity metrics have been proposed in the image processing literature. Those metrics differ in such characteristics as computation time and robustness against outlier pixels.
Figure 2.2. Principle of area-based image registration. The template image is shifted to cover each location in the reference image to compute a similarity metric. A distinct peak is the similarity suggests the correct alignment.

A class of metrics called the sequential similarity detection algorithms (SSDAs) [58] provides a computationally simple similarity measure $E(u, v)$ based on absolute differences between the reference image and a template:

$$E(u, v) = \sum_x \sum_y |T(x, y) - I(x - u, y - u)|$$

(2.1)

where $T$ is the template, $I$ is the window under the template in the reference image. The summation is taken over the values of $x$ and $y$ for the reference image such that the image and the template overlap. In order to account for possible difference in dynamic range in the reference and the template a normalization measure is introduced in the metric:

$$E(u, v) = \sum_x \sum_y |T(x, y) - \hat{T} - I(x - u, y - u) + \hat{I}(u, v)|$$

(2.2)

where $\hat{T}$ and $\hat{I}$ are the average intensities of the template and the local image window respectively.
A similar metric, but based on squared distances is designed to put more penalty on a match in case of outliers:

$$D(u, v) = \sum_x \sum_y (T(x, y) - I(x - u, y - u))^2$$  \quad (2.3)

One of the most used metrics for area-based registration is the normalized cross-correlation [59]. Cross-correlation is often used for template matching and recognition in statistical tasks. It maximizes the ratio of signal power to expected noise and is well-fit to deal with outliers. Normalization of the cross-correlation allows aligning images with different dynamic ranges. The metric is defined by the following relation:

$$\gamma(x, y) = \frac{\sum_x \sum_y [T(x, y) - \bar{T}] [I(x - u, y - v) - \bar{I}]}{\sqrt{\sum_x \sum_y [I(x - u, y - v) - \bar{I}]^2 \sum_x \sum_y [T(x, y) - \bar{T}]^2}}$$  \quad (2.4)

The value $\gamma(x, y)$ ranges from -1 to 1. A high value for $\gamma(x, y)$ indicates a good match between the template and the image, when the template is centered at coordinates $(x, y)$.

2.2 Linear Regression with Total Least Squares

The method of least squares is a standard approach used in the field of optimization and data fitting. It is used in overdetermined systems, when the number of observations is greater than the number of unknowns. The method minimizes the sum of the squared differences between the observations and fitted values of the model.

Least squares methods are generally divided into two categories: ordinary (linear) least squares and non-linear least squares, depending on whether or not the
residuals are linear in all unknowns. The linear least-squares problem has had wide application in statistical regression analysis [60]. It has a closed-form solution and can be expressed by a simple relation. The non-linear problem has no closed-form solution and is usually solved iteratively.

The linear least square problem can be further divided into problems with uncertainties only in dependent variables and with uncertainties in both dependent and independent variables. While the former has found a much wider use in practical applications because of its simplicity, it is only applicable when uncertainties are present in the observed dependent variables. The latter category, total least squares, is more expressive, as it allows more general linear models be built (see Figure 2.3). In this section a solution for the total least squares is presented.

The solution to total least square for $AX = B$ that minimizes error matrices $E$ and $F$ for $A$ and $B$ can be expressed with the help of singular value decomposition (SVD). $A$ is an $m$-by-$n$ augmented matrix of independent variables and $B$ is an $m$-by-$k$ matrix of observations. Formally, such optimization is expressed as

$$\argmin_{E,F} \| [E F] \|_F, \quad (A + E)X = B + F$$

(2.5)

where $[E F]$ is the augmented matrix with $E$ and $F$ side by side and $\| \cdot \|_F$ is the square root of the sum of squares of the lengths of the rows or columns of the matrix.

The relation (2.5) can be written as

$$\begin{bmatrix} (A + E) & (B + F) \end{bmatrix} \begin{bmatrix} X \\ -I_k \end{bmatrix} = 0.$$ 

(2.6)
Figure 2.3. Geometrical interpretation of simple and total least squares. In simple least squares the residuals are calculated as vertical distances between the observations and the fit values. In total least squares the residuals represent the minimum distances to the fit line.

where $I_k$ is the $k \times k$ identity matrix. Let $[U][\Sigma][V]^*$ be the SVD of the augmented matrix $[A \ B]$:

\[
[A \ B] = [U_A \ U_B] \begin{bmatrix} \Sigma_A & 0 \\ 0 & \Sigma_B \end{bmatrix} \begin{bmatrix} V_{AA} & V_{AB} \\ V_{BA} & V_{BB} \end{bmatrix}^* = [U_A \ U_B] \begin{bmatrix} \Sigma_A & 0 \\ 0 & \Sigma_B \end{bmatrix} \begin{bmatrix} V_{AA}^* & V_{BA}^* \\ V_{AB}^* & V_{BB}^* \end{bmatrix} \tag{2.7}
\]

where $V_{XX}$ is the part of SVD’s $V$ partitioned into blocks corresponding to the shape of $A$ and $B$.

After some algebraic manipulations detailed in [61]:

\[
[(A + E) \ (B + F)] \begin{bmatrix} -V_{AB}V_{BB}^{-1} \\ -V_{BB}V_{BB}^{-1} \end{bmatrix} = [(A + E) \ (B + F)] \begin{bmatrix} X \\ -I_k \end{bmatrix} = 0 \tag{2.8}
\]

and

\[
X = -V_{AB}V_{BB}^{-1}. \tag{2.9}
\]
The obtained vector $X$ is the vector of parameters of the linear model.

2.3 Kalman Filter

The Kalman filter [62], [63] provides a recursive solution to the linear optimal filtering problem. Its domain of applications includes stationary and non-stationary environments. The solution is recursive; each updated estimate of the state is computed from the previous estimate and the new input data. The Kalman filter needs to store only the previous estimate, eliminating the need for storing the entire past observed data, and, thus is computationally more efficient than computing the estimate from the entire past observed data at each step of the filtering process.

The block diagram shown in Figure 2.4 describes a linear, discrete-time dynamic system. The state vector, denoted by $x_k$, is defined as the minimal set of data that uniquely describes the unforced dynamical behavior of the system; the subscript $k$ denotes discrete time. The state vector is the least amount of data on the past behavior of the system that is sufficient to predict its future behavior. In general, the state $x_k$ is unknown. To estimate it, a set of observed data, denoted by the vector $y_k$, is used. Process noise $w_k$ and measurement noise $v_k$, shown in the diagram, are represented by covariance matrices $Q$ and $R$, and are considered constant. The following set of equations describes the dynamic system mathematically:

- Process equation

\[
    x_{k+1} = F_{k+1,k} x_k + w_k
\]  

(2.10)

where $F_{k+1,k}$ is the transition matrix, which relates the state $x_k$ from time $k$ to time $k + 1$ in the absence of noise. The process noise $w_k$ is assumed to be additive, white, with normal probability. Its probability distribution has zero
mean with a covariance matrix defined by

$$E[w_n w_k^T] = \begin{cases} Q_k & \text{for } n = k \\ 0 & \text{for } n \neq k \end{cases}$$ (2.11)

where $T$ denotes matrix transposition. The dimension of the state space is denoted by $m$.

- Measurement equation

$$y_k = H_k x_k + v_k$$ (2.12)

where $y_k$ is the observable at time $k$ and $H_k$ is the measurement matrix. The measurement noise $v_k$ is assumed to be additive, white, and normal probability. Its probability has zero mean and with covariance matrix defined by

$$E[v_n v_k^T] = \begin{cases} R_k & \text{for } n = k \\ 0 & \text{for } n \neq k \end{cases}$$ (2.13)
The measurement noise $v_k$ is uncorrelated with the process noise $w_k$. The dimension of the measurement space is denoted by $n$.

The Kalman filtering problem, namely, the problem of jointly solving the process and measurement equations for the unknown state in an optimum manner may now be formally stated as follows: use the entire observed data, consisting of the vectors $y_1, y_2, \ldots, y_k$, to find for each $k \geq 1$ the minimum mean-square error estimate of the state $x_k$. The problem is called filtering if $i = k$, prediction if $i > k$ and smoothing if $1 \leq i \leq k$. Detailed techniques for solving Kalman filter problem is shown in [62].

2.4 Scale Invariant Feature Transform

Scale-Invariant Feature Transform (SIFT) was initially proposed by David Lowe in 1999 [64] and fully described in 2004 [54] as a method to improve object recognition, multi-view matching, and object tracking. The method allows detection and description of local image features based on their appearance. The advantages of SIFT include scale and orientation invariance, robustness to illumination, viewpoint change, and partial inclusion, high distinctiveness and ease of computation.

SIFT transforms the image into a collection of keypoints and descriptors specific to particular locations of the image called SIFT features. Once collected, the SIFT features can be matched against other SIFT features from other images in order to find the same objects. Invariance to scale or orientation changes allows stitching images of the same scenery taken from different viewpoints or with different cameras if some parts of the scenes intersect.

The main parts of the algorithm include the following steps:

- Space-scale extrema detection to generate Keypoint candidates.

- Localization and filtering of the detected keypoints to obtain a stable set of well-localized local features.
• Orientation assignment to achieve orientation invariance. All operations with
the keypoints are performed on image data transformed according to the chosen
scale and orientation.

• Computation of SIFT descriptors for each of the keypoints in the stable set.

2.4.1 Scale-Space Extrema Detection

In the first stage of SIFT, identification of location and scale of interest points
is performed. The interest points, or interest features, of the object are important
because they are assigned uniquely to the same parts of the object. In order to
achieve scale invariance for such features a search across all possible scales needs to
be done. Several steps are performed for that purpose as described below.

The input image is convolved with a special space-scale kernel based on the
Gaussian function [65]:

\[ L(x, y, k\sigma) = G(x, y, k\sigma) \ast I(x, y) \]  \hspace{1cm} (2.14)

where \( G \) is the Gaussian function

\[ G(x, y, \sigma) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}} \]  \hspace{1cm} (2.15)

and \( \ast \) is the convolution operation in \( x \) and \( y \), and \( L(x, y, k\sigma) \) is the result of the
convolution. The convolution is performed at different scales and the differences of
successive Gaussian-blurred images \( D(x, y, \sigma) \) are taken:

\[ D(x, y, \sigma) = L(x, y, k_i\sigma) - L(x, y, k_j\sigma) \]  \hspace{1cm} (2.16)

where \( k_i \) and \( k_j \) are constant multiplicative factors of two nearby scales \( i \) and \( j \).
Figure 2.5. Selecting minima/maxima in a scale space. If the central pixel, marked with a cross sign, represents minima/maxima when comparing to its 26 neighbors it is selected as a candidate interest keypoint.

The initial interest keypoints are identified as local minima/maxima of difference of Gaussians by comparing each pixel in scale-space to its neighbors as shown in Figure 2.5. Each pixel is compared with 26 neighbors: with eight of its neighbors in its own scale and 18 corresponding neighboring pixels in the scale above and below. If the pixel’s value is the minima/maxima among those 26 neighbors it is selected as a candidate interest point.

An efficient discrete method of constructing the scale space is shown in Figure 2.6. An octave is a set of images obtained from a single image by convolving it with several Gaussian kernels with increasing values of $\sigma$ separated by a constant factor. Each octave is composed of an integer number of images processed by Gaussian kernel with incrementally larger $\sigma$, such that the total change of $\sigma$ from the first to the last image in the octave represents doubling of scale. Adjacent image scales are used to generate difference of Gaussians as shown on the right of the Figure 2.6. Once the octave is processed the first image for the next octave is obtained by resampling the last image of the previous octave by taking every second pixel.
Figure 2.6. Obtaining scale space with difference of Gaussians. Each octave represents doubling of scale. Images within each octave represent convolutions with the Gaussian kernel with incrementally larger $\sigma$ separated by a constant factor. Adjacent Gaussian-smoothed images are subtracted to produce difference of Gaussians which are used in local maxima/minima search.
2.4.2 Keypoint Localization

Many of the detected scale extrema points are unstable due to a possible low contrast appearance [54] or because their location is poorly defined along edges. The keypoint localization step aims to localize such detected points to a subpixel accuracy and filter out the points are likely to be unstable. This is done by fitting the local scale space extrema to the data from neighboring pixels and verifying the ratio of principal curvatures.

Fitting the scale space extrema is done through interpolation of the difference of the Gaussian scale space function \( D(x, y, \sigma) \) with quadratic Taylor expansion:

\[
D(x) = D + \frac{\partial D^T}{\partial x} x + \frac{1}{2} x^T \frac{\partial^2 D}{\partial x^2} x
\]  

(2.17)

where, \( x = (x, y, \sigma) \) is an offset of the candidate point. The location of the extremum, \( \hat{x} \), is determined by taking the derivative of (2.17) with respect to \( x \) and setting it to zero:

\[
\hat{x} = -\frac{\partial^2 D^{-1} \partial D}{\partial x^2} \frac{\partial D}{\partial x}
\]  

(2.18)

If the location of extrema \( \hat{x} \) is larger than the original candidate point by 0.5 in any dimension the extrema lies closer to a different sample point. In this case, the candidate keypoint is changed and the interpolation is performed around that point. Otherwise, the found location is used as the real location of the extrema. The described interpolation of the location substantially improves matching and stability accuracy as shown in [54].

Keypoints with low contrast are filtered out by comparing the value of the second-order Taylor expansion \( D(x) \) computed at \( \hat{x} \) with the value of the original candidate.
If this value is less than 0.03, the candidate keypoint is discarded. Otherwise, the keypoint is retained.

The robustness of keypoint selection is further increased by passing the selected points through edge response elimination filter. Typically, many detected points are located on the edge, but because of the inherent noise their location they show up as extrema points in scale space with poorly defined peaks. In order to filter them out, an approach similar to the one adopted in the Harris corner detector [66] is used. The principal curvatures across and along the edge are compared using 2x2 Hessian matrix $H$, computed at the location and scale of a keypoint:

$$
H = \begin{bmatrix}
D_{xx} & D_{xy} \\
D_{xy} & D_{yy}
\end{bmatrix}
$$

The eigenvalues of $H$ are proportional to the principal curvatures of $D(x)$. For computational purposes, however, it is more appropriate to calculate the ratio of eigenvalues indirectly. The trace of $H$, $D_{xx} + D_{yy}$, gives us the sum of the two eigenvalues, while its determinant, $D_{xx}D_{yy} - D_{xy}^2$, gives the product of eigen values. The ratio $R = \frac{\text{Tr}(H)^2}{\text{Det}(H)}$ as shown in [54] is equal to $\frac{(R + 1)^2}{R}$. Because $R$ is minimum when the eigenvalues are equal, the higher the absolute difference between the two eigenvalues, the higher the value of $R$. In order to filter out the keypoint poorly localized along the edge, the value $R$ for a candidate keypoint is compared with a user-specific threshold eigenvalue-ratio $R_s$. If $R$ is larger than $\frac{(R_s + 1)^2}{R_s}$, that keypoint is rejected.
2.4.3 Orientation Assignment

In order to achieve rotation invariance, each keypoint is assigned one or more orientation depending on gradient direction around the keypoint similarly to approach in [67]. The keypoint descriptor is calculated relative to such orientation(s).

The computation of the orientation is performed in a scale-invariant manner by selecting a Gaussian smoothed image at the location and scale of the keypoint \( L(x, y, \sigma) \). For an image sample \( L(x, y) \) at scale \( \sigma \), the gradient magnitude, \( m(x, y) \), and orientation, \( \theta(x, y) \), are computed using pixel differences:

\[
m(x, y) = \sqrt{(L(x + 1, y) - L(x - 1, y))^2 + (L(x, y + 1) - L(x, y - 1))^2} \tag{2.20}
\]

\[
\theta(x, y) = \tan^{-1}\left(\frac{L(x, y + 1) - L(x, y - 1)}{L(x + 1, y) - L(x - 1, y)}\right) \tag{2.21}
\]

The magnitude \( m \) and direction \( \theta \) of the gradient are computed for every pixel in a neighboring region around the keypoint in the Gaussian-blurred image \( L \). An orientation histogram is formed from gradient orientations of sample points within the region. The histogram has 36 bins, each bin covering 10 degrees. Each sample added to a histogram bin is weighted by its gradient magnitude and by a Gaussian-weighted circular window with a \( \sigma \) that is 1.5 times that of the scale of the keypoint. The peaks in this histogram correspond to dominant orientations. Once the histogram is filled, the orientations corresponding to the highest peak and other local peaks that are within 80% of the highest are assigned to the keypoint. When such approach produces multiple orientations, an additional keypoint is created having the same location and scale as the original keypoint for each additional orientation.
2.4.4 Keypoint Descriptor

The keypoint found in previous steps is characterized by its location, scale, and orientation. In order to be able to match the keypoint to other keypoints in a dataset, a reliable descriptor of the local appearance needs to be computed. SIFT descriptor described below is highly distinctive and is partially invariant to affine distortion and change of illumination. The computation of the descriptor is done in the neighborhood of the keypoint and at the scale and orientation previously found.

A set of 16 orientation histograms are created on 4x4 pixel neighborhoods with 8 bins each. Similarly to computation of orientation in Section 2.4.3, the histograms are computed from magnitude and orientation values of samples in such neighborhood: the value of orientation is weighted by the magnitude of the gradient. The magnitudes are further weighted by a Gaussian function with $\sigma$ equal to one half the width of the descriptor window. The SIFT descriptor concatenates the values of these histograms. Since there are $4 \times 4 = 16$ histograms each with 8 bins the SIFT vector has 128 elements. This vector is normalized to unit length in order to enhance invariance to affine changes in illumination. The effect of non-linear illumination is reduced by applying a threshold of 0.2 after which the vector is again normalized. Figure 2.7 shows interpretation of SIFT descriptor obtained on a local neighborhood of the keypoint on a gradient image.
Figure 2.7. Interpretation of SIFT descriptor. A Gaussian-smoothed image is used to compute gradients along eight directions. Gradients from a subregion of 4x4 pixels are summarized in an orientation histogram. 4x4 such subregions create 16 orientation histograms, which, after concatenation, create a 128-element SIFT descriptor.
In the application of a buoy-based visual maritime surveillance, the camera is installed approximately parallel to the ocean surface. Such a forward-looking camera has a limited field of view (which depends on the optics) and has a limited resolution. The acquired data comes from the visible part of the spectrum and is represented by the RGB-color model. The focus of the camera is set to infinity, and, thus, allows detection only of far away objects located just above the horizon line. Because the camera is firmly attached to its non-stationary buoy-platform the scenery it observes is directly dependent on the uncontrolled motion of the buoy. In this work we assume that such motion of the camera is erratic, even though flotation patterns of the platform are generally repeatable. Thus, motion pattern of the buoy provide us with the three categories of possible images. The situation when the optical axis of the moving camera lies close to parallel to the ocean surface will result in horizon images, i.e. images where the horizon line separates the two most significant regions of the image - sky and water (Figure 3.1(a)). The other two categories of images are obtained when only sky region is visible (camera points off the water surface, see Figure 3.1(b)) or only water region are visible (see Figure 3.1(c)).

An important constraint in our model is that possible objects of interest, i.e. ships, comprise only a small fraction of the field of view, and thus, most of the scenery is represented by a water and sky regions. An exception occurs when such
Figure 3.1. Categories of horizon images. Camera that is attached to the untethered buoy will provide images of several possible categories: (a) Images with horizon clearly discernable. (b) Camera points to the sky and sky is the only scenery visible in the image. (c) Water surface images. (d) Objects such as ships when they take a big fraction of the field of view may seriously affect horizon detection. (e) Blur that comes from water splashes is another factor affecting horizon detection.

a constraint is not held (see Figure 3.1(d)-(e)), such condition is detected, and the results are not affected.

We address all the mentioned cases by relying on one important feature that is always present in maritime scenery: the horizon line. Our approach can be summarized as the following:

- Images with high confidence in the location of the horizon line are considered for further analysis.

- High confidence in the absence of the horizon line in an image is an indication that the camera is pointing to the sky or water region only, and thus, no further search of the ships is possible and such image is disregarded.

- Low confidence in the location of the horizon line addresses the situation when an exceptional situation occurs: field of view is blocked or quality of imagery is corrupted by a water splash. Images with such conditions are detected and taken away from the processing pipeline.
Figure 3.2. Outline of the detection and tracking algorithm. Algorithm consists of four main modules: horizon detection, image registration, segmentation, and tracking. Depending on the confidence of horizon detection or registration the current frame can be dropped from consideration.

### 3.1 Overview of Algorithms

The algorithm for the detection and localization of ships uses a four-step strategy to find possible targets. Each of those components may either produce output for the consecutive step or take away the current image frame from the processing pipeline. It is important to rule out the inappropriate frame early in the process, if it is found that model assumptions are not satisfied because no correct detection is possible. In that case the frame is labeled as 'intractable' and the algorithm moves onto the next frame in the image sequence. Figure 3.2 shows the basic structure of our approach and the order of how the components of the algorithm are used.
Horizon detection is the base step in the algorithm for several reasons. The found horizon line is used to determine if the current frame satisfies the model assumptions about water and sky regions. The horizon line is also used in the following image registration step as a reference line for image alignment. Finally, the horizon line is used in the segmentation step to reduce the search space of all possible targets as objects of interest are expected to be above the horizon line and also adjacent to it.

The purpose of the image registration module is to register all frames of the video sequence in the common coordinate system [56]. High confidence in the correct alignment of horizon images is another verification that the observed scenery satisfies model assumptions. Low confidence in the correct alignment indicates that the horizon detector failed and a new detection of the horizon is required. However, the most important role of the registration step is the ability of subsequently track the detected targets. As described before, the camera is a subject to a rapid random motion associated with the buoy’s floatation. This results in large inter-frame motion in the field of view that lends to ambiguity in the correspondence of targets between frames in the video sequence. Correctly registered frames of the video sequence allow tracking of the targets on the horizon with a linear Kalman filter [68]. Details on the implementation of the image registration module are given in Section 3.3.

The segmentation step of the algorithm localizes the regions of the image above the horizon that potentially contain ships. The most important factor affecting localization is the appearance of ships, which depends on the illumination, orientation, and weather condition. Low-contrast profiles of ships are also subject to compression artifacts of the video encoder of the camera. Output of the segmentation step is important for subsequent tracking, because segmented targets are used as an input into the tracking algorithm. Section 3.4 provides details on the implementation of the segmentation algorithm.
The tracking step is used to increase the accuracy of ship detection and reduce false alarms. Due to limited interframe motion of the camera an assumption is made that the object present in the current frame will be present in the next frame in the vicinity of the current location. The size and appearance of the detected objects must also show consistency between frames. The tracking module accepts as its input the output from the segmentation algorithm and rules out targets that do not have consistent history of location, size, and appearance. The tracking algorithm provides the final output of detected and localized targets in the obtained imagery. Details on its implementation are given in Section 3.5.

The modules in Figure 3.2 provide confidence for their output. This allows us to streamline the processing of the data by stopping further processing of the frames which have been found to be a poor fit for the assumed model. For example if the horizon detection step shows low confidence in the horizon, or high confidence in the absence of horizon, further processing of the frame is halted. For such frames it is impossible to detect ships with other steps of the algorithm because they assume the correct horizon line. Similarly, low confidence in the correct registration of the frame in the global coordinate system is a reason for the frame to be disregarded from further processing because the segmentation and tracking results depend on it.

3.2 Horizon Detection

The two most relevant horizon detection algorithms [41, 39] mentioned in Section 1.2 were tested in [69]. It was found that in case of presence of the horizon line in the field of view, all of the approaches worked well for the localization of the horizon line. The approach to horizon detection described in this work combines features from [41, 39, 11] and addresses situations mentioned in the beginning of Chapter 3 as described below.
Figure 3.3. Outline of the horizon detection algorithm. Several candidates lines for horizon are selected using the Hough transform. The final horizon line is checked against a statistical model to detect situations when the horizon is not present in the image and only water or sky regions are present.

Similarly to [41], the horizon is modeled as a straight line in a normal form. The RGB color is used as a measure of appearance as in [39]. For any given hypothesized horizon line, the pixels above the line are labeled as sky, and pixels below the line are labeled as water. This provides two distributions of pixels, where each pixel is represented by a vector of RGB colors. A sky-water separation criterion is a Mahalanobis distance, with which we seek to maximize intra-class difference between the distributions. However, instead of searching for horizon line in line-parameter
space to maximize the criterion as in [39], we adopted an approach of selecting a limited number of candidates first, and then, checking them against the criterion.

This, ‘candidates-first’, approach provides two benefits. First, the selection of candidates significantly reduces computational load as compared to two-dimensional search in line-parameter space. Second, in most practical cases such an approach provides better accuracy of horizon line detection when compared to the direct implementation [39]. The reason for better accuracy of horizon localization is explained by the fact that the algorithm in [39] assumes two regions, water and sky, which are homogeneous in color. This assumption seldom holds true. Because of perspective effects of a forward-looking camera and uneven illumination, water regions vary in color significantly. Similarly, sky regions are often non-homogeneous in color. This usually results in a horizon line found close to the correct position but shifted far enough that the subsequence segmentation of ships is impossible.

In order to select the candidate lines, the Hough transform on the edge map of the image is performed, similarly to [11] and [41]. Once the candidates are found the sky-water separation criterion is computed for each candidate and the candidate with the maximum score is selected for further processing.

The next step in the horizon detection pipeline is the extreme camera attitude detection, which addresses the situation when only sky or only water is present in the image. We use approach from [39], which relies on running statistical models of sky and water and a set of criterions for determining the situation of extreme attitude. The initial statistical models of sky and water are obtained from several initial images of a video sequence under assumption that the horizon is present.

Exceptional situation with the blurred images or with images where an external object is blocking the view is addressed by checking the value of the sky-water separation criterion against the threshold $T1$. When the sky/water regions do not
show much distinction, then the model of two regions separated by a line does not hold, and thus, no ships can be detected on such a horizon. In order to choose a value of $T_1$ we created a separate learning dataset of images where a boat blocks a significant part of the field of view or where the camera has been splashed by water. Once trained offline on such dataset, this value was used throughout the rest of the experiments.

The candidate line for horizon with the maximum sky-water separation criterion which passed the tests for extreme attitude of the camera and other mentioned exceptions is declared the horizon. Figure 3.3 summarizes all steps of the horizon detection algorithm and the following is the detailed description of the approach.

All hypothesized sky pixels are denoted as,

$$x_i^s = [r_i^s, g_i^s, b_i^s], \quad i \in \{1, \ldots, n^s\} \quad (3.1)$$

where $r_i^s$ denotes the value for intensity of the red channel in RGB, $g_i^s$ denotes the green channel value and $b_i^s$ denotes the blue channel value of the $i$-th sky pixel. All the hypothesized water pixels are noted in a similar manner:

$$x_i^w = [r_i^w, g_i^w, b_i^w], \quad i \in \{1, \ldots, n^w\} \quad (3.2)$$

and the following relation

$$\bigcup_{i=1}^{n_s} x_i^s = \overline{\bigcup_{i=1}^{n_w} x_i^w} \quad (3.3)$$

shows that the regions of water and sky are complements of each other in a frame. $n_w$ and $n_s$ are numbers of water and sky pixels correspondingly.
We use the following criterion as a measure of correct fit of the horizon line which separates the image into the water and sky regions:

\[ J(\Theta, \rho) = (\mu_s - \mu_w)'(\Sigma_s + \Sigma_w)^{-1}(\mu_s - \mu_w) \quad (3.4) \]

The horizon line is parameterized with angle \( \Theta \) and distance \( \rho \) from the origin of the polar coordinate system. \( \mu_s \) and \( \mu_w \) are mean vectors of color-intensity of the sky and water pixel distributions and are defined as

\[ \mu_s = \frac{1}{n_s} \sum_{i=1}^{n_s} x_i^s, \quad \mu_w = \frac{1}{n_w} \sum_{i=1}^{n_w} x_i^w \quad (3.5) \]

\( \Sigma_s \) and \( \Sigma_w \) are the covariance matrices and of the two pixel distributions,

\[ \Sigma_s = \frac{1}{n_s - 1} \sum_{i=1}^{n_s} (x_i^s - \mu_s)(x_i^s - \mu_s)^T \quad (3.6) \]
\[ \Sigma_w = \frac{1}{n_w - 1} \sum_{i=1}^{n_w} (x_i^w - \mu_w)(x_i^w - \mu_w)^T \quad (3.7) \]

where \( n_s \) is the number of sky-pixels, \( n_w \) - number of water pixels.

The next step is the collection of the statistical models that are used to identify situations when only sky or water is in the view. We use the same models as in [39]:

\[ \Sigma_s(t+1) = \alpha \Sigma_s(t) + (1 - \alpha)\Sigma_s \quad (3.8) \]
\[ \Sigma_w(t+1) = \alpha \Sigma_w(t) + (1 - \alpha)\Sigma_w \quad (3.9) \]
\[ \mu_s(t+1) = \alpha \mu_s(t) + (1 - \alpha)\mu_s \quad (3.10) \]
\[ \mu_w(t+1) = \alpha \mu_w(t) + (1 - \alpha)\mu_w \quad (3.11) \]

where \( \Sigma_s(t), \Sigma_w(t), \mu_s(t), \mu_w(t) \) are the running-model covariances and means, when \( \Sigma_s, \Sigma_w, \mu_s, \mu_w \) are covariances and means for the current image. Indices \( t \) and \( t + 1 \)
signify sequential frames of the video. Parameter $\alpha$ controls the speed of change of the running-model. Only those frames that have been identified to contain horizon and follow the accepted model are used for the running-model update.

In order to evaluate the current frame for only-sky or only-water situations the following distance measures are used:

\[
D_1 = (\mu_s - \mu_s(t))'\Sigma_s^{-1}(\mu_s - \mu_s(t)) + \\
(\mu_s - \mu_s(t))'\Sigma_s^{-1}(\mu_s - \mu_s(t)) \\
D_2 = (\mu_s - \mu_w(t))'\Sigma_w^{-1}(\mu_s - \mu_w(t)) + \\
(\mu_s - \mu_w(t))'\Sigma_w^{-1}(\mu_s - \mu_w(t)) \\
D_3 = (\mu_w - \mu_s(t))'\Sigma_s^{-1}(\mu_w - \mu_s(t)) + \\
(\mu_w - \mu_s(t))'\Sigma_s^{-1}(\mu_w - \mu_s(t)) \\
D_4 = (\mu_w - \mu_w(t))'\Sigma_w^{-1}(\mu_w - \mu_w(t)) + \\
(\mu_w - \mu_w(t))'\Sigma_w^{-1}(\mu_w - \mu_w(t))
\]

The description of the distance measures is given in Table 3.1.

Table 3.1. Distance measures for estimation of presence/absence of horizon line in an image.

<table>
<thead>
<tr>
<th>Metric</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>D1</td>
<td>Similarity between the region in the current frame selected as sky and the sky model.</td>
</tr>
<tr>
<td>D2</td>
<td>Similarity between the region selected as sky in the current frame and the water model</td>
</tr>
<tr>
<td>D3</td>
<td>Similarity between the region selected as water in the current frame and the sky model</td>
</tr>
<tr>
<td>D4</td>
<td>Similarity between the region selected as non-sky in the current frame and the non-sky model</td>
</tr>
</tbody>
</table>

Images, where $D_1 < D_2$ and $D_3 > D_4$ are considered to contain the horizon line.
3.3 Image Registration

The image registration step serves two purposes. First, image registration can help detect a situation when no horizon is present in the image, i.e. when the horizon detection fails. Second, if the horizon line is present in the image, then registration of such an image in the coordinate system common for all frames allows simplified tracking of the targets.

The first situation can be exemplified by the following. If the horizon line is not present in the image (for example, the previous horizon detection step failed and only water regions are present), registration of such an image will not be possible because of the dynamic nature of the water surface. Ship detection will not be possible in the image either, since ships may be located on the horizon line only. In the case of only sky regions present, registration is possible. However, correct registration will point to the absence of horizon line, and thus, absence of ships in the image.

The approach to simplified tracking is shown in Figure 3.4. Original video, obtained from a camera installed on a buoy, contains high-magnitude random motion of the field of view (Figures 3.4 (a),(b),(c)). Such motion is erratic in nature and has a big inter-frame distance between the same features in neighboring frames of the video sequence. By registering those frames (see Figure 3.4 (d)) it is possible to use a simple linear Kalman-based tracker to track each target in the global coordinate system.

Use of the horizon line allows us to simplify the registration significantly. Using the horizon line as the axis in the coordinate system common for all frames reduces the search for alignment into one dimension, along the horizon line. Also, since the horizon line is found prior to the registration step, no additional computations are required.
Figure 3.4. Registration of frames in global coordinate system. Even consecutive frames of a video sequence exhibit a high magnitude non-linear intra-frame motion in (a),(b),(c) which is the result of rapid camera motion. By registering frames in a single coordinate system it is possible to track ships with a simple Kalman-based linear tracker. (d) shows registration of (a)-(c) in such a coordinate system. The horizon line simplifies area-based registration of frames by constraining the number of degrees of freedom to one. (e) shows the values of one-dimensional normalized cross-correlation scaled in the range [0..1] during registration of frames (a) and (b) along the horizon line. The peak corresponds to the optimal alignment along the horizon line.

The methods for image registration, as described in Section 3.3, are divided into two major categories: area-based and feature based. In our approach, we assume that all the frames are related to each other through affine transformation due to the motion of camera. Thus, the area based method is sufficient. Moreover, having constrained the search for alignment to one dimension, the area-based method is significantly faster. One-dimensional normalized cross correlation is used as a similarity measure between the registered images:

$$\gamma(x) = \frac{\sum_x [T(x) - \hat{T}][I(x - u) - \hat{I}]}{\sqrt{\sum_x [I(x - u) - \hat{I}]^2 \sum_x [T(x) - \hat{T}]^2}}$$  \hspace{1cm} (3.16)
where $I$ is the base image with the coordinate system fixed on it, $T$ is the template which is being registered against $I$, $\hat{T}$ is the mean of the template, $\hat{I}$ is the mean of $I$ in the region under template. Figure 3.4 (e) shows the values of (3.16) as a function of alignment between images in Figures 3.4(a) and (b). The peak in its value corresponds to the best alignment of the images along the horizon. Only rectangular patches of the edge images which overlay sky regions adjacent to the horizon are used for registration. The height of the rectangular patches is parameterized by $L$. For correlation purposes the images are converted into real-valued gradient images where the intensity of the channel is computed using the magnitude of the color gradient [70]:

$$F(x, y) = \left\{ \frac{1}{2} [(g_{xx} + g_{yy}) + (g_{xx} - g_{yy})] \times \cos 2\theta(x, y) + 2g_{xy} \sin 2\theta(x, y) \right\}^{1/2} \tag{3.17}$$

where $\theta(x, y)$ is the direction of magnitude defined as

$$\theta(x, y) = \frac{1}{2} \tan^{-1} \left[ \frac{2g_{xy}}{g_{xx} - g_{yy}} \right] \tag{3.18}$$

and quantities $g_{xx}$, $g_{yy}$ and $g_{xy}$ are defined as follows:

$$g_{xx} = \left| \frac{\partial R}{\partial x} \right|^2 + \left| \frac{\partial G}{\partial x} \right|^2 + \left| \frac{\partial B}{\partial x} \right|^2 \tag{3.19}$$

$$g_{yy} = \left| \frac{\partial R}{\partial y} \right|^2 + \left| \frac{\partial G}{\partial y} \right|^2 + \left| \frac{\partial B}{\partial y} \right|^2 \tag{3.20}$$

$$g_{xy} = \frac{\partial R}{\partial x} \frac{\partial R}{\partial y} + \frac{\partial G}{\partial x} \frac{\partial G}{\partial y} + \frac{\partial B}{\partial x} \frac{\partial B}{\partial y} \tag{3.21}$$

In the aforementioned quantities, $R$ is the intensity of the red channel, $G$ - green, and $B$ - blue. The magnitude of the color gradient was selected as a method to
emphasize the edges of ships’ silhouettes in a low-contrast maritime environment. However, to reduce computation, the Laplacian filter applied to the grayscale image can also be used. Before taking the gradient, the image strip is processed through the Gaussian filter to smooth effects of compression artifacts. The size of the filter and parameters of the gaussian function are chosen manually to compensate for effects of compression artifacts and to preserve most important features. For our dataset the size of the filter was 9 pixels and the sigma parameter of the gaussian function was chosen to be 0.5.

The first two frames of the video, that passed through the previous blocks of the algorithm, initialize the global coordinate system and create the base image. Later frames are used to grow the base image and to update its pixel values. The pixels of the base image are updated using the following formula:

\[
B_{t+1}(x, y) = \alpha B_t(x, y) + (1 - \alpha) T(x, y)
\]  (3.22)

where \(B_t(x, y)\) and \(B_{t+1}(x, y)\) is the intensity of the pixel at the position \((x, y)\) in the base (consecutive) frames \(t\) and \(t + 1\), \(T(x, y)\) is the intensity of the pixel in the template, and \(\alpha\) is the rate of update. This technique makes the registration adaptive to the changes in the environment.

Threshold value \(T2\) is used to check the value of cross-correlation for significance of registration. Those frames of the video sequence, where the maximum cross-correlation is below the threshold, are not considered further. For example if the horizon line is found incorrectly, the maximum of cross-correlation will not show significant values, and thus the frame will not be further processed. Figure 3.4(e) shows an example where the maximum of the similarity measure between Figure 3.4(a) and
(b) is above the threshold, confirming the correct horizon detection. The value for $T_2$ is learned experimentally on a small dataset of maritime images.

Another important use of the threshold $T_2$ is for those images where the sky regions above the horizon are completely uniform in appearance. Use of the area-based methods is limited only to the regions of the image above the horizon line which provides some non-uniform texture (for example, cloud formations or floating objects). The cross-correlation metric will not show significance for images with uniform sky and without floating objects (including ships) on the horizon, thus, halting further processing of those images.

### 3.4 Segmentation

The segmentation block of the algorithm identifies the regions of the image that potentially contain ships. The input to the block is a real-valued gradient image from the registration step and parameters of the horizon from Section 3.2. The choice for real-valued gradient image as an input for segmentation has several advantages. In original images of RGB-format, ships often share same colors with the sky regions. Thus, it is not practical to directly use color information to threshold the input images in order to segment out the ships. By focusing on the rapid change between colors, i.e. edges, it is possible to outline areas of the image that potentially contain ships. In this case unsupervised segmentation of targets can be done based on the histogram of such real-valued images by choosing a threshold value that separates well-defined peaks. Also, the real-valued gradient image is readily available after the registration step, and thus, no additional computations are required.

A strip of the input image $M$ pixels high that is adjacent to the horizon is used for the segmentation step. The segmentation is performed through global thresholding and results in a binary map. The value for the threshold is chosen by the Otsu’s
method [71], which maximizes the between-class variance of foreground and background in the histogram. The value of the threshold is compared to the minimum value threshold $T_3$, which is learned on a separate dataset of images of clear sky and without ships. This eliminates the false positives originating from compression artifacts and natural gradient attributed to the sky appearance.

After the binary image of the foreground and background is obtained, the result is refined by eliminating spurious foreground objects caused by other factors: sea waves and curvature of the horizon line due to the radial distortion of the camera. Foreground pixels adjacent to the horizon line which result in a projection profile [72] of 20% or more of the length of the horizon are disregarded from consideration.

The next filtering step evaluates the dimensions of the remaining connected objects. The component labeling algorithm [73] is used to find connected foreground
objects and Principal Component Analysis [74] is applied to each of those objects to find their principal axes. The objects that are less than $N$ pixels wide along the second largest eigen-vector are filtered out. This $N$ defines the sensitivity of our approach toward the smallest objects.

All the remaining foreground objects are checked for proximity to the horizon line. The foreground objects that are close to the horizon are outputted as a result of the segmentation step. In the case that the foreground objects are not found, the image is disregarded from further processing. Figure 3.5 shows the sequence of steps in the segmentation algorithm. The values for parameters $M,N$ are described in Chapter 5.

### 3.5 Tracking

The tracking algorithm relies on a Multiple-Hypothesis Tracking framework [75, 76] which allows generation of multiple hypotheses for a track that can be later resolved with future observations. Each hypothesis is managed by the linear Kalman filter [77] that provides a prediction step for possible location of targets in following frames of the video sequence. In our implementation of Reid’s algorithm [75] up to two hypotheses are generated for each observation: existing tracks are projected forward to the time of observations and the nearest two (in terms of Euclidian distance) tracks are assigned to two hypotheses. The depth of such a parallel hypotheses tree-branch is three frames. Such depth is enough to resolve ambiguity of data association between tracks and new observations for our dataset.

Two different filters are defined to track the following two entities: the position of the center of the bounding box (centroid) and the bounding box dimensions. Each of these two filters tracks the behavior of the two variables. Two auxiliary variables for each filter describe the speed of change of the main variables. The tracking
algorithm operates in a global coordinate system for all frames. This coordinate system is anchored to the horizon line.

For the first filter the pair of variables that describe the position of the centroid - namely the $x$ and $y$ coordinates are supplemented by auxiliary variables that describe the speed of movement of this centroid along the $x$ and $y$ axes; horizontal and vertical dimensions of the bounding box $W$ and $H$ are supplemented with the speed of change of these variables in the horizontal and vertical directions.

Equations (3.24)–(3.26) show the implementation of the Kalman filter for visual tracking. The transformation matrix in (3.24) establishes the relation between the main and auxiliary variables in the current and next frames. This relation reflects the linear nature of the motion of the modeled object: the predicted value of the main variable (such as location of the corner) in the next state $t + 1$ is different from the previous state $t$ on the amount of value of the corresponding auxiliary variable $\Delta x_t$: $x_{t+1} = x_t + \Delta x_t$. The measurement matrix in (3.26) shows correspondence between the state vector and measurement vector. Other important variables used in the model are state $w_t$ and measurement $v_t$ noise which are described by a normal distribution.

\[
x_{t+1} = Ax_t + w_t \quad (3.23)
\]

\[
\begin{bmatrix}
x_{t+1} \\
y_{t+1} \\
\Delta x_{t+1} \\
\Delta y_{t+1}
\end{bmatrix}
= 
\begin{bmatrix}
1 & 0 & 1 & 0 \\
0 & 1 & 0 & 1 \\
0 & 0 & 1 & 0 \\
0 & 0 & 0 & 1
\end{bmatrix}
\begin{bmatrix}
x_t \\
y_t \\
\Delta x_t \\
\Delta y_t
\end{bmatrix}
+ w_t \quad (3.24)
\]

\[
y_{t+1} = Hx_t + v_t \quad (3.25)
\]
\[
\begin{bmatrix}
x_{m_t} \\
y_{m_t}
\end{bmatrix}
= 
\begin{bmatrix}
1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0
\end{bmatrix}
\begin{bmatrix}
x_t \\
y_t \\
\Delta x_t \\
\Delta y_t
\end{bmatrix}
+ v_k
\] (3.26)

Here \(x\) and \(y\) represent the main variables such as the position of the center or dimension of the bounding box. \(\Delta x\) and \(\Delta y\) are auxiliary variables.

A track for a new object is initiated if 1) the object was found outside of validation regions of existing tracks (defined by covariance matrices); 2) if it was detected in two consecutive frames and the bounding boxes of such object in these two frames intersect. The values for the state and measurement vectors for that track are initiated from these two frames. The track for an object is terminated if the number of valid detections in the history of the track is less than half of the number of consecutive frames for which the track exists. Objects with a track length of more than 10 contiguous frames are considered marine vehicles and are provided as the system output. Figure 3.6 shows the flowchart of the tracking algorithm.
Figure 3.6. Outline of the tracking algorithm. The algorithm compares the detections in the current video frame with the history of previous detections. The existing tracks are updated or deleted depending on how the detections from the current frame support the history.
The purpose of this section is to describe the methods of evaluation for the developed algorithm and its components. Several performance metrics are used to evaluate horizon detection performance. For detection of ships on a frame and sequence levels we use several metrics that account for many important measures of system performance such as the number of objects detected and missed, false positives, fragmentation in a temporal dimension, and localization error of detected objects. Three evaluation thresholds $ET_1 - ET_3$ are introduced. Those thresholds are intended for performance evaluation only and are not part of the ship detection algorithm like thresholds $T_1 - T_3$. The evaluation is conducted experimentally on a number of datasets which are described in Chapter 5.

### 4.1 Performance of Horizon Presence Detection Algorithm

This set of metrics is intended to measure the precision and recall of horizon status detection in images. When the horizon is present in an image, the image is labeled as ‘horizon-image’. When the horizon is not present or when it is present but is obscured, blurred, or uncertain, the image is labeled as ‘no-horizon-image’. The following are the metrics used to evaluate detection of horizon presence:

$$A_1 = \frac{tp}{tp + fp} \quad A_2 = \frac{tp}{tp + fn}$$

(4.1)
where $A_1$ is the precision of detection, $A_2$ is the recall, $tp,fp,fn$ are true positive, false positive, and false negative outcomes of the classification correspondingly.

### 4.2 Performance of Horizon Localization Algorithm

Localization of the horizon is the process of determining the parameters of the horizon line in those images where the horizon is present. In order to evaluate such performance the algorithm’s output is compared with the ground-truth using two performance metrics to reflect different aspects of performance. Both the metrics use pixelwise comparison of the candidate and target data. The first metric represents the percentage of pixels in all images correctly separated into water and sky regions by the found horizon line:

$$A_3 = \frac{1}{k} \sum_{i=1}^{k} \frac{N_i^c}{N_i}$$

where $k$ is the number of images in the dataset, $N_i^c$ - number of pixels correctly separated by the found horizon line in image $i$, $N_i$ - number of pixels in the image $i$. This metric provides a reference to the general performance of the horizon localization algorithm on a dataset of images.

The second metric reflects performance of horizon localization on each of the images of the dataset. The horizon line in an image is considered detected correctly if the percentage of pixels in the image correctly separated by the horizon line is above the specified evaluation threshold. The percentage of such images in the dataset will define the second accuracy metric:

$$A_4 = \frac{n_c}{k}$$
where $k$ is the total number of images in the dataset and $n_c$ is the number of images in the dataset where the horizon line was detected with accuracy above the given threshold:

$$\frac{N^i_c}{N^i} \geq ET_1$$  \hspace{1cm} (4.4)

where $ET_1$ is the evaluation threshold value.

### 4.3 Performance of Ship Detection and Localization

The performance of ship detection is analyzed in two stages. In the first stage, a frame-wise localization and one-to-one matching is done between the ground-truth targets and detected objects. A decision for each ground-truth target in every frame is made: detected or missed. The false positives are also counted. In the second stage of evaluation a decision is made on a sequence level: the target is detected in the sequence, missed, or a false alarm occurred.

The spatial localization of ground-truth targets and detected objects is performed using rectangular bounding boxes. For simplicity of representation, a bounding box around targets and detected objects is always oriented so that its sides are parallel to the axes of the image plane. To facilitate creation of the ground-truth, the ViPER tool [78] is used to create bounding boxes around targets in the data. The spatial overlap between matched ground-truth targets and detected objects is computed according to several metrics.

The precision and recall of localization for an object in a frame is computed according to the following relations:

$$\text{PrecisionLoc} = \frac{G \cap D}{G} \quad \text{RecallLoc} = \frac{G \cap D}{D}$$  \hspace{1cm} (4.5)
where $G$ is a area of the bounding box under the ground-truth target, $D$ - area of the bounding box under the detected object matched to its ground-truth, $G \cap D$ - spatial overlap between the matched ground-truth target and detected object.

The dice coefficient [78] is computed as follows:

$$
Dice = \frac{2(G \cap D)}{G + D}
$$

The dice metric avoids the asymmetry of recall and precision for localization, and thus, is better suited for detection on higher levels of analysis. The recall and precision of localization are only reported on the frame-level. All three metrics are naturally normalized between 0 and 1.

In order to decide if the object is detected in the frame, missed, or a false alarm occurred, the localization according to the dice overlap (4.6) for each ground-truth target in the frame is compared to a pre-defined frame-level evaluation threshold $ET2$. The target is declared detected in the frame if the dice overlap is above the specified threshold. If the overlap is less than the threshold, the target is declared missed for the frame. If the target does not exist for the frame (ground truth for an object is absent), but detection exists, such detection is declared a false detection. This way the first stage of performance evaluation reports the counts of detections, misses, and false positives in each frame.

The second stage of evaluation analyzes detection of ships on the sequence level. From the first stage of evaluation, a particular target may be detected in its sequence for a certain number of frames. If the set of frames where the target is detected overlaps the appropriate number of frames where the ground-truth for it exists, the target is declared detected in the sequence. Otherwise, the target is declared missed. The overlap on the sequence level is computed according to the temporal
dice coefficient, which is compared to the sequence-level evaluation threshold $ET_3$.

$$\text{TemporalDice} = \frac{2(\mathcal{F}_G \cap \mathcal{F}_D)}{\mathcal{F}_G + \mathcal{F}_D}$$  \hspace{1cm} (4.7)$$

where $\mathcal{F}_G$ is the set of frames where the ground-truth for the target exists, $\mathcal{F}_D$ - set of frames where the object is detected, $\mathcal{F}_G \cap \mathcal{F}_D$ - temporal overlap between the detected object and its groundtruth in the sequence, i.e. the set of frames where the detected object and its groundtruth exists and they are spatially matched in the first stage of analysis.

By using the temporal dice coefficient the counts of targets detected, targets missed, and false positives are found for each sequence. The precision of detection and recall of detection are, then, calculated for all targets in all sequences. The precision and recall of detection in all the sequences is defined as follows:

$$\text{Precision}_{\text{Seq}} = \frac{C_D}{C_G} \hspace{1cm} \text{Recall}_{\text{Seq}} = \frac{C_D}{C_D + C_F}$$  \hspace{1cm} (4.8)$$

where $C_D$ is the count of objects in all sequences that are correctly detected, $C_G$ - count of all ground-truth objects, $C_F$ - count of all false positives.

These two metrics are used to report results of detection of all targets across all the sequences provided evaluation thresholds $ET_2$ and $ET_3$. 

52
CHAPTER 5
PERFORMANCE OF SHIP DETECTION ALGORITHM

A large dataset of video sequences was collected from a prototype of the system described in [13]. The system was tested during six deployments in several locations off the Florida coastline. The deployment locations included the open ocean, as well as in littoral areas when the camera was pointing away from the shore. Some of those deployment locations, such as John’s pass waterway, were busy maritime traffic points. The video was taken in various weather conditions which included clear sky, partially cloudy sky, and overcast during day-time. The video, which was recorded in a resolution of 800x600 with 10 frames per second, had significant compression artifacts due to camera’s internal compression of data in MJPEG format.

A subset of the video data, called *Performance Evaluation dataset*, was annotated for performance evaluation. This dataset consisted of 550 video sequences selected randomly from all available data. Each sequence consisted of 100 frames representing a contiguous 10-second interval. Thus, the evaluation dataset included approximately 55,000 images. Instances of ships within a video sequence included single and multiple vessels (see Figure 5.1), some entering and leaving the field of view, as well as periods with no ships in the view. All 550 sequences were annotated with ground-truth ships by using ViPER tool [78]. The other datasets used in the training and evaluation of the algorithm are as follows:

- *Development dataset* of 150 images divided into three subsets: (a) A subset to learn the value of the threshold parameter $T_1$ for horizon anomaly detec-
The subset consisted of 50 images outside of the data collected from the prototype. The images were split into two categories: ‘horizon-present’, ‘horizon-not-present’. (b) A subset to learn the value of the threshold parameter $T_2$ for registration validation. The subset consisted of 50 images of the dataset from the prototype split into two categories: ‘clear-sky’ and ‘textured-sky’ to mark the images where the area above the horizon was uniform (no clouds, no floating objects) and images with non-uniform sky. (c) A subset to learn the value of the threshold parameter $T_3$ for segmentation validation. The subset consisted of 50 images split into two categories: ‘clear-horizon’, ‘ships-horizon’ to mark the images where the horizon did not have any objects of interest and the images having ships on the horizon.

- *Horizon Presence Evaluation dataset*. The dataset was created to evaluate performance of horizon presence detection and consisted of approximately 10% of the images of the annotated dataset. Those images where the horizon was
not present were marked as ‘no-horizon’. All other images in the dataset were considered ‘horizon’-images. The majority of the images in the dataset contained horizon.

- *Horizon Localization Evaluation dataset.* The dataset was created to evaluate the accuracy of horizon line localization and consisted of 150 images with annotated water and sky regions.

Table 5.1 summarizes the datasets and Table 5.2 shows the values of the non-threshold parameters used in different steps of the algorithm.

<table>
<thead>
<tr>
<th>Description</th>
<th># Images</th>
</tr>
</thead>
<tbody>
<tr>
<td>Performance Evaluation dataset</td>
<td>55000</td>
</tr>
<tr>
<td>Development dataset</td>
<td>150</td>
</tr>
<tr>
<td>Horizon Presence Evaluation dataset</td>
<td>5400</td>
</tr>
<tr>
<td>Horizon Localization Evaluation dataset</td>
<td>150</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>K</td>
<td>Number of candidate lines for horizon consideration</td>
<td>5</td>
</tr>
<tr>
<td>L</td>
<td>Height of rectangular patches used for image registration, pixels</td>
<td>75</td>
</tr>
<tr>
<td>M</td>
<td>Height of rectangular patches over horizon used for segmentation, pixels</td>
<td>40</td>
</tr>
<tr>
<td>N</td>
<td>Minimum width of the object along the second largest eigenvector, pixels</td>
<td>6</td>
</tr>
<tr>
<td>α</td>
<td>Rate of update for registration map</td>
<td>0.05</td>
</tr>
</tbody>
</table>

5.1 Performance of Horizon Presence Detection Algorithm

Approximately 95% of the images in the *Horizon Presence Evaluation dataset* contained horizon and belonged to the category ‘horizon’ images. The ‘no-horizon’
category was represented by 5% of the images, where 4% of the images contained water regions and 1% of the images contained only sky regions. Table 5.3 shows the results of classification of the dataset into those two categories.

Table 5.3. Results of evaluation of horizon presence detection algorithm.

<table>
<thead>
<tr>
<th></th>
<th>Actual ‘Horizon’ images</th>
<th>Actual ‘No-horizon’ images</th>
</tr>
</thead>
<tbody>
<tr>
<td>Classified as ‘Horizon’</td>
<td>5148</td>
<td>98</td>
</tr>
<tr>
<td>Classified as ‘No-horizon’</td>
<td>19</td>
<td>135</td>
</tr>
</tbody>
</table>

Accuracy and recall for the ‘horizon’ category is achieved at 98.1% and 99.6% correspondingly. For the ‘non-horizon’ category those figures are at 88.0% and 57.9%. The algorithm is biased toward the ‘non-horizon’ category, but since we are more interested in those images which potentially have ships, this performance is acceptable.

5.2 Performance of Horizon Localization

The accuracy of horizon localization on the Horizon Localization Evaluation dataset according to the $A_3$ metric (4.2) was 99.26%. Figure 5.2 shows the value of accuracy for horizon localization according to metric $A_4$ (4.3) for a number of evaluation threshold values $ET_1$. Overall, the accuracy of localization is high for both metrics. The metric (4.3) sees degradation of performance with the threshold of 98% and above. However, such threshold is very close to 100% and the human error during the process of creating ground-truth may be the primary factor resulting in such performance.
5.3 Performance of Ship Detection

5.3.1 Localization on Frame Level

The performance of localization on a frame level is computed according to metrics (4.5) by averaging the results from each frame in all sequences and is a general metric for qualitative overlap between the ground truth and found targets, not the fraction of ships detected. The precision of localization is 76% with lower score for recall - 61%. According to our observations, such recall performance is observed because of the high number of false positives that occur due to artifacts in MJPEG data, especially small objects with height around 6 pixels that occur during rapid erratic motion of the camera attached to the buoy. The next section is focused on the fraction of ships detected.
5.3.2 Detection on Sequence Level

The ship detection precision and recall of the algorithm on a sequence level can be evaluated by three parameters. The first parameter, the evaluation threshold $ET_2$, corresponds to the detection of a ship on a frame level depending on the value of overlap between the ground truth and found objects. Once a hard decision (i.e. detected or not) is made in each each frame regarding each target it is possible to evaluate detection on a sequence level. The detection according to the second evaluation parameter, the threshold $ET_3$, produces a hard decision for detection of a target in a sequence. The third evaluation parameter, minimum height of the object, is needed to evaluate performance according to the size of the acceptable object. Thus, the measure of performance may vary greatly depending on the combination of evaluation parameters, although the performance of the algorithm itself does not depend on the evaluation parameters. In this work we show performance only for a subset of those evaluation combinations.

Table 5.4 and Figure 5.3 shows performance of detection for all targets on all the sequences according to the precision and recall metrics (4.8) with different values of the temporal threshold $ET_3$ with the localization threshold $ET_2$ fixed at the level of 0.5 and with the minimum height set to 6. The value 0.5 is the most natural choice for a symmetric metric such as dice. The algorithm performed well even for values of the temporal threshold $ET_3$ up to 0.7. Up to 88% of ships were detected in all sequences when at least a small temporal overlap between ground truth and the system’s output is counted. According to our observations, misses and false positives usually occur due to small objects, with object-heights close to the minimum height of 6 pixels. In our dataset we have a substantial number of ships with sizes in the vertical dimension close to that minimum. The detection recall for objects with heights larger than 6 pixels is higher in those frames where the horizon is detected.
correctly. For example, Table 5.5 and Figure 5.4 show the dynamics of performance when evaluation threshold values for \( ET_2 \) and \( ET_3 \) are fixed at 0.5 and when the ground-truth and system output are filtered by the minimum height of the objects.

5.4 Discussion

Overall, the detection scheme proved to be robust with reliable horizon detection, registration, and tracking of targets. The visual results obtained on the data from the prototype outside the annotated 55,000 images showed good performance as well (subjectively to the human evaluator). It is important to note that the data in the collected dataset was 'difficult' not only because of the non-stationary camera, but because of the abundance of compression artifacts due to the low-quality imaging sensor present in the first prototype of the system. It is expected that for the imagery not affected by the mentioned artifacts the performance of the system will be better, especially in relation to the false positives.

Table 5.4. Value of precision and recall of detection on the sequence level according to various values of the evaluation threshold \( ET_3 \). Threshold \( ET_2 \) fixed at 0.5.

<table>
<thead>
<tr>
<th>Threshold ET3</th>
<th>Precision, %</th>
<th>Recall, %</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.1</td>
<td>88.63</td>
<td>72.76</td>
</tr>
<tr>
<td>0.2</td>
<td>83.89</td>
<td>68.87</td>
</tr>
<tr>
<td>0.3</td>
<td>82.46</td>
<td>67.70</td>
</tr>
<tr>
<td>0.4</td>
<td>80.57</td>
<td>66.15</td>
</tr>
<tr>
<td>0.5</td>
<td>75.83</td>
<td>62.26</td>
</tr>
<tr>
<td>0.6</td>
<td>72.04</td>
<td>59.14</td>
</tr>
<tr>
<td>0.7</td>
<td>66.82</td>
<td>54.86</td>
</tr>
<tr>
<td>0.8</td>
<td>49.29</td>
<td>40.47</td>
</tr>
<tr>
<td>0.9</td>
<td>22.27</td>
<td>18.29</td>
</tr>
<tr>
<td>1.0</td>
<td>2.84</td>
<td>2.33</td>
</tr>
</tbody>
</table>
Figure 5.3. Precision and recall of detection on the sequence level according to various values of threshold ET2. Threshold ET3 is fixed at 0.5.

Table 5.5. Value of precision and recall of detection on the sequence level according to various values of minimal height of objects. Evaluation thresholds ET2 and ET3 are fixed at 0.5

<table>
<thead>
<tr>
<th>Minimum Height</th>
<th>Precision, %</th>
<th>Recall, %</th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
<td>75.83</td>
<td>62.26</td>
</tr>
<tr>
<td>7</td>
<td>75.83</td>
<td>62.50</td>
</tr>
<tr>
<td>8</td>
<td>75.83</td>
<td>65.57</td>
</tr>
<tr>
<td>9</td>
<td>75.36</td>
<td>68.24</td>
</tr>
<tr>
<td>10</td>
<td>75.36</td>
<td>70.98</td>
</tr>
<tr>
<td>11</td>
<td>75.36</td>
<td>72.27</td>
</tr>
<tr>
<td>12</td>
<td>70.73</td>
<td>71.82</td>
</tr>
</tbody>
</table>
Figure 5.4. Precision and recall of detection on the sequence level according to various minimum heights of objects. Evaluation thresholds ET2 and ET3 are fixed at 0.5.
CHAPTER 6
DETECTION OF THE VANISHING LINE OF THE OCEAN SURFACE THROUGH TEXTURE ANALYSIS

In Chapter 3 a robust horizon detection algorithm was presented. The algorithm relies on assumption of the existence of a line that separates two regions relatively homogeneous in color. In case of the absence of the horizon a statistical model is used in order to identify the situation when only one region is present. Such a statistical model provides only a boolean answer - the horizon is present or not. In this chapter, a new algorithm is presented to estimate the vanishing line of a surface plane, which can be used for the purpose of detecting the horizon line and as well as detection of the situation of horizon presence or absence. Although not integrated in the main algorithm for ship detection, because of the current accuracy of estimation, this algorithm for estimation of plane orientation has the potential for future use for horizon detection in complex situations. For example, a possible area of use of such a horizon detection scheme is for imagery where the coastline is present in the image. Development of such an algorithm also adds a theoretical contribution to the topic of Shape From Texture (SFT), which has a broad use in computer vision applications.

In the developed method the horizon line is estimated indirectly. The texture of the ocean surface is analyzed under the assumption of the presence of perspective distortion and vanishing line is found as a line where the textured surface converges in the image plane. The novelty of the approach is that it estimates the vanishing line of a plane with very difficult stochastic texture from a single image taken by an uncalibrated perspective camera where other similar methods perform poorly.
6.1 Overview of Vanishing Line Detection in Texture Images

Traditionally, methods of a plane’s orientation estimation are divided into two main categories: spectral [79, 80] and structural [81, 82] (although, the combinations of the two are well represented as well [83, 84]). The spectral category of methods aims to estimate the orientation of a plane by finding the gradient of perspective distortion. Different spectral measurements are used to qualitatively estimate the gradient, making certain assumptions about texture, such as homogeneity [48] or isotropy [85]. The methods from the structural category rely on explicit identification of texture elements (textels) which then are used to find the geometric solution for the vanishing line.

In this work, we choose the second path to find the orientation of a plane by identifying the vanishing line directly through elements constituting texture. However, in order to avoid a very fragile process of segmentation of textels the appearance of which depend on marine environment, we rely on scale-invariant keypoints [86]. First, the vanishing points in the image are estimated by finding pairs of similarly looking scale-invariant keypoints that are different in scale. The vanishing line of the plane is then estimated from a ensemble of such vanishing points selected that satisfy several geometrical constraints.

Unlike the traditional structural methods of vanishing line detection, the keypoints are only similar looking within pairs, thus, each pair of keypoints may actually represent very different features of the texture. For example, in the image of the water surface such keypoints may be top of the ridges of sea waves, or hollow-looking depressions in the water. Thus, the homogeneity of appearance of structural elements is not required. Only consensus of, possibly heterogeneous, pairs regarding the orientation of the plane is necessary. In order to select the keypoints comprising a pair we chose the Scale-Invariant Feature Transform (SIFT) [54] method, described
Figure 6.1. Example of detecting vanishing line by matching SIFT keypoints. Two pairs of SIFT keypoints are matched by appearance similarity. The first pair is shown in red and the second pair is shown in green. Based on the difference in size of the neighborhood of their descriptors, caused the scale change, it is possible to estimate a vanishing point. Several vanishing points are used to estimate the vanishing line of the textured surface.

in Section 2.4, which allows detection of highly distinctive interest keypoints that are partially invariant to affine transformation.

The method works in several steps. First, the SIFT algorithm is used to detect candidate interest points within the image that are highly distinctive. Opposite to their original purpose of finding correspondences in pairs of images, these keypoints are, then, matched to other keypoints of different scales within the same image. Each potential pair consists of points that are similarly looking and are, necessarily, belonging to different scales. Thus, we try to find similar features in the image (for example peaks of waves) that are located at different distances from the camera and, due to the foreshortening effect of perspective projection, are of different size in the image. We use the scales and positions of those keypoints in the pair to triangulate the position of an individual vanishing point in the image. Having identified a number
of such vanishing points the position of the vanishing line, and thus, affine geometry of the plane is estimated (see Figure 6.1).

The appeal of such method is its potential to estimate the plane’s orientation in real images with very difficult stochastic textures in a simple geometric way. The method does not require prior knowledge of appearance of individual similar textels that comprise an image, avoiding the segmentation step from traditional structural methods.

6.2 Vanishing Line Estimation

In the developed algorithm the scene is modeled as a plane coated with stochastic texture and is viewed with a uncalibrated perspective camera. Since the parameters of the camera are unknown, it is not possible to recover full perspective geometry. However, Criminisi and Zisserman have shown in [48] that if the vanishing line is detected in the image, the affine geometry of the scene can be established.

The basic idea to identify the vanishing line comes from the properties of intersection of two sets of imaged parallel lines [87]. Each set of parallel lines intersects at the infinity point, which under perspective projection is a vanishing point on the vanishing line. In classical approaches of plane orientation estimation, special structural elements (textels) are selected with the assumption that their size is uniform and the set of lines can be drawn along their sides. We adopt a similar approach for vanishing point triangulation, however, instead of selecting textels we select pairs of scale-invariant keypoints. Such keypoints provide very intuitive locations, scale and orientations of features which can be matched with high probability with other keypoints. Different pairs of matched keypoints may come from different features of underlying texture, however, they should point to the same vanishing line.
6.2.1 Assumptions and Constraints

For geometrical computations, we make an assumption, that each keypoint in the pair is of equal size under orthographic projection. Thus, under perspective projection, the position and sizes of two similar keypoints in the image can be used for computing a single vanishing point on a vanishing line. The size of matched keypoints, in our algorithm, is proportional to the scale where the keypoints are detected.

Two algorithms, SURF [88] and SIFT (described in Section 2.4), have been tested on a number of images of ocean surface. However, the results in this work are only reported for the SIFT algorithm. The number of scales in a scale pyramid is increased from the default value to minimize the effect of scale discontinuity on the accuracy of vanishing point estimation.

SIFT keypoints are only partially invariant to affine transformations [54] and the percentage of correctly matched points drops when significant affine transformation occurs. Thus, one may argue, that it is impossible to compare keypoints of the same image features because they would be distorted by projective transformation. However, if keypoints in the matched pair are showing high similarity they are detected at those locations of the image that are not distorted too much, because otherwise the match would not have taken place.

Individual vanishing points computed from pairs of matched keypoints provide an imprecise location of true points on a line at infinity because of instability of appearance of stochastic texture. However, a substantial number of such vanishing points provides meaningful information about the vanishing line orientation. Some pairs of matched keypoints may not be pointing to the vanishing line at all, because the stochastic texture does provide the conditions for such pairs of keypoints to exist
(see Figure 6.2). However, those outliers can be removed from consideration using several geometrical constraints.

Figure 6.2. Example of supporting vectors and vanishing points generated by pairs of matched SIFT keypoints. Each supporting vector originates at one of the keypoints in the pair, passes through the other keypoint, and ends in the vanishing point (points 1-9). The green line shows the vanishing line hypothesis. Vanishing points 4 and 7 point away from the line and, thus, cannot support such a hypothesis. Points 1 and 9 make a very acute angle with the line, and are disregarded from consideration due to a higher chance of error. Points 2, 3, 5, 6, and 8 satisfy geometrical constraints and support the vanishing line hypothesis.

The following terms are used to describe triangulation of a vanishing point. In a pair of matched keypoints the keypoint with the smaller scale, and thus, with the bigger radius of its descriptor’s region in the original image is called the originating point in the pair. The other point in the pair with the larger scale and smaller radius of its descriptor’s region is called the middle point. The vanishing point generated by the pair of keypoints is located on a ray that begins in the originating point and
goes through the middle point. The vector that starts at the originating point, goes through the middle point and ends at the vanishing point is called a supporting vector for a vanishing line. The term reflects the idea of a feature in the image that supports the hypothesis of a specific vanishing line and is not related to support vector from the Support Vector Machine learning algorithm.

The constraints listed below are geometric rules applied to the terms mentioned above in order to produce a good vanishing line estimate:

1. A vanishing line of the plane can only be estimated if the matched keypoints in the image come from the texture of a surface plane. Thus, those keypoints that come from features of the image that do not belong to the textured plane should not be considered.

2. Keypoints in a pair should be of different scale. Difference in scale and the position of keypoints allows triangulation of a vanishing point.

3. Keypoints in a pair should be located far enough from each other that the regions under their descriptors do not intersect. In practice, since we are relying on perspective projection that changes the sizes of the same features in the image depending on the distance to the camera, the distance between the keypoints in the image should justify the scale change.

4. The matched keypoints that create a supporting vector with an acute angle to the vanishing line below the minimal threshold need to be disregarded for vanishing line computation $A > T_A$. The reason behind it is that when the supporting vector is at a very acute angle to the vanishing line, both keypoints from which the vanishing point is computed are located at approximately same distance from the vanishing point, because the gradient of perspective distortion is always perpendicular to the vanishing line. Thus, the scale change
caused by perspective distortion is small and susceptible to error caused by
noise.

5. The supporting vectors cannot point away from the estimated vanishing line. If for a particular estimated vanishing line the supporting vector points away, the vanishing line (as shown by the example in Figure 6.2), the vanishing point of such supporting vector should be considered an outlier.

The algorithm for estimation of vanishing line works in two steps. First the keypoints are found in the image, were matched to other keypoints in the image to generate vanishing points, and the orientation of the vanishing line is estimated. Second the selected orientation of the vanishing line is used in a search for a position of the line that satisfies the heuristic constraints and maximizes the optimization criterion.

6.2.2 Estimating Direction of Vanishing Line

The keypoints are selected by directly applying the SIFT algorithm to the raw image data. In order to increase the accuracy of vanishing point localization, the number of scales in the scale pyramid is increased as compared to default values proposed in the description of SIFT algorithm [54]. The only indirect parameter during the SIFT keypoint localization used in our algorithm is the number of keypoints per image. This number drives the selection of other parameters for SIFT, such as the keypoint response threshold.

Once the SIFT keypoints are identified in the image, their corresponding SIFT descriptors are computed (see Chapter 2 for details) and matched to the descriptors of other keypoints in that image. Since there are \( n^2 \) matches in the image with \( n \) keypoints, and most of them are of no interest, an effective constraint on the distance in the similarity space needs to be imposed to reduce the number of matches. In
this work, we used a k-nearest neighbor [89] in appearance space to select a limited number of closest neighbor keypoints. The choice of selection of such a constraint as opposed to the radial distance in similarity space is based on the fact that some keypoints are very distinctive, and when unrestricted in number of neighbors will dominate in the set of matches, skewing results for vanishing line.

The matches selected from the previous step are checked against the constraints (1)-(3) listed in Section 6.2.1. Those matches that are not consistent with the assumptions are ruled out from further consideration. The remaining matches are sorted by the similarity measure and the top $l$ percent of the matches is used to generate vanishing points.

In order to generate a vanishing point from a pair of matched keypoints the following assumption is used. The lines parallel in the world would intersect under perspective projection in a vanishing point. Assuming the sizes of matched keypoints in the real texture are the same the lines drawn along their sides are parallel. The intersection of those lines in an image is the vanishing point. Figure 6.3 shows the geometrical interpretation of triangulation of a vanishing point. Let the distance between the matched keypoints be $D_1$. The radiuses for areas under the SIFT descriptors are proportional to the scales where the matched keypoints were found. If $S_1$ is the scale for the first keypoint with the bigger radius for SIFT descriptor, and $S_2$ is the scale for the second keypoint with the smaller radius, the relation $D_2$ for the distance from the second keypoint to the vanishing point is the following:

$$D_2 = \frac{D_1 S_2}{S_1 - S_2}$$  \hspace{1cm} (6.1)

Using (6.1) the positions of all vanishing points are computed (see Figure 6.4). Having an ensemble of vanishing points it is possible to estimate a line that best
models the vanishing points using the sum of least squares as an optimization function. The line is found using linear square regression. However, since the position of a vanishing point is subject to error in both dimensions orthogonal (or total) linear least square fit, described in Section 2.2, is more appropriate than the regular linear square regression. The fit of the vanishing line to the points is done twice. First, the fit is performed on all vanishing points found in the previous step. After such initial computation, a certain percentage of the vanishing points that have the biggest distance from the computed line are removed as outliers and the line parameters are recomputed on the remaining points. The found line is represented in a slope-intercept form $y = mx + b$, where $m$ is the slope and $b$ is the intercept.

The computed line can be used as an estimate of vanishing line directly. However, despite the orientation of such line is close enough to the orientation of the real vanishing line in the image, it was found that the positional accuracy can be improved.
Figure 6.4. Estimating direction of the vanishing line from a ensemble of vanishing points. The green line shows the ground truth. The red line shows the line estimated from the ensemble of vanishing points (red dots in the image) using total least squares algorithm. The vanishing points that are located far away from the estimated line are outliers.

6.2.3 Estimating Position of Vanishing Line

As mentioned in Section 6.2.1, the nature of stochastic texture provides conditions when matched SIFT features in an image look very similar, however, do not correspond to the features of texture what would be of the same size when they are imaged orthographically. For example, Figure 6.2 shows some wave-features close to the camera that are smaller in real size than similarly looking waves away from the camera (points 4 and 7). Thus, the most important assumption for the vanishing line estimation does not hold. Although the number of such matches is small compared to the number of correct matches, it is still important to filter them out from consideration in order to improve accuracy of the vanishing line estimation. To
remove such outliers the heuristics (4)-(5) from the list in Section 6.2.1 are applied as described below.

For each supporting vector the angle between between it and the vector of gradient of perspective distortion is extracted and compared with the allowed threshold. The vanishing points of those supporting vectors that make an angle of more than $T_a$ degrees from the gradient are removed from consideration for vanishing line position determination. In order to obtain the vector of perspective gradient the following procedure is applied. First, since the vector of perspective gradient is perpendicular to the vanishing line (as shown in [48]) the slope of such perpendicular is found as $m_\perp = \frac{1}{m}$, where $m$ is the slope of the vanishing line. Second, to convert the slope into a vector all the supporting vectors are projected onto the slope of the perpendicular and the projections are summed. The sign of such sum will indicate the direction along the slope as the direction of the gradient vector.

The orientation of the vanishing line found in Section 6.2.2 is used in the search for the second parameter of the line - its position. The implementation of such search is done as following. The vanishing line in slope-intercept form is converted into normal representation of a line:

$$\rho = x \sin \Theta + y \cos \Theta \quad (6.2)$$

where $\rho$ is the distance from origin of coordinate system to the estimated line and $\Theta$ is the angle between the normal to the line and x-axis. During the search, the $\Theta$ parameter is fixed, and the value of $\rho$ is varied.

A special criterion is used in order to find the optimal $\rho$. The criterion minimizes the sum of absolute distances from the estimated line to the set of vanishing points.
from supporting vectors that make an angle of less than $T_a$ degrees:

$$\rho = \arg_{\rho} \min_i \sum D(L_{\Theta, \rho}, V_i) \quad (6.3)$$

where $L_{\Theta, \rho}$ is the vanishing line in a normal form parameterized by $\Theta$ and $\rho$, $V_i$ is the vanishing point $i$, supporting vector of which makes an angle with the perspective gradient less than $T_a$ degrees.

6.3 Performance Evaluation

In order to evaluate the performance of the developed algorithm a special evaluation metric needs to be established and an evaluation dataset needs to be defined. Although the output of vanishing line estimation is similar to the output of horizon detection algorithm it was decided to have different metrics than those used for horizon detection in Chapter 4. The main reason for that is that the pixel-based measure used in Chapter 4 is only meaningful for those algorithms which provide highly accurate results. This is not the case for SFT methods in general, as SFT methods do not provide high accuracy for horizon detection because are based on indirect measurements. Thus, we are not comparing the accuracy of methods for horizon detection and vanishing line detection and different metrics for performance evaluation are used for these two algorithms. The performance for the vanishing line detection is evaluated in units directly related to the line parameters. The chosen metric is composed of average error in the $\Theta$ angle and average error in normalized distance from the origin $\rho_N$. The normalized distance $\rho_N$ is obtained for each image by dividing $\rho$ of the evaluated line by the length of diagonal of the image. Such normalization allows comparison of errors in images of different sizes.

A special dataset of real images of ocean surface was collected using Google Images search engine. Each image contained a scene of ocean surface with significant
perspective effect present. Each image in the dataset was assigned the groundtruth and preprocessed according to the following rules. First, each image forms approximately a square in its dimensions so there is no bias in orientation due to the aspect ratio of the image. Second, each image was mostly showing ocean surface to minimize the chance that the keypoints come from the features of the image not related to the texture. The dataset of original 83 images was split into the development set and the test set. The development set included 20 images used to tune the parameters of the algorithm. The test set included 53 images used to evaluate the performance.

The Table 6.1 shows the values of parameters in the algorithm obtained experimentally in order to maximize performance on the development set.

Table 6.1. Values of parameters in the algorithm for vanishing line detection.

<table>
<thead>
<tr>
<th>Parameter name and description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of layers in scale pyramid for SIFT</td>
<td>6</td>
</tr>
<tr>
<td>Number of octaves in each layer for SIFT</td>
<td>6</td>
</tr>
<tr>
<td>Number of SIFT keypoints</td>
<td>500</td>
</tr>
<tr>
<td>Length of SIFT descriptor vector</td>
<td>128</td>
</tr>
<tr>
<td>Number of nearest neighbors for each keypoints to consider when matching</td>
<td>10</td>
</tr>
<tr>
<td>Proportion of scales in a matched pair of keypoints to consider for supporting vector</td>
<td>1.5</td>
</tr>
<tr>
<td>Top percentage of matches by similarity distance to consider for vanishing line orientation</td>
<td>30%</td>
</tr>
<tr>
<td>Percentage of outliers in vanishing line orientation estimation</td>
<td>20%</td>
</tr>
<tr>
<td>Maximum angle deviation from gradient vector for supporting vector in vanishing line position estimation, $T_o$</td>
<td>45 degrees</td>
</tr>
</tbody>
</table>

The performance of the algorithm with parameters from Table 6.1 was evaluated on the independent test set of 53 images. Table 6.2 shows the results of performance for two setups. In the first setup, the vanishing line orientation and position are estimated in one step as described in Section 6.2.2 (thus, without correction of the position of the line). In the second setup, two step approach is applied: first, estimation of the orientation of the line as in Section 6.2.3, and then, correction of
the position as in Section 6.2.3. Figure 6.5 shows some examples of images with identified vanishing line.

Table 6.2. Performance of vanishing line detection according to two setups.

<table>
<thead>
<tr>
<th></th>
<th>First Setup - Estimation of orientation and position of vanishing line in one step</th>
<th>Second Setup - Estimation of orientation and position of vanishing line in two steps</th>
</tr>
</thead>
<tbody>
<tr>
<td>Error $\Theta$, degrees</td>
<td>7.84</td>
<td>7.84</td>
</tr>
<tr>
<td>Standard deviation, Error $\Theta$, degrees</td>
<td>10.20</td>
<td>10.20</td>
</tr>
<tr>
<td>Error $\rho_N$</td>
<td>0.23</td>
<td>0.18</td>
</tr>
<tr>
<td>Standard deviation, Error $\rho_N$</td>
<td>0.19</td>
<td>0.15</td>
</tr>
</tbody>
</table>

According to our experiments, the accuracy in the line position estimation was not high. However, the images of the real stochastic texture are difficult for SFT methods in general. With structural methods, for example, it is impossible to define a textural element for such textures, which makes them inappropriate for the task. With spectral methods it is hard to define the scale which can be used to compare neighborhoods in the image to extract meaningful changes in the perspective gradient. Thus, the developed method represents a novel approach capable to work with the very difficult data.

6.4 Discussion

This chapter describes direct and simple method for estimating the vanishing line of stochastically textured planes in real world images. The algorithm does not require pre-segmentation of textels, but uses SIFT detector to find pairs of correspondent image features satisfying special properties. It can be applied directly to raw images. The algorithm demonstrates a good performance on the examples of texture where other algorithm of its class perform poorly. However, the currently achieved accuracy is limiting its use in horizon line detection. In order to integrate this algorithm
into a ship detection scheme the accuracy of vanishing line detection needs to be improved and several issues need to be addressed. First, in those images where the horizon is present the SIFT keypoints may belong to regions of the image other than surface plane. Thus, the some feature points may vote for different vanishing lines from different textured planes, breaking the most important assumption. In order to resolve such ambiguity a clustering step needs to be performed to cluster the SIFT points by appearance. Using only points from the same cluster it is possible to minimize the influence of the keypoints not belonging to the textured plane. Another important requirement for algorithm to work is the presence of well textured plane showing significant perspective distortion. The current limited accuracy of the algorithm can be explained by a simplistic geometrical triangulation shown in Figure 6.3. Possible improvements in this triangulation process may be achieved by considering distortion of area around the keypoints as a function of distance to the estimated line, similarly to the approach in [90].
Figure 6.5. Examples of vanishing line detection on real images of water surface. The green line shows the groundtruth. The red line shows the output of the algorithm.
CHAPTER 7
CONCLUSIONS

This work presents novel algorithms for open-sea visual maritime surveillance using a highly non-stationary camera. The camera installed on a buoy is a subject to rapid erratic motion. The proposed algorithm detects, localizes, and tracks ships in the field of view of the camera and outputs images of the found targets. The experiments, conducted on a large dataset of video data obtained from a prototype of a buoy-based surveillance system, show good results. Specifically, the algorithm detects and tracks correctly of up to 88% of ships. In the context of ship detection, a new horizon detection scheme was developed for a complex maritime domain that provides accuracy of horizon localization of 98%, and detection of horizon images with the rate of 99%. To the best of our knowledge this is the first work that focuses on low-quality image data from highly non-stationary camera. The developed algorithms are fast and are well suited for low-powered autonomous systems deployed for long periods of time.

Another contribution of this dissertation is the development of a direct method for estimating vanishing line in difficult stochastic textures in real world images. The primary motivation for the development of such algorithm is to pave the way for the ship detection in situations with non-open sea. The ability to detect the sea horizon in such maritime images along with other changes in ship detection algorithm will extend the surveillance application. The advantage of the developed algorithm for vanishing line detection is in the fact that it does not require pre-segmentation of
textels, but uses SIFT detector to find pairs of correspondences satisfying special properties. Conducted evaluation of the algorithm showed good performance of vanishing line detection on a large set of images with various sea texture. Although the algorithm’s current accuracy limits its use in the ship detection algorithm, further improvements are possible and are suggested for future work.
REFERENCES


