Detecting Scallops in Images from an AUV

Einar Óli Guðmundsson

Faculty of Industrial Eng., Mechanical Eng.
and Computer Science
University of Iceland
2012

DETECTING SCALLOPS IN IMAGES FROM AN AUV

Einar Óli Guðmundsson

60 ECTS thesis submitted in partial fulfillment of a
Magister Scientiarum degree in Industrial Engineering

Advisor
Tómas Philip Rúnarsson

Faculty Representative
Magnús Örn Úlfarsson

M.Sc. committee
Tómas Philip Rúnarsson
Gunnar Stefánsson

Faculty of Industrial Eng., Mechanical Eng.
and Computer Science
School of Engineering and Natural Sciences
University of Iceland
Reykjavik, October 2012

Detecting Scallops in Images
from an AUV
Scallop Detection
60 ECTS thesis submitted in partial fulfillment of a M.Sc. degree in Industrial Engineering

Faculty of Industrial Eng., Mechanical Eng.
and Computer Science
School of Engineering and Natural Sciences
University of Iceland
Hjarðarhagi 2
107, Reykjavik
Iceland

Telephone: +354 525 4000

Bibliographic information:
Einar Óli Guðmundsson, 2012, Detecting Scallops in Images
from an AUV, M.Sc. thesis, Faculty of Industrial Eng., Mechanical Eng.
and Computer Science, University of Iceland.

Printing: Háskólaprent, Fálkagata 2, 107 Reykjavík
Reykjavik, Iceland, October 2012

Abstract

The University of Iceland owns an AUV and intends to use it for scallop abundance estimation. In November 2011 the AUV acquired images of the seabed in Breiðafjörður Fjord. These images are the basis for the topic of this thesis: seabed imaging and automatic detection of scallops with a scallop detector. The scallop detector first scans an image with a detection window and uses a classifier to predict wether a window includes a scallop or not. A voting method is then used on the positive windows to discard unwanted detections. First, the calibration of the AUV's camera is conducted and the results show minimal lens distortion affecting the camera. The training set is made from the AUV's images and then used for feature extraction. Two simple feature extraction methods are applied: gray-level thresholding; and color histograms. Three classification methods are proposed: nearest neighbor algorithm; distance to the average feature image; and SVMs. The classification methods are then used by the scallop detector. Experiments on the classifiers show that SVMs outperform the two other methods. Tuning of voting parameters is then conducted using three different scallop detectors. The combination detector, which first uses the color histogram classifier to find the most prominent area of the images and then the gray-level classifier on that area, shows the best results. Finally, the combination detector is tested on 20 images. The detector shows about 80% prediction accuracy.

Útdráttur

Háskóli Íslands á kafbát sem nota á til að meta stofnstærð hörpudisks. Í nóvember 2011 tók kafbáturinn myndir af hafsbotni Breiðafjarðar. Þessar myndir eru grunnurinn að umfjöllunarefni ritgerðarinnar; myndataka af hafsbotni og sjálfvirk talning hörpudiska með hörpudiskateljara. Hörpudiskateljarinn byrjar á því að skanna mynd með talningarglugga og notar svo flokkara til að spá fyrir um hvort glugginn innihaldi hörpudisk eða ekki. Kosningaraðferð er svo notuð á jákvæðu gluggana til að losna við óæskilegar talningar. Fyrst var kvörðun á myndavél kafbátarins framkvæmd sem leiddi í ljós litla bjögun á linsu myndavélarinnar. Þjálfunargögn voru búin til úr myndum frá kafbátnum sem síðan voru notuð til útdrátts sérkenna (e. features). Tvö einföld sérkenni voru skoðuð, annarsvegar þröskuldur á gráma (e. gray-level thresholding) og hinsvegar litastuðlarit (e. color histograms). Settar voru fram þrjár flokkunaraðferðir; reiknirit næsta nágranna, fjarlægð til meðalsérkennismyndar og SVM. Flokkunaraðferðirnar voru í framhaldi notaðar af hörpudiskateljaranum. Tilraunir á flokkurunum sýna að SVM flokkararnir standa sig betur en hinar tvær aðferðirnar. Stillingar á kostningarstikum voru framkvæmdar á þremur mismunandi hörpudiskateljurum. Samsetningarteljarinn, sem fyrst notar litastuðlaritsflokkarann til að finna staðsetningar sem líklegast innihalda hörpudiska, og síðan grámaþröskuldsflokkarann á þau svæði, sýndi bestu niðurstöðurnar. Samsetningarteljarinn var prófaður á 20 myndum. Teljarinn nær um 80% spánákvæmni.

Contents

List of Figures

List of Tables

Abbreviations

- AUC - Area Under the (ROC) Curve
- AUV - Autonomous Underwater Vehicle
- CCD - Charged-coupled Device
- k-nn - k-Nearest Neighbor
- OSH - Optimal Separating Hyperplane
- RBF - Radial Basis Function
- ROC - Receiver Operating Characteristic (Curve)
- SVM - Support Vector Machine

1. Introduction

A scallop is a type of shellfish which many people find a delicacy. The commercial catching of scallops in Breiðafjörður Fjord, West Iceland, started in 1970 but stopped in 2004 when the scallop population reached a record low. Since then a lot of research has been done to estimate the abundance of scallops in Breiðafjörður. The traditional research method is to use a plough to catch scallops from known scallop sites and then count them and measure their size (Jónasson, 2007). This method is time consuming and covers only a small part of the known scallop sites. Jónasson (2007) notes that automatic submarines could be a useful tool in the future for estimating scallop abundance since they can cover a greater area and give additional information, such as salinity and temperature, than the traditional method.

An autonomous underwater vehicle (AUV) is a robot that travels underwater and does not require input from an operator. The University of Iceland owns an AUV and intends to use it to research the Icelandic scallop in Breiðafjörður Fjord. The AUV has a digital camera which takes pictures of the seabed which could then be used to count scallops.

1.1. Motivation

Manually counting objects in images can be a very time consuming task, especially if the images number in the thousands. Therefore, if an automatic method for the task is available it could save a lot of time and effort. Object detection in images is a subject that has been widely documented in literature in the last couple of decades. The objects detected can, for example, be faces (see e.g. Papageorgiou and Poggio (2000)) or cars (see e.g. Zhao and Nevatia (2003)). The only reference for automatic detection of scallops, to my knowledge, is by Gallager et al. (2009). The authors do not mention which method they use, only that classification was used. However, the detection of scallops is in many ways similar to face detection, which is a highly referenced subject. Face detection, like the detection of scallops, tackles the problem of finding objects in images which have clutter-intensive backgrounds (Hjelmas and Low, 2001). The faces usually cover a small part of the image, which is also the case of the scallop images. Ideas from the face detection community can therefore be used when an automatic scallop detector is made. The question, therefore, is: how can digital images of the seabed be used for scallop detection?

1.2. Objective and Contribution

The objective of this thesis is to use images from an AUV of the seabed to find and count scallops. This includes the calibration of the AUV's camera, which gives parameters that describe lens distortion, and the construction of an automatic scallop detector.

The author's contribution is the following:

- Calibration of the AUV's camera that results in the intrinsic parameters that describe lens distortion and can also be used for inverse mapping from pixel space to metric space.
- Training set with 905 scallop (positive) instances and 15,536 "no scallop" (negative) instances.
- Two support vector machine classifiers that predict if a detection window includes a scallop or not. The classifiers use two simple feature extraction methods, gray-level thresholding and color histograms.
- Voting method that discards probable false positives and overlapped detections from the classifiers when an image from the AUV is scanned.
- Automatic scallop detector that uses the classifiers and the voting method with about 80% detection accuracy.

1.3. Thesis Structure

Chapter 2 gives technical details of the AUV and how camera calibration and image editing are useful tools for images from an AUV. In Chapter 3 there is a description of how scallop images are acquired with the AUV and how the images are used to make a training set for the classification methods, which is then used for feature extraction. Two different feature extraction techniques are explained: gray-level thresholding and color histograms. The features are then used by the classification methods, which are described briefly in Chapter 4. Chapter 4 also explains how scallop detection can be performed by using the classifiers and a voting method. In Chapter 5 experiments are conducted on the classification methods with the two different features. Voting parameters are then tuned for three scallop detectors that use the classifiers that performed best in the experiments. Finally, the most promising scallop detector is used on 20 test images and the results presented. In the last chapter the results of the experiments are discussed and improvements and other methods are mentioned.

2. Seabed Imaging

Benthos is the name given to the community of organisms which live on or near the seabed. Acquiring images of the benthos requires some sort of a device which carries a camera to take underwater images. An autonomous underwater vehicle is one such device as it can carry a digital camera and/or other instruments to monitor the seabed. The images are used for various purposes, like counting benthic organisms for abundance or size estimation. However, seabed images are often imperfect. There are various factors which can affect the quality of the images, such as lighting conditions, visibility and camera settings. Imperfect components of the camera can also cause a distortion of the image which can affect the size estimation of objects because of pixel displacement in the image. There are numerous methods available to enhance images to improve their utility.

This chapter describes the AUV used for imaging and the image acquisition hardware. This is followed by a section on camera calibration, using a mathematical model of the camera system, in order to correct image distortion. Furthermore, a section describing some basic image processing procedures for enhancing the images is presented.

2.1. The Gavia AUV

The Gavia is a modular based AUV made up of several individual cylindrical modules. It is manufactured by Teledyne Gavia ehf in Kópavogur, Iceland. The Gavia, owned by the University of Iceland, was manufactured in 2007. It has the following five modules:

- Nosecone/camera module.
- Battery module.
- DVL (doppler velocity log) module.
- Control module.

- Propulsion module.

At the bottom of the control module is the camera strobe which is made of 20 powerful LEDs from Lumiled. The total light intensity coming from the strobe, according to the AUV's manual, is 2400 lum. The strobe is synced to the camera which is located in the nosecone module. The light from the LEDs is cold white which makes color images look blue. Underwater most attenuation is in the red part of the spectrum and the blue part gets the smallest attenuation which makes color images look even bluer, see Figure (2.1) (Gavia, 2011).

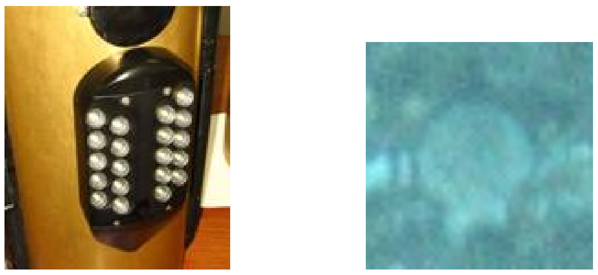


Figure 2.1: The camera strobe located under the control module (on the left) and an image of a scallop (on the right). Note the blue tone of the image.

The digital camera, located in the nosecone module, is a high-performance low-light variable frame rate camera. It is manufactured by Point Grey Research Inc. The camera comes with a dedicated Linux-PC and a solid state disc for data storage. The frame rate of the camera is configured to guarantee bottom coverage at 2 m/s speed of the Gavia (approximately 600 rpm) with 50% overlap between images. This is done to keep data gathering to a minimum (Gavia, 2011). Technical specifications of the camera are shown in Table (2.1).

Figure (2.2) shows a Gavia AUV, the position of the camera strobe and camera, and the resulting image area. The light beam from the strobe comes at an angle to the image area. The area of the image closest to the strobe is therefore brighter than the area furthest away from the strobe. The strobe also creates a shadow from objects in the image on the end facing away from the strobe.

2.1.1. Quality of the AUV's Images

The AUV's camera is designed to acquire images of fairly large objects, such as mines (Gavia, 2011), and therefore is not ideal for the research of benthos organisms. In addition, the pixel resolution in color mode is *only* 800 × 600 pixels. Because of this, the AUV's altitude (distance from the AUV to the seabed) needs to be low so that small benthos organisms can be detected. Figure (2.3) shows a part of an

Table 2.1: Technical specifications for the SCOR-20SO digital camera (Research, 2005).

Specification	Description
Sensor Overview	Sony 1/1.8" ICX274 Progressive Scan CCD (Monochrome/Color) Global shutter
Sensor Resolution	1600x1200 pixels
Sensor Size	Diagonal 8.923mm 1/1.8" CCD
Sensor Chip Size	8.50mm(H) x 6.80mm(V)
Sensor Unit Cell Size	4.4μm(H) x 4.4μm(V)
Frame Rates (max)	15FPS, 1600x1200
Analog/Digital Converter	12-bit A/D
Focal length	6mm
Gain	-10 to 26dB
Shutter	0.03ms to 66.6ms @ 15FPS
Extended Shutter	0.02ms to 3296ms @ 15FPS
Gamma	0.50–4.00

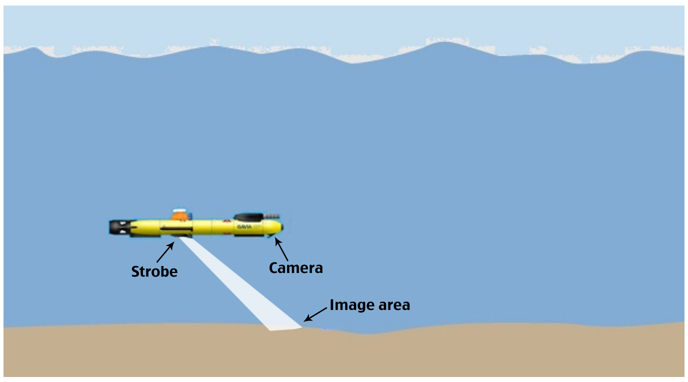


Figure 2.2: The Gavia strobe, camera and image area.

image obtained by the AUV, at altitude 2.17 m. Even at a relatively low altitude it is difficult to identify the benthos organisms in the image.

The following issues affect the quality of the AUV's images:

- The images look blue because of the light from the AUV's strobe and because of attenuation underwater.
- Brightness is greater in the lower part of images since it is closer to the AUV's strobe. Upper corners and edges of images have low brightness.
- Image noise is visible because of bad lighting conditions.

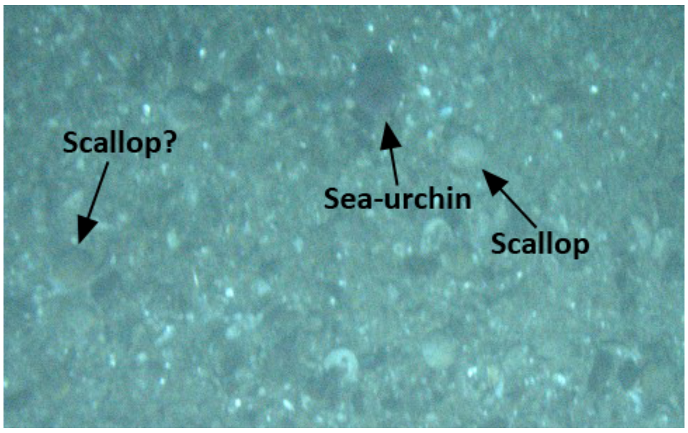


Figure 2.3: Image sample from the AUV's camera, with altitude 2.17 m.

- The altitude of the AUV has to be low (around 2 m) so that benthic organisms can be detected on the images.

- Plankton (such as algae) can affect visibility even at low altitude.

2.2. Camera Calibration

Camera calibration, or camera resectioning, is the process of finding the parameters that describe the internal geometric and optical characteristics, called *intrinsic* parameters, and/or the three dimensional orientation and position of the camera relative to a world coordinate system, called *extrinsic* parameters (Heikkila and Silven, 1997). The parameters from the camera calibration can be useful for the following two reasons:

1. Distortion parameters are estimated. If distortion is significant in an image it will distort the object and change the "real" size of the object.

2. The intrinsic parameters can be used to map points from pixel space to the camera coordinate system (called inverse mapping). They can then be used to estimate metric distances between two points in an image.

Camera calibration is usually performed by taking multiple pictures of a calibration pattern at different lengths and orientations from the camera. The model used in calibration is described below and the camera parameters are estimated from the model. The model described is based on (Zhang, 2000) and (Heikkila and Silven, 1997) which are the basis of the MATLAB camera calibration toolbox. The

MATLAB camera calibration toolbox is then used to perform a calibration on the AUV's camera.

2.2.1. The Pinhole Model

The pinhole camera model is a simplified camera model that describes the projection from a point in object space, the world coordinate system, to image space, the image coordinate system (see Figure (2.4)). In the pinhole camera model each point in object space is projected by a straight line through a projection center to the image plane (Heikkila and Silven, 1997). The following coordinate systems are used in this section:

- (X, Y, Z) - world coordinate system with origin at O that represents the coordinates of any visible point P in object space.
- (x, y, z) - camera-centered coordinate system with origin at O_c that represents the same point, P, in camera coordinates.
- $(\widetilde{u}, \widetilde{b})$ - image coordinate system with origin at the principle point, O'.
- (u, v) - image coordinate system in *pixels* with origin at the top-left corner of the image.

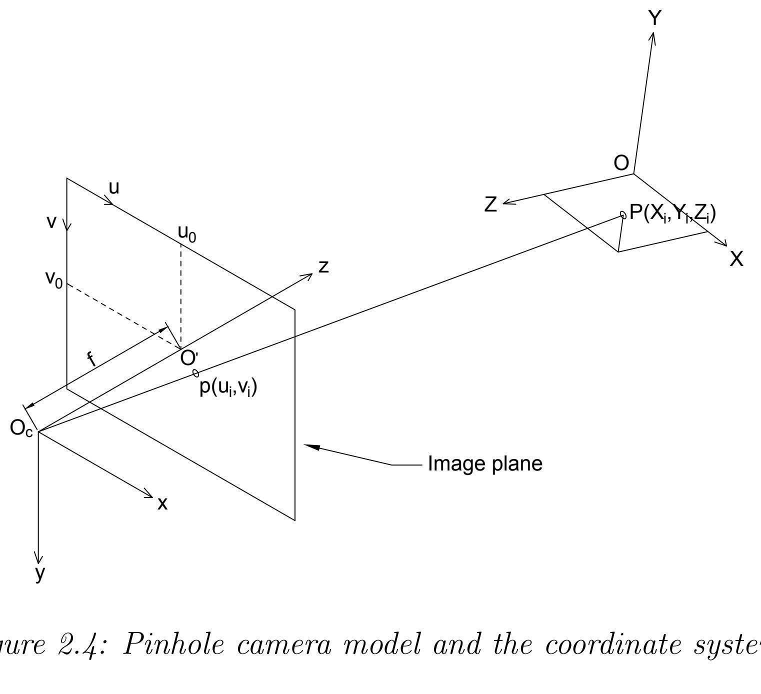


Figure 2.4: Pinhole camera model and the coordinate systems.

The image plane, (u, v), is assumed to be parallel to the camera plane, (x, y, z), and set at a distance f to the origin, where f denotes the (effective) focal length of the camera (Weng et al., 1992). When a point in object space is projected to camera coordinates the *extrinsic* parameters are needed. The relationship between an arbitrary point $P(X_i, Y_i, Z_i)$ in the world coordinate system and its location in the camera coordinate system, $p(x_i, y_i, z_i)$, is given by

$$\begin{bmatrix} x_i \\ y_i \\ z_i \end{bmatrix} = \mathbf{R} \begin{bmatrix} X_i \\ Y_i \\ Z_i \end{bmatrix} + \mathbf{t} \tag{2.1}$$

where $\mathbf{R}$ is a 3×3 rotation matrix defining the camera orientation and $\mathbf{t} = (x_0, y_0, z_0)^{\mathrm{T}}$ is a translation vector defining the camera position (Weng et al., 1992). The rotation matrix uses the Euler angles ω, φ and κ to represent three rotations around the x, y and z-axis respectively.

The projection from camera coordinates to image coordinates is well defined for the pinhole model (by using similar triangles). The *intrinsic* parameters are needed to perform the transformation. They usually include the (effective) focal length f, scale factor s_u, and the principle point (u_0, v_0) (the image center) (Heikkila and Silven, 1997). The projection of point $p(x_i, y_i, z_i)$ to the image plane is given by

$$\begin{bmatrix} \widetilde{u}_i \\ \widetilde{v}_i \end{bmatrix} = f \begin{bmatrix} x_i/z_i \\ y_i/z_i \end{bmatrix} = f\mathbf{x}_n \tag{2.2}$$

where $\mathbf{x}_n \equiv (x_{n_1}, x_{n_2})^{\mathrm{T}} = (x_i/z_i, y_i/z_i)^{\mathrm{T}}$ are called normalized image coordinates. To transform $(\widetilde{u}_i, \widetilde{v}_i)$ from metric coordinates to pixels, two coefficients, D_u and D_v, are needed. The transformation is expressed as

$$\begin{bmatrix} u'_i \\ v'_i \end{bmatrix} = \begin{bmatrix} D_u s_u \widetilde{u}_i \\ D_v \widetilde{v}_i \end{bmatrix} + \begin{bmatrix} u_0 \\ v_0 \end{bmatrix} \tag{2.3}$$

where the point (u'_i, u'_i) represents the arbitrary point P on the image plane in pixels. The two coefficients D_u and D_v can usually by obtained from the camera manufacturer but in fact they are not needed since they are linearly dependent on the focal length f (Heikkila and Silven, 1997). It is therefore convenient to define two new coefficients, $f_u \equiv D_u s_u f$ and $f_v \equiv D_v f$, called row focal length and column focal length respectively (Weng et al., 1992). The model then becomes

$$\begin{bmatrix} u_i \\ v_i \end{bmatrix} = \begin{bmatrix} f_u & 0 \\ 0 & f_v \end{bmatrix} \mathbf{x}_n + \begin{bmatrix} u_0 \\ v_0 \end{bmatrix} \tag{2.4}$$

or in homogeneous coordinates

$$\begin{bmatrix} u_i \\ v_i \\ 1 \end{bmatrix} = \begin{bmatrix} f_u & 0 & u_0 \\ 0 & f_v & v_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_{n_1} \\ x_{n_2} \\ 1 \end{bmatrix}. \tag{2.5}$$

This model is only an approximation of the real camera model. The point (u_i, v_i) will generally not equal the observed point because of lens distortion, see Figure (2.5).

Distortion

The model described by Equation (2.4) does not usually hold true due to positional error caused by geometrical distortion. This distortion causes an observed point in the image plane to shift slightly. By adding an error term to Equation (2.2) this distortion can be accounted for:

$$\begin{bmatrix} u_i' \\ v_i' \end{bmatrix} = \begin{bmatrix} \widetilde{u}_i \\ \widetilde{v}_i \end{bmatrix} + \begin{bmatrix} \delta_u(\widetilde{u}_i, \widetilde{v}_i) \\ \delta_v(\widetilde{u}_i, \widetilde{v}_i) \end{bmatrix} \tag{2.6}$$

where $(\widetilde{u}_i, \widetilde{v}_i)$ is the non-observable distortion-free point in image coordinates, (u_i', v_i') is the corresponding point with distortion, and $\delta_u(\widetilde{u}_i, \widetilde{v}_i)$, $\delta_v(\widetilde{u}_i, \widetilde{v}_i)$ represent the error caused by distortion. For reasons of convenience it is more suitable to use the normalized coordinates in Equation (2.6) so that:

$$\begin{bmatrix} u_i' \\ v_i' \end{bmatrix} = f\left(\begin{bmatrix} x_{n_1} \\ x_{n_2} \end{bmatrix} + \begin{bmatrix} \delta_u(x_{n_1}, x_{n_2}) \\ \delta_v(x_{n_1}, x_{n_2}) \end{bmatrix} \right). \tag{2.7}$$

Two types of distortions are most commonly used in camera models, radial lens distortions and tangential lens distortion. Figure (2.5) shows how a point is distorted because of radial and tangential distortion. Weng et al. (1992) consider different causes of distortion that either induce radial distortion or both radial and tangential distortion:

1. **Radial Distortion:** Mainly caused by a flawed radial curve of the camera's lens elements. The distortion causes the actual image point to displace radially in the image plane.

2. **Decentering Distortion:** In real optical systems the optical centers of lens elements are typically not collinear and therefore cause decentering distortion. This type of distortion has both radial and tangential components but only the tangential components are usually used (Heikkila and Silven, 1997).

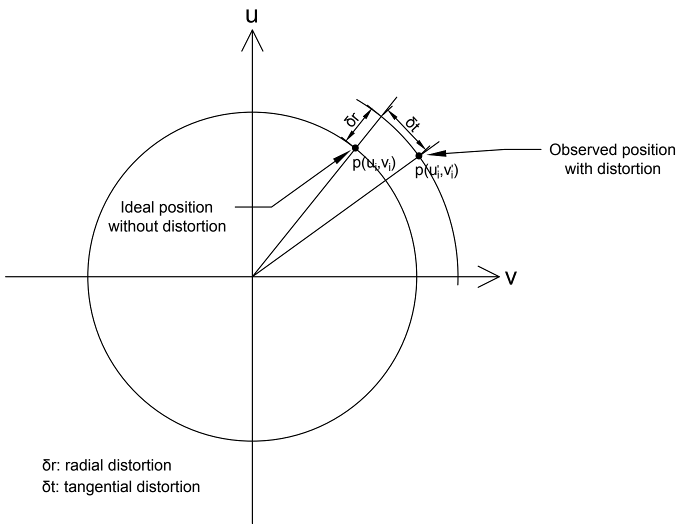

u
v
δr
δt
p(u_i,v_i)
p(u'_i,v'_i)
Observed position with distortion
Ideal position without distortion
δr: radial distortion
δt: tangential distortion

Figure 2.5: Effect of radial and tangential distortion (Weng et al., 1992).

The formulation for radial distortion and tangential (decentering) distortion is shown in (Heikkila and Silven, 1997). The image coordinates with distortion can then be written as

$$\begin{bmatrix} u'_i \\ v'_i \end{bmatrix} = \begin{bmatrix} x_{n_1} \\ x_{n_2} \end{bmatrix} + \begin{bmatrix} x_{n_1}(k_1 r^2 + k_2 r^4 + k_3 r^6) + 2p_1 x_{n_1} x_{n_2} + p_2(r^2 + 2x_{n_1}^2) \\ x_{n_2}(k_1 r^2 + k_2 r^4 + k_3 r^6) + 2p_2 x_{n_1} x_{n_2} + p_1(r^2 + 2x_{n_2}^2) \end{bmatrix} \tag{2.8}$$

where $r^2 \equiv x_{n_1}^2 + x_{n_2}^2$, k_1, k_2 and k_3 are radial distortion parameters and p_1 and p_2 are tangential distortion parameters.

Other types of distortion have been proposed, such as thin prism distortion and linear distortion (Weng et al., 1992). Heikkila and Silven (1997) note that these correction terms are only relevant if the image axes are not orthogonal and in most cases the linear distortion component is insignificant.

The Pinhole Model With Distortion

The complete camera model is achieved by combining the basic model with radial and tangential distortion. The pinhole model with distortion can be expressed with the homogeneous equation

$$\begin{bmatrix} u_i \\ v_i \\ 1 \end{bmatrix} = \mathbf{K} \begin{bmatrix} u_i' \\ v_i' \\ 1 \end{bmatrix} \tag{2.9}$$

where the matrix $\mathbf{K}$ is known as the camera matrix, and is defined as

$$\mathbf{K} = \begin{bmatrix} f_u & 0 & u_0 \\ 0 & f_v & v_0 \\ 0 & 0 & 1 \end{bmatrix}.$$

This model has the following set of non-distortion intrinsic and extrinsic parameters

$$m = (u_0, v_0, f_u, f_v, \mathbf{t}, \omega, \phi, \kappa)^{\mathrm{T}} \tag{2.10}$$

and a set of distortion parameters

$$d = (k_1, k_2, k_3, p_1, p_2)^{\mathrm{T}}. \tag{2.11}$$

The objective of the camera calibration is to find the optimal values for the parameters in Equations (2.10) and (2.11) by using observations of a known 3D target, called a calibration target.

Solution Methods for The Calibration Problem

The camera calibration problem is nonlinear and consists of two sets of parameters to be estimated, seen in Equations (2.10) and (2.11). Let (U_i, V_i), $i = 1, \ldots, N$ denote the image coordinates of N observations of calibration points. The camera parameters (from the camera matrix $\mathbf{K}$) can then be solved by minimizing the residuals between the model and observed points. If the error is assumed to be white Gaussian noise then the cost function becomes

$$\min_{m,d} F = \frac{1}{2}\left(\sum_{i=1}^{N}(U_i - u_i(m,d))^2 + \sum_{i=1}^{N}(V_i - v_i(m,d))^2\right) \tag{2.12}$$

where $u_i(m,d)$ and $v_i(m,d)$ are pixel coordinates of point i as functions of the parameters in Equations (2.10) and (2.11). The Levenberg-Marquard algorithm for nonlinear optimization has been shown to provide the fastest convergence for this problem. However, the quality of the solution is dependent upon the initial estimates of parameters, and improper initial estimates can cause the optimization to find a local minimum instead of the global solution (Heikkila and Silven, 1997). Good initial estimates can be found with different methods. One method is based on so called vanishing points constraints, see Bouguet and Perona (1998) and Liu et al. (2004).

2.3. Calibration of The AUV's Camera

For the camera calibration a planar calibration rig with a checker board pattern was made. The AUV was brought to a local swimming pool and the camera was used for taking pictures of the calibration rig at various altitudes and orientations. The MATLAB camera calibration toolbox is used for computations. Figure (2.6) shows the 45 images that were used for the calibration.

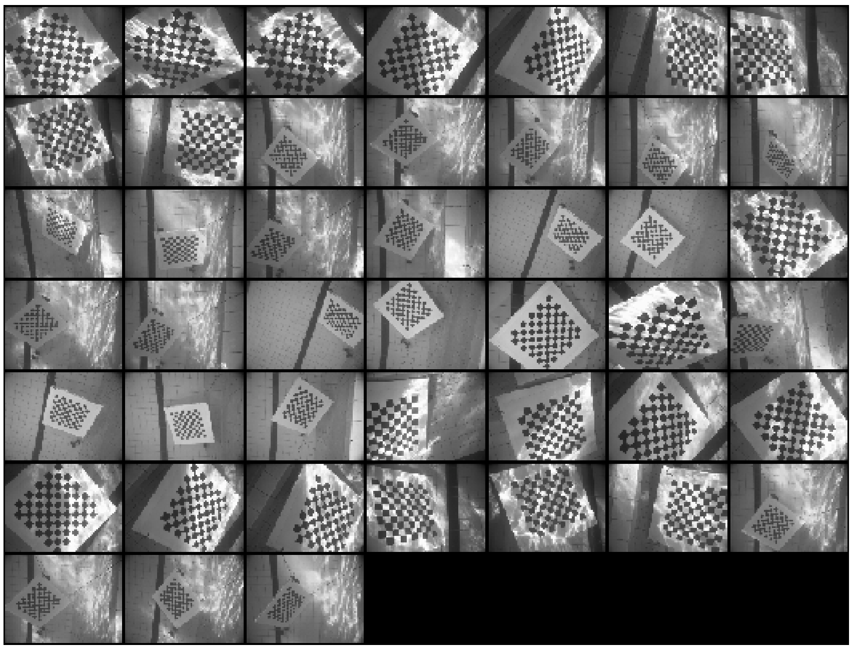


Figure 2.6: The images used for calibration.

The camera calibration resulted in the following camera matrix:

$$\mathbf{K} = \begin{bmatrix} f_u & 0 & u_0 \\ 0 & f_v & v_0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 957.6 & 0 & 419.9 \\ 0 & 957.3 & 333.4 \\ 0 & 0 & 1 \end{bmatrix} \pm \begin{bmatrix} 2.8 & 0 & 3.3 \\ 0 & 2.7 & 3.0 \\ 0 & 0 & 0 \end{bmatrix}$$

and distortion parameters:

$$d = \begin{bmatrix} k_1 \\ k_2 \\ p_1 \\ p_2 \end{bmatrix} = \begin{bmatrix} 0.059 \\ 0.253 \\ 0.008 \\ 0.013 \end{bmatrix} \pm \begin{bmatrix} 0.009 \\ 0.050 \\ 0.001 \\ 0.002 \end{bmatrix}.$$

The radial distortion parameter k_3 was found to be statistically insignificant and was therefore not used in the calibration. It is interesting to see that the principle point, (u_0, v_0), is some distance from the true center of the image, $(400, 300)$, which implies that the AUV's camera lens is not a perfect sphere. The ratio f_v/f_u, often called *aspect ratio*, describes the ratio between the horizontal and vertical lengths of the CCD sensor cell (Zhang, 2000). Table (2.1) shows that the aspect ratio for the AUV's camera is 1, which is the same as the result from the camera calibration.

Figure (2.7) shows the distortion of the AUV's camera. The figure shows the impact of the lens distortion on each pixel of the image. The arrows represent the pixel displacement caused by the lens distortion. There is little displacement in most of the image except for the upper and lower right corners, which have over 12 pixels of displacement. The displacement will have little impact on the estimation of scallop sizes since the largest scallops only cover roughly 40×40 pixels of an image. If a scallop were situated in the bottom right corner of an image the pixel displacement would cause a maximum error of 3-4 pixels.

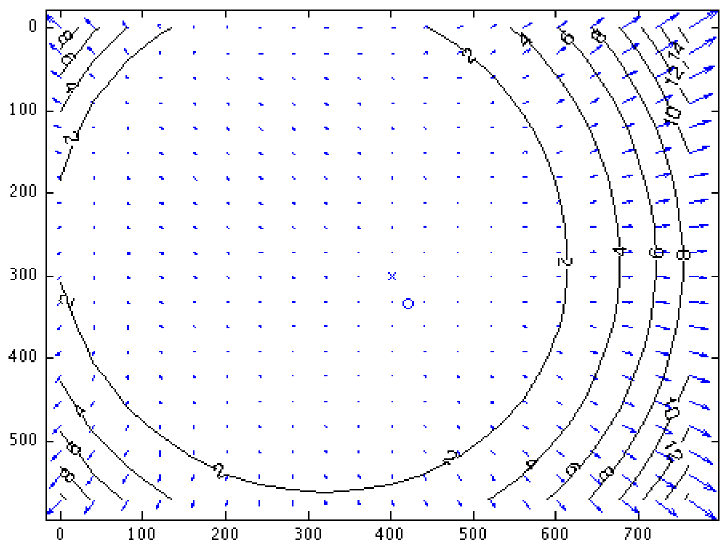


Figure 2.7: Image distortion of the AUV's camera. The x represents the image center and the ring the principle point, (u_0, v_0).

2.3.1. Inverse Mapping from Pixel Space to Metric Space

When the camera matrix, $\mathbf{K}$, has been found it can be used for the inverse mapping from distorted pixel coordinates, (u, v), to normalized camera coordinates, (x_{n_1}, x_{n_2}). There exists no analytic solution for the inverse mapping because of the high degree distortion model, however, it can be solved numerically. Using Equation (2.9) the distorted image coordinates can be found by:

$$\begin{bmatrix} u' \\ v' \end{bmatrix} = \begin{bmatrix} (u-u_0)/f_u \\ (v-v_0)/f_v \end{bmatrix}.$$

Then by using Equation (2.8) the normalized camera coordinates can be found recursively by Algorithm 1.

Algorithm 1 Recursive algorithm for finding normalized camera coordinates

```
Initialize $(x_{n_1}, x_{n_2}) = (u', v')$
Choose $N$ (e.g. 20)
for $k = 1,\ k++,\ k \leq N$ do
    $r^2 \leftarrow x_{n_1}^2 + x_{n_2}^2$
    $x_{n_1} \leftarrow u' - \frac{2p_1 x_{n_1} x_{n_2} + p_2(r^2 + 2x_{n_1}^2)}{1 + k_1 r^2 + k_2 r^4}$
    $x_{n_2} \leftarrow v' - \frac{2p_2 x_{n_1} x_{n_2} + p_1(r^2 + 2x_{n_2}^2)}{1 + k_1 r^2 + k_2 r^4}$
end for
```

When the normalized camera coordinates have been calculated the camera coordinates, (x, y), can easily be found by using Equation (2.2), *if* it is assumed that the camera is perpendicular to the image area (or equivalently z = constant). The AUV gives information of the altitude for each image, which is then used for the estimation of z. This assumption is justified, in the case of the AUV's camera, by the fact that the AUV is set on bottom tracking and therefore aims to be parallel to the seabed, and hence the camera to be perpendicular to the image area. Finally, with known camera coordinates of two points in an image, the metric distance between them is found by calculating the euclidean distance between the two points.

To validate the method, an image of a pattern with known lengths is used. Figure (2.8) shows this image and the estimated lengths in centimeters using the above method. The real length is 13 cm. The estimated lengths have a mean of 12.9 cm and a standard deviation of 0.2 cm. There may be many causes for the underestimation of the length; the error could be caused by the assumption that the camera is perpendicular to the image area, there could be error in the altitude given by the AUV or the error could be caused by the user when clicking on the points in the image.

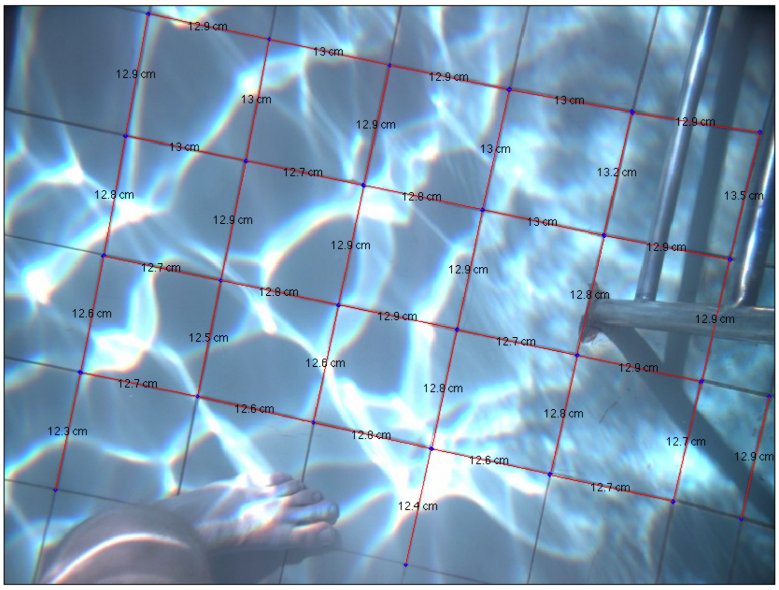


Figure 2.8: Image from the AUV of a pattern with known lengths and the estimated lengths.

2.4. Image Editing

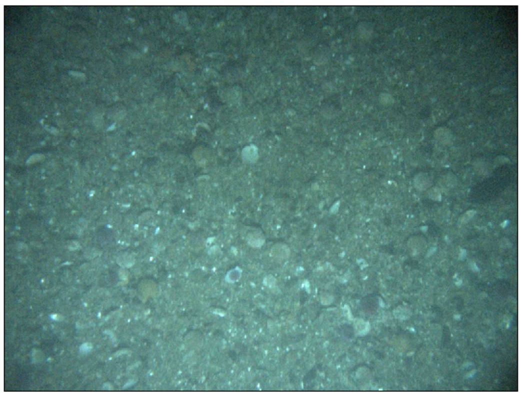
Image editing is the name given to the process of altering images. Image editing can be useful for enhancing underwater images which often have low brightness, compressed pixel intensity range and noise. Here a simple method is presented to tackle each of these common issues in underwater image acquisition. The image editing methods are beneficial when scallops are labeled (see Chapter 3) since they make it easier to detect scallops manually on the AUV's images. They can also be used in image preprocessing for the classification methods described in Chapter 4.

2.4.1. Intensity

Figure (2.9a) shows one common issue which arises in underwater images: they appear blue because the blue part of the spectrum gets the smallest attenuation. This causes the range of the pixel intensity values to be compressed, as the histogram in Figure (2.9b) shows. The intensity range can be expanded by using a normalization of the intensity values. The *imadjust* function in the MATLAB's image processing toolbox is used for the intensity adjustment. The function finds a limit for each color channel (Red, Green and Blue) of the image where the limit specifies the bottom 1% and the top 1% of all pixel values (these percentages can be changed by the user).

Then all pixel values between the limits are normalized to fit between pixel values 0–255 (where 0 denotes no intensity and 255 the most intensity). All pixel values below the lower limit are set to 0 and the pixel values above the upper limit are set to 255.

(a)



(b)

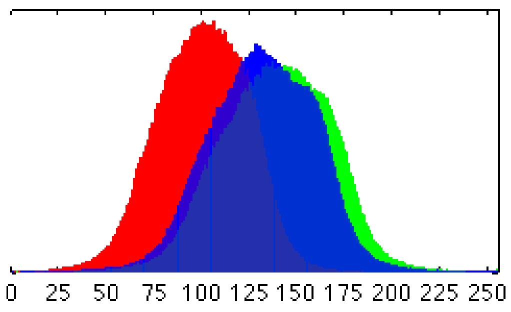
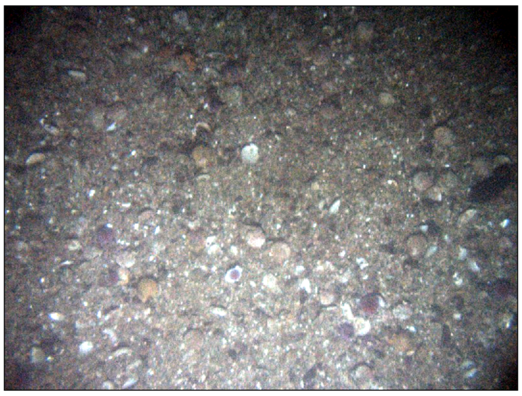


Figure 2.9: Raw underwater image (a) and its histogram of the three color channels, R, G and B (b).

The intensity adjusted version of the image in Figure (2.9) is shown in Figure (2.10). The image colors appear more "natural" and the blue tone has disappeared. The histogram also shows how the range of the pixel intensity values has changed, instead of being compressed, the pixel values are more evenly spread between 0 and 255.

(a)



(b)

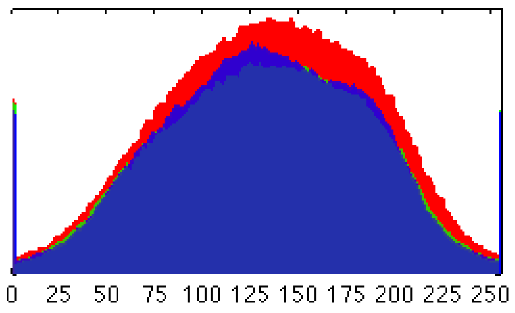


Figure 2.10: Intensity adjusted version of the image in Figure (2.9) (a) and its histogram of the three color channels, R, G and B (b).

2.4.2. Brightness

Adjusting the brightness of an image can be achieved by using the following point-wise operation (Farid, 2001):

$$g(u) = u^{\gamma}$$

where $u \in [0,1]$ denotes the original image pixel intensity, $g(u)$ the gamma adjusted pixel intensity and γ controls how much the brightness is changed. If $\gamma < 0$ then the new pixel value will be brighter than the original value and if $\gamma > 0$ then the new pixel value will be darker. Figure (2.11) shows a gamma adjusted version of the intensity adjusted image in Figure (2.10), with the original image in the center, the image on the left has $\gamma = 0.5$ and the image on the right has $\gamma = 1.5$.

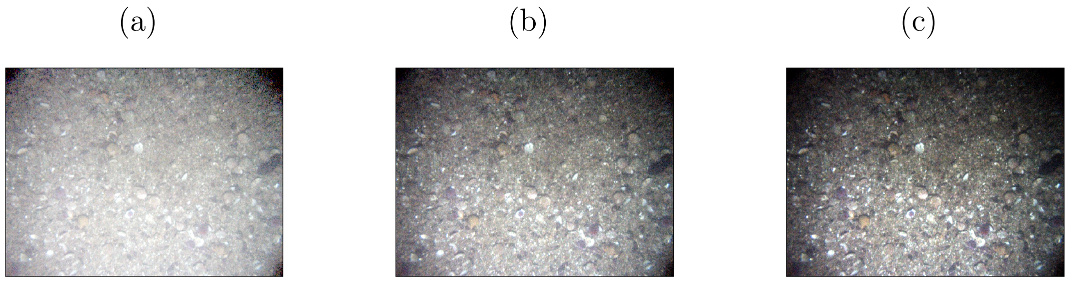


Figure 2.11: Gamma adjusted versions of the image in Figure (2.10), (a) $\gamma = 0.5$, *(b)* $\gamma = 1.0$ *and (c)* $\gamma = 1.5$.

2.4.3. Noise

In the intensity adjusted image in Figure (2.10) noise is clearly visible, represented by randomly occurring black and white pixels. This type of noise, often called salt and pepper noise, can be accounted for by using an image filter. The simplest type of image filter is called an average filter. Using an average filter gives each pixel the average pixel value of its n neighbor pixels. For example, if $n = 1$ then each pixel gets the average pixel value of all the pixels that are next to it. Figure (2.12) shows the upper half of the image in Figure (2.10) without and with an average filter. The noise, which is visible in the image without an average filter, is reduced substantially with the average filter. Other types of noise reduction filters exist, such as a median filter and a Gaussian filter.

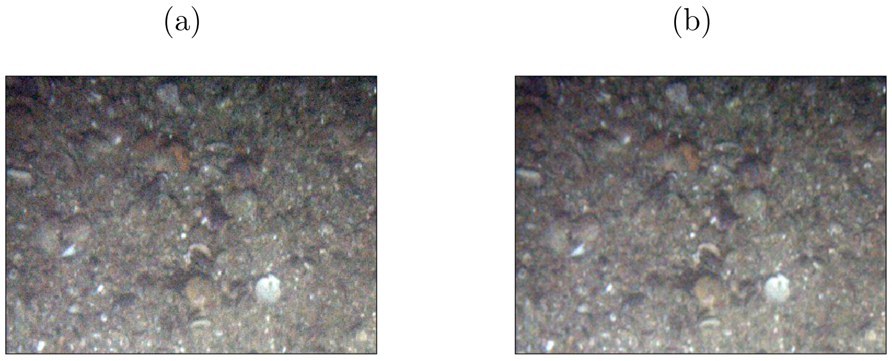


Figure 2.12: The upper half of the image in Figure (2.10) without an average filter (a) and with an average filter (b).

2.5. Summary

This chapter has described methods which are useful for underwater images from an AUV. The technical details of the AUV and its digital camera were given. The mathematical formulation of the camera calibration problem was described and a camera calibration was performed on the AUV's camera by using the MATLAB calibration toolbox. The results from the calibration were then used to describe the lens distortion of the AUV's camera and to perform the inverse mapping from pixel space to metric space, which was confirmed with a pattern of known distance. Finally, three image editing methods were introduced which are useful for the processing of underwater images.

3. Detecting Scallops on the Seabed

A seabed survey was conducted in a fjord in the West of Iceland known as Breiðafjörður. The aim was to estimate the abundance of scallops in the fjord. Images were acquired using the AUV described in the previous chapter. Counting scallops by hand from the thousands of images would be quite laborious so an automatic detection method would be of significant benefit. In order to achieve this goal a training set, with positive and negative examples of scallops, must be created (Alpaydin, 2010). It is important that the training set be carefully constructed in order for the classifiers, described in Chapter 4, to work properly. The construction of the scallop training set is discussed in this chapter. The cautious construction of features from training images is also of paramount importance. Two feature extraction methods are considered, both being very simple to implement and requiring little computational effort. Other feature extraction methods are also mentioned and the reason why they do not work for scallop detection is discussed.

3.1. Seabed Imaging

In November 2011 the AUV was brought to Stykkishólmur, West Iceland, for image collection. Stykkishólmur is a town on the south side of Breiðafjörður Fjord which is known for its scallop population. Images were collected from an area a few hundred meters north of Stykkishólmur, see Figure (3.1). The depth of the survey area is below 20 m so the only light source on the images is the light from the AUV's strobe. The AUV was set at 2 m bottom tracking (2 m from the camera to the seabed). The images are stored as 16-bit JPEG files in 800×600 pixel resolution. Figure (3.2) shows an example of one raw image collected by the AUV. It can be seen that the seabed is cluttered with different objects: scallops, shell fragments and sea-urchins, which are labeled on the figure. The quality of the images is also varied, some have good visibility, whilst others are either out of focus or have lower visibility.

The Icelandic scallop (*chlamys islandica*) is a rather small type of scallop, with the largest having a diameter of < 10 cm (Jónasson, 2007). It varies in size and color

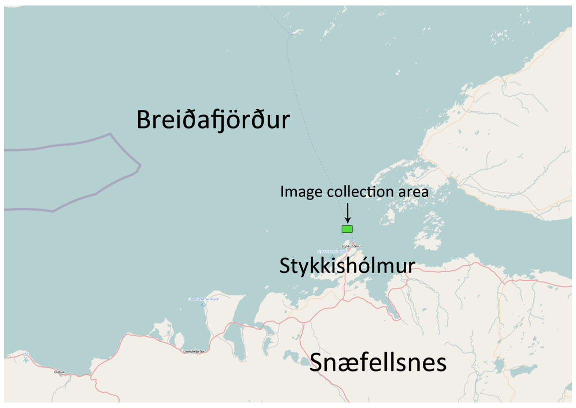


Figure 3.1: Breiðafjörður Fjord, West Iceland. The area where images were collected is highlighted.

as can been seen in Figures (3.2) and (3.3).

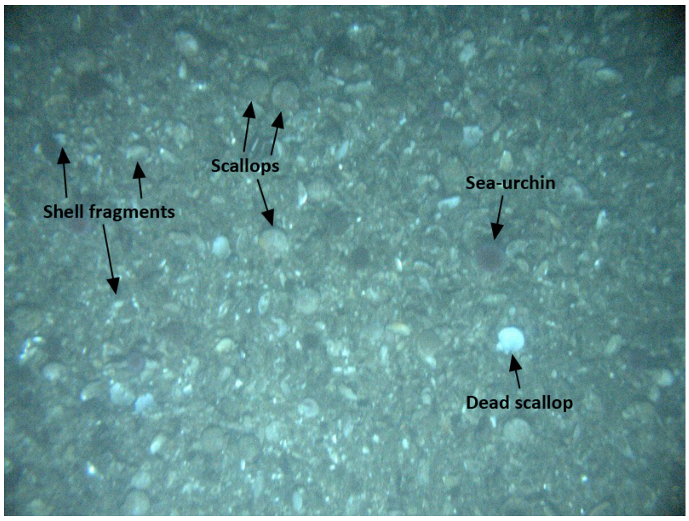


Figure 3.2: An example of a raw image collected north of Stykkishólmur and some of the common objects seen in the survey's images.

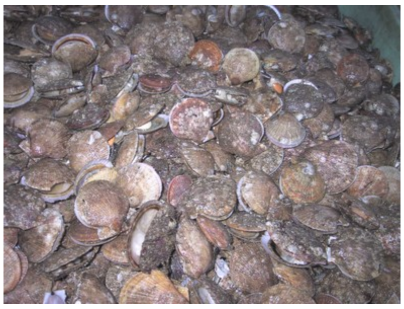


Figure 3.3: Image of a group of Icelandic scallops taken onshore.

3.2. Image Labeling

From the data survey in Breiðafjörður 1207 image frames were selected from different transects of the survey. Images with an altitude (distance from the AUV to the seabed) of less than 2.20 m are the only one used since the quality of the images (visibility and contrast) is reduced at higher altitudes. The altitude on these 1207 images is between 1.51–2.19 m and the average altitude is 2.014 m.

A MATLAB script is used to flip through the images and positive (scallop) and negative (no scallop) instances are found manually. For each positive instance the center of the scallop is targeted (with a few pixels of tolerance) and its coordinate on the image frame is stored. All positive instances are selected independent of size and orientation of the scallop. For the negative instances a coordinate where there is no scallop or a scallop far from the center is stored. Figure (3.4) shows labeled positive and negative instances of the same image frame. The low image quality makes it sometimes difficult to decide if an object is a scallop or not, therefore only definite instances of scallops are stored.

For each instance a 50×50 pixel bounding window, around the center of the stored coordinate, is extracted from the image frame. These 50×50 pixel windows are then used for feature extraction. The reason for the fixed size of the bounding window is:

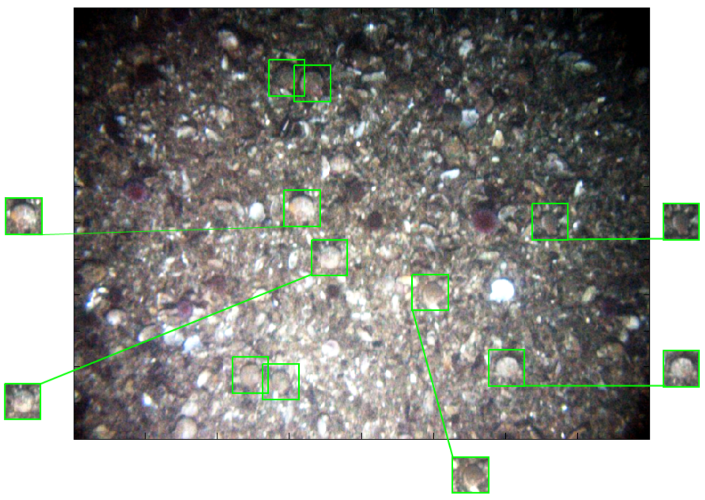
- All positive instances (scallops) fit inside a 50×50 pixel window. The largest scallop found has a diameter of ≈ 46 pixels.
- A fixed size window means that only one scale has to be considered which makes all calculations simpler.

The drawback is that bigger scallops will cover a greater area of the feature extraction window than smaller scallops, since only one scale is considered. This could favor bigger scallops over smaller in the detection phase. However, the fixed window size is justified by the fact that the quality of the images is low and thus it is very difficult to label small scallops for the training set.

A total of 905 positive and 15,536 negative examples were selected from the 1,207 image frames. The number of negative examples is higher than positive examples since the set of possible negative instances $\mathcal{N}$ is much bigger than the set of possible positive instances $\mathcal{P}$, or $|\mathcal{N}| \gg |\mathcal{P}|$.

Figure (3.5) shows the position (in each image frame) of all the labeled positive and negative instances. For the positive instances it is evident that most of the instances are around the center of each image, where the light from the AUV's strobe has the

(a)



(b)

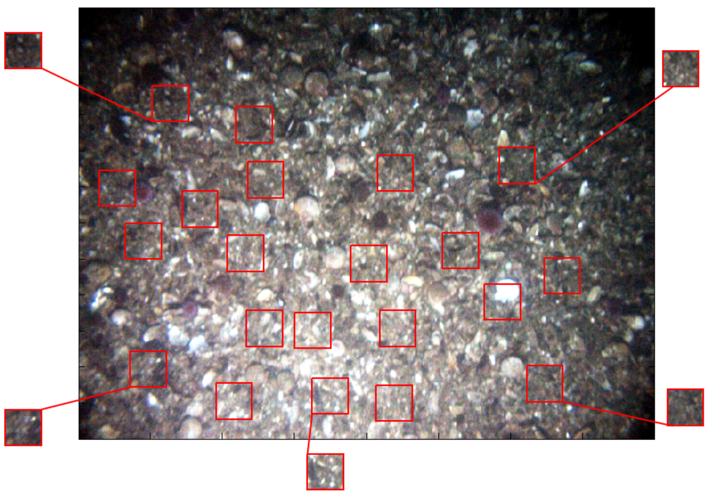


Figure 3.4: Example of labeled positive instances (a), and labeled negative instances (b), for the same image frame (1.88 m altitude and intensity adjusted).

greatest effect. In the top left and right corners, where the brightness is low, there are no positive instances labeled. Therefore, no definite scallop instances were found from this part of any image. The negative instances are more evenly spread since they were only selected by the notion that no scallops were situated inside the bounding window.

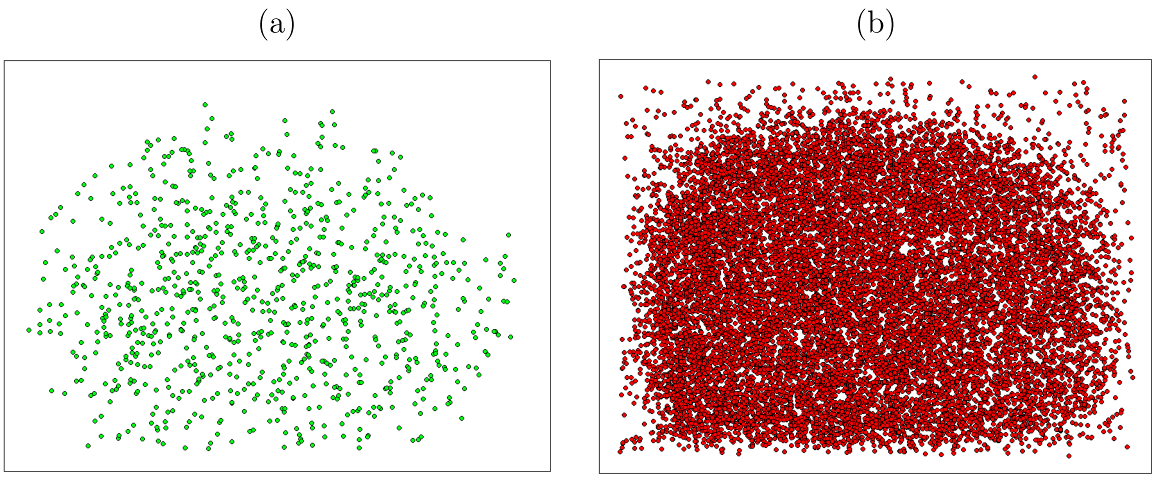


Figure 3.5: Image position of all labeled positive instances (a) and negative instances (b).

3.2.1. Augmenting the Positive Set

For better balance between the positive and negative sets the positive set is augmented by shifting the coordinate center of the bounding window by a few pixels in all directions since the coordinate center is selected with a tolerance of a few pixels anyway. For each positive instance 8 new positive instances are made by shifting the center of the window by 1 pixel in all directions from the original center. The size of the augmented positive set is then $9 \times 905 = 8{,}145$.

3.3. Warsha's Scallops

Warsha Singh, a Ph.d. student in marine biology at the University of Iceland, has counted around 1,400 scallops for her stock estimation of the Breiðafjörður scallop. Her scallops are from the same data survey as the labeled positive instances in the previous section, however, very few are from the same set of images. Of the 1,400 scallop instances from Warsha's data, 1,111 are from images with an altitude below 2.2 m. Her scallop instances are used for measuring the performance of the classifiers later on. Figure (3.6) shows examples of scallops found by Warsha. It can be seen that her dataset contains a greater number of smaller scallops than the positive dataset used for training. In addition, the center of the bounding window is not always as close to the center of the scallop as in the positive dataset.

In Figure (3.7) the image position of all of Warsha's scallops is shown. Compared to the position of the positive instances, shown in Figure (3.5), Warsha's scallops

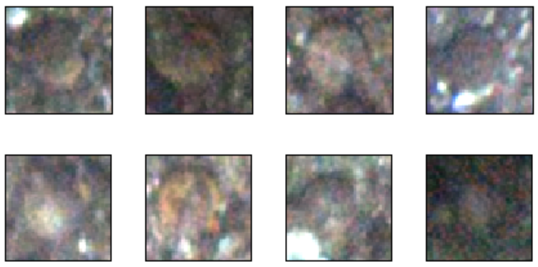


Figure 3.6: Examples of scallops found by Warsha (intensity adjusted).

are more evenly spread, even though most are in the center part of the image. This shows that Warsha's dataset and the positive dataset include somewhat different instances of scallops.

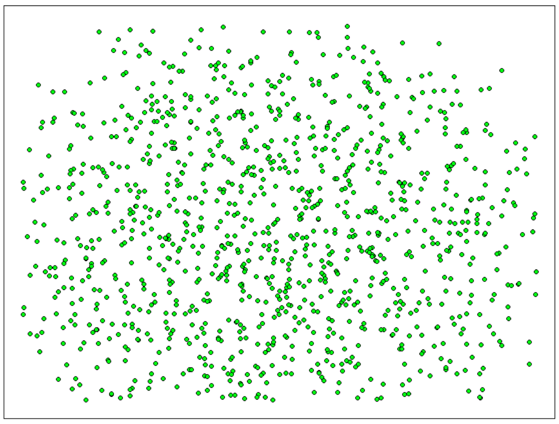


Figure 3.7: Image position of Warsha's scallops.

3.4. Feature Extraction

Choosing the right features for one's classifier is crucial for classifier performance. Classification literature describes numerous different feature extraction techniques, from Haar-Wavelets (Papageorgiou and Poggio, 2000) to pixel segmentation (Phung et al., 2005). When looking at an image of a scallop two features figure the most prominently: shape and color, as can be seen in Figure (3.8). The shape is fairly consistent - circular with tail (if visible) at the back of the scallop. The circular shape of the scallop is visible to an observer because of the shadow above of the scallops, which is the result of the AUV's strobe, or because of the difference in color between the scallop and the background. The color is more variable between individual instances of scallops.

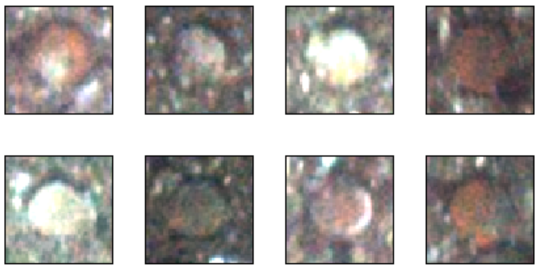


Figure 3.8: Images of scallops from the positive set (intensity adjusted). Note the circular shape.

3.4.1. Gray-Level Thresholding

Gray-level pixel values of objects (e.g. scallops) often vary significantly from the gray-levels of the background pixels. In these cases, thresholding can be an effective way to separate the background from the object (and hence detect shape). There are many methods available for gray-level thresholding but the output is always a binary image, where the foreground is represented by gray-level 0 and the background by the gray-level 255, or vice versa (Sezgin and Sankur, 2004). Gray-level thresholding has been used before in classification, see for example Rosin and Ellis (1995) where gray-level thresholding is used for shadow detection.

Otsu's Method

In Sezgin and Sankur (2004), an exhaustive survey of thresholding methods is conducted and the methods are evaluated based on a quantitative performance. They mention that Otsu's method is probably the most referenced thresholding method in literature and that it shows decent results compared to the other methods. MATLAB also has a function, as part of the Image Processing Toolbox, that performs gray-level thresholding using Otsu's method. For these two reasons Otsu's method is chosen for gray-level thresholding.

Otsu's method is a clustering-based method where the background and foreground pixels are assumed to belong to two classes, f and b. These classes are then clustered into two parts by minimizing the weighted sum of within-class variance. By noting that minimizing the within-class variance is equivalent to maximizing the between-class scatter we get the following thresholding function (Otsu, 1979):

$$\sigma_B^2(k^*) = \arg \max_{1\leq k<L} \left\{ \frac{[\mu_T \omega(k) - \mu(k)]^2}{\omega(k)[1-\omega(k)]} \right\} \tag{3.1}$$

where

$$\begin{aligned}
i = [0, \ldots, L] &\quad \text{(Pixel values of gray-level image)} \\
n(i) &\quad \text{(Total number of pixels at level i)} \\
N = n(1) + \cdots + n(L) &\quad \text{(Total number of pixels)} \\
p(i) = \frac{n(i)}{N} &\quad \text{(Probability mass function)} \\
\omega(k) = \sum_{i=0}^{k} p(i) &\quad \text{(Cumulative probability function)} \\
\mu(k) = \sum_{i=1}^{k} ip(i) &\quad \text{(Foreground mean)} \\
\mu_T = \mu(L) = \sum_{i=1}^{L} ip(i) &\quad \text{(Total mean of image)} \\
\sigma_B^2 &\quad \text{(Between-class variance)} \\
k^* &\quad \text{(The gray-level threshold).}
\end{aligned}$$

In Figure (3.9) both positive (scallop) and negative (no scallop) instances are shown, and their resulting binary images made with Otsu's method. The binary images are made with the *graythresh* function in MATLAB. The circular shape is clearly visible in the first and third positive class threshold images. The circular shape is not as clear in the positive class threshold image in the center. The explanation could be that the scallop in the center image has a dark tone color which is similar to the background color. None of the negative class threshold images show a clear circular shape. More examples of gray-level threshold images of scallops can be seen in Figure (4.1).

Figure (3.10a,b) shows the mean feature image of the positive and negative sets, made with Otsu's method. The positive image is made from the augmented 50×50 pixel bounding window positive set and the negative image is made from the whole 50×50 pixel bounding window negative set. The circular shape of the scallop and the shadow from the AUV's strobe can clearly be seen on the positive image. The negative image is more or less uniformly gray, however the effect from the AUV's

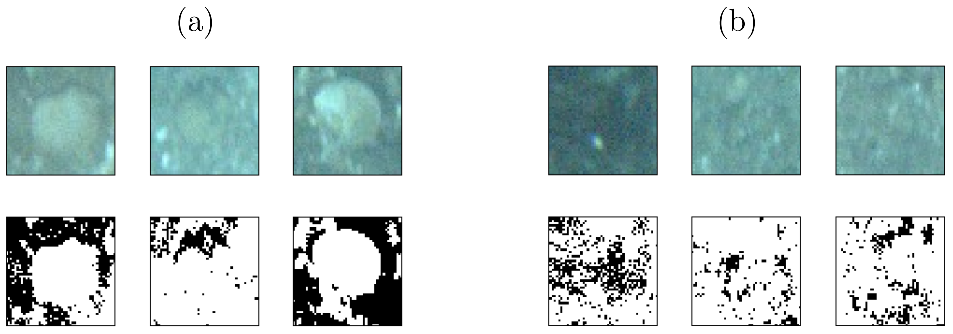


Figure 3.9: Positive class images and the resulting threshold images with Otsu's method (a), negative class images and resulting threshold images with Otsu's method (b).

strobe is also visible (the upper half of the image is darker than the lower half). Figure (3.10c) shows the mean feature image made from Warsha's data. It is very similar to the mean feature image from the positive set except that the white circle is slightly smaller since Warsha's data has a greater number of smaller scallops than the positive set.

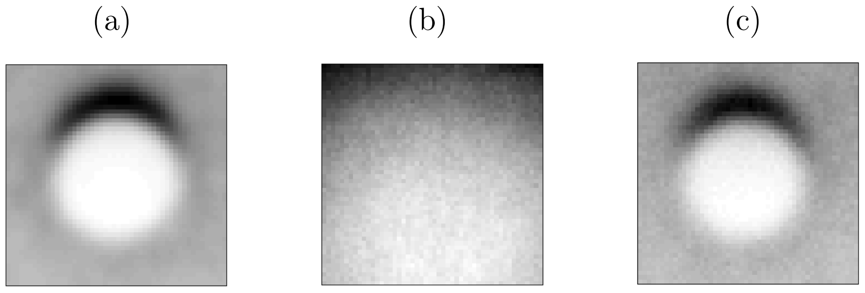


Figure 3.10: Mean feature images of the positive set (a), the negative set (b) and Warsha's positive set (c).

3.4.2. Color Histograms

Color histograms are histograms made from the three dimensional vector that represents color on an image. Chapelle et al. (1999) note that the color histogram technique is a very simple low level method and has shown good results in practice, particularly where feature extraction has to be simple and fast. The three dimensional vector that represents color on images can belong to different color spaces. Three widely used color spaces are:

- **RGB:** Colors are represented in terms of the three primary colors, red (R), green (G) and blue (B).
- **HSV:** Colors are represented in terms of hue (H), saturation (S) and intensity value (V) which are the three attributes that are perceived about color. The transformation between RGB and HSV color spaces is nonlinear (Phung et al., 2005).
- **CIE-Lab:** Is a color opponent space with colors represented in terms of lightness (L) and a and b for color-opponent dimension. The CIE-Lab color space is related to the RGB color space through highly nonlinear transformation (Phung et al., 2005).

The HSV color space is widely used in the literature, perhaps because it is attractive in theory. It separates the luminance component (V) from the color components (HS) and is therefore not as sensitive to illumination change. Chapelle et al. (1999), however, note that it does not seem to matter in practice if the HSV or the RGB color space is chosen.

After the color space is selected the number of bins per color component has to be decided. In Chapelle et al. (1999) they recommend calculating the histogram using the highest spatial resolution available. They use 16 bins per color component in their experiments, resulting in a length $16^3 = 4{,}096$ of the feature vector. The resulting color histogram is four dimensional (if all three color components are used) and is thus not possible to visualize directly.

3.4.3. Other Feature Extraction Methods

Other feature extraction methods were also tried for the scallop images. Edge-based methods, such as canny edge detection, gradient method and Haar-Wavelets, sound appealing since the scallop has a clear a circular shape. However, these methods did not work as well as expected. The reason might lie in the low image resolution of the AUV's images. Scallops cover at most 40 pixels horizontally and vertically and the edge detection methods did not detect the outer edges of the scallop in many cases. If the resolution and visibility of the images were better then these methods could prove to be a good feature extraction tool.

3.5. Summary

This chapter described the data collection survey in Breiðafjörður Fjord when images of scallops were acquired. The images were then used for making a training set for classification. Two feature extraction methods were then introduced, the Otsu's gray-level thresholding method which extracts the scallop's shape and the color histogram method which extracts features based on color. Finally, other feature extraction methods were considered which did not perform well in practice.

4. Classifiers and Scallop Detection

Predicting if an instance belongs to the positive scallop class, or the opposite, is a classification problem. Three different classification methods are discussed in this chapter. The idea of ROC curves and the AUC, which are used to compare the performance of classifiers, is also described. Then a description of how the classifiers are used by the scallop detectors follows.

4.1. Nearest Neighbor Algorithm

The nearest neighbor algorithm is a simple nonparametric classification method. The k-nearest neighbor algorithm (k-nn) is a special case of a nonparametric classification method. In the case of two classes (which is the case for the scallop detection) and when $k = 1$ it is possible to write the nearest neighbor algorithm as:

$$\begin{array}{ll} \text{Choose } \mathcal{C}_1 & \text{if } f(\mathbf{x}) = y'||\mathbf{x} - \mathbf{x}'||^{-1} > 0 \\ \text{Choose } \mathcal{C}_2 & \text{Otherwise} \end{array} \tag{4.1}$$

where $\mathbf{x}'$ is the nearest neighbor to the unlabeled instance $\mathbf{x}$, y' is the nearest neighbor's label (1 or -1), $f(\mathbf{x})$ is the *decision function* and $\mathcal{C}_i$ denotes class i. Unlabeled instance $\mathbf{x}$ is therefore classified as positive ($\mathcal{C}_1$) if the decision function returns a positive value (the nearest neighbor belongs to the positive class) and vice versa.

4.2. Distance to the Average Feature Image

The average feature images, shown in Figure (3.10), can be used for classification of scallops. The algorithm used is written as:

$$\begin{array}{ll} \text{Choose } \mathcal{C}_1 & \text{if } f(\mathbf{x}) = ||\mathbf{x} - \bar{\mathbf{x}}_n|| - ||\mathbf{x} - \bar{\mathbf{x}}_p|| > 0 \\ \text{Choose } \mathcal{C}_2 & \text{Otherwise} \end{array} \tag{4.2}$$

where $\bar{\mathbf{x}}_p$ is the average feature image vector for the positive class and $\bar{\mathbf{x}}_n$ is the average feature image vector for the negative class. $||\mathbf{x}-\bar{\mathbf{x}}_p||$ is the euclidean distance from the unlabeled instance $\mathbf{x}$ to the positive average feature image and $||\mathbf{x}-\bar{\mathbf{x}}_n||$ is the euclidean distance from the unlabeled instance to the negative average feature image. The instance $\mathbf{x}$ is therefore classified as positive ($f(\mathbf{x}) > 0$) if it is closer to the positive average feature image than to the negative average feature image.

4.3. Support Vector Machines

Support vector machine (SVM) is a maximum margin discriminant-based method that writes the model as a sum of the influences of a subset of training instances, called *support vectors*. These influences are given by so-called kernels that are application-specific measurements of similarity between training instances (Alpaydin, 2010). SVM is chosen as a classifier because it is widely used in classification literature (see e.g. Chapelle et al. (1999), Papageorgiou and Poggio (2000) and Phung et al. (2005)) and because of the excellent LIBSVM library which is available on the internet for free. Below is a very brief introduction to SVMs, a further introduction is shown in Appendix A.

Let $(\mathbf{x}_i, y_i)_{1\leq i\leq N}$ denote a set of training examples, $\mathbf{x}_i \in \mathbb{R}^d$, where d is the dimension of the input space. Each sample belongs to class $\mathcal{C}_1$ or $\mathcal{C}_2$ where $\mathbf{x}_i \in \mathcal{C}_1$ if $y_i = 1$ and $\mathbf{x}_i \in \mathcal{C}_2$ if $y_i = -1$.

The objective of an SVM is to define a hyperplane that separates the set of examples such that all the points belonging to the same class are on the same side of the hyperplane. The hyperplane is represented by a vector $\mathbf{w}$ and scalar b such that the closest point to the hyperplane has a distance of $1/||\mathbf{w}||$. There are a countless number of hyperplanes but the one with the largest distance to the closest point (input instance) is called the *optimal separating hyperplane* (OSH). In a case where the data is not linearly separable then the OSH is the solution of

$$\begin{aligned}
&\min\left(\frac{1}{2}||\mathbf{w}||^2 + C\sum_{i=1}^{N}\xi_i\right) \\
&\text{s.t.} \\
&y_i(\mathbf{w}\cdot\mathbf{x}_i + b) \geq 1 - \xi_i,\ i = 1,\ldots,N
\end{aligned} \tag{4.3}$$

where $(\xi_1,\ldots,\xi_N)$, $\xi_i \geq 0$, are slack variables and C is a penalty factor on errors, chosen by the user. Chapelle et al. (1999) note that when dealing with images the dimension of the input space is most often large compared to the number of training instances so the training data is usually linearly separable. In these cases, they argue, the value of the penalty parameter C will have little impact on the performance of the classifier.

The training optimization problem is equivalent to maximizing

$$\begin{aligned}
&W(\boldsymbol{\alpha}) = \sum_{i=1}^{N}\alpha_i - \frac{1}{2}\sum_{i,j}^{N}\alpha_i\alpha_j y_i y_j K(\mathbf{x}_i,\mathbf{x}_j) \\
&\text{s.t.} \\
&\alpha_i \geq 0,\ \sum_{i=1}^{N} y_i\alpha_i = 0
\end{aligned} \tag{4.4}$$

where $\boldsymbol{\alpha} = (\alpha_1,\ldots\alpha_N)$ are the N non negative Lagrange multipliers associated with the constraints in Equation (4.3), where the *support vectors* have $\alpha > 0$, and $K(\mathbf{x}_i,\mathbf{x}_j)$ is called a *kernel function.* The hyperplane *decision function* is written as

$$f(\mathbf{x}) \;=\; \sum_{i=1}^{N}\alpha_i y_i K(\mathbf{x}_i,\mathbf{x}) + b \tag{4.5}$$

where

$$\begin{array}{ll}
\text{Choose } \mathcal{C}_1 & \text{if } f(\mathbf{x}) > 0 \\
\text{Choose } \mathcal{C}_2 & \text{if } f(\mathbf{x}) < 0
\end{array} \tag{4.6}$$

4.3.1. Kernel Selection

The SVM architecture requires a selection of the kernel, in Equation (4.4), and the value of the parameter C. The selection of the kernel is important, and an inappropriate choice of kernel can lead to poor performance (Chapelle et al., 1999). There is no technique available to find the best suited kernel for any given problem. However, the following kernels are the most popular in the classification literature and are therefore used in this research:

- **Polynomial Kernel:** is written as

$$K(\mathbf{x}, \mathbf{y}) = (\mathbf{x} \cdot \mathbf{y} + 1)^p$$

  where p is the degree of the polynomial and is selected by the user. For example, when $p = 1$, then the kernel is linear and when $p = 3$ the kernel is a 3rd degree polynomial kernel.

- **Radial Basis Function (RBF):** is written as

$$K(\mathbf{x}, \mathbf{y}) = e^{\gamma ||\mathbf{x} - \mathbf{y}||_2^2}$$

  where the parameter γ defines the smoothness of the separating hyperplane and is chosen by the user. The RBF kernel is used as an example in (Chapelle et al., 1999) and (Papageorgiou and Poggio, 2000).

The parameters C and γ should be optimized using cross-validation. For the polynomial kernel only C has to be tuned so C can be searched on some interval with fixed increments. In the case of the RBF kernel, where both γ and C have to be tuned, the search should be done on a two dimensional grid (Alpaydin, 2010).

Since the scallop training data is unbalanced (≈ 1.9 negative instances for 1 positive instance), an extra weight parameter on the penalty parameter C in training is introduced. This will balance the penalty on errors between the positive and negative instances in training. This can be written as:

$$w_i C = \begin{cases} kC & \text{if } y_i = 1 \text{ (positive instance)} \\ C & \text{if } y_i = -1 \text{ (negative instance)} \end{cases}$$

where k decides how much extra penalty is added to the positive errors (e.g. $k = 1.9$).

4.4. k-Fold Cross-Validation

Cross-validation is a method which divides the dataset into two parts: a training set that is used for training a classifier and a validation set. The classifier is then used on the validation set and the classifier's performance is measured. Cross-validation is used, for example, in tuning the parameters of the SVM classifiers, C and γ.

In k-fold cross-validation, the dataset (training set), $\mathcal{X}$, is divided randomly into k parts of equal size, $\mathcal{X}_i,\ i = 1, 2, \ldots, k$. Then a training-validation pair is generated by keeping one of the k parts out as a validation set and the remaining $k - 1$ parts as a training set. This is done k times, leaving out another of the k parts each time. If $\mathcal{T}_i$ denotes the ith training set and $\mathcal{V}_i$ denotes the ith validation set then the following k training-validation pairs are obtained:

$$\begin{array}{ll} \mathcal{V}_1 = \mathcal{X}_1 & \mathcal{T}_1 = \mathcal{X}_2 \cup \mathcal{X}_3 \cup \cdots \cup \mathcal{X}_k \\ \mathcal{V}_2 = \mathcal{X}_2 & \mathcal{T}_1 = \mathcal{X}_1 \cup \mathcal{X}_3 \cup \cdots \cup \mathcal{X}_k \\ \vdots & \\ \mathcal{V}_k = \mathcal{X}_2 & \mathcal{T}_1 = \mathcal{X}_1 \cup \mathcal{X}_2 \cup \cdots \cup \mathcal{X}_{k-1} \end{array}$$

By doing this we allow small validation sets to keep the training set large. Each training and validation set should have the same proportion of negative and positive instances, which is called *stratification* (Alpaydin, 2010).

4.5. ROC Curve and The AUC

Receiver Operating Characteristics (ROC) curve is a graphical representation of the true positive rate (sensitivity) of a classifier versus the false positive rate , as the decision threshold is varied. ROC curves can be used to measure the performance of a classifier and are widely used in literature. ROC curves can be constructed by plotting the fraction of true positives out of the positives against the fraction of false positives out of the negatives. The AUC (area under the curve) is the area under this ROC curve. Ideally the classifier has an AUC of 1, and AUC values of different classifiers can be used to compare the general performance of the classifiers (Alpaydin, 2010).

For the complete scallop training set, 5-fold cross-validation is used to make five training-validation pairs, which are then used to make five different ROC curves for the same classifier. Since the positive set is augmented (see Section (3.2.1)) it is ensured that for each training-validation pair *all* nine instances, made from the same labeled scallop, belong either to the training set or the validation set. For each classifier, a single ROC curve is then constructed by taking the average of the five ROC curves made from the cross-validation.

4.5.1. The SVM Scallop Classifier

The training phase of an SVM scallop classifier is shown in Figure (4.1). The first step is to label the seabed images that are acquired by the AUV, which is described in Section (3.2). Then image editing is performed on the labeled images (if needed) prior to the feature extraction, which is discussed in Section (3.4). The training set will thus include feature vectors of each labeled instance. Finally the SVM classifier is trained with tuned parameters (found with cross-validation). The result is a classifier that predicts if an detection window includes a scallop or not.

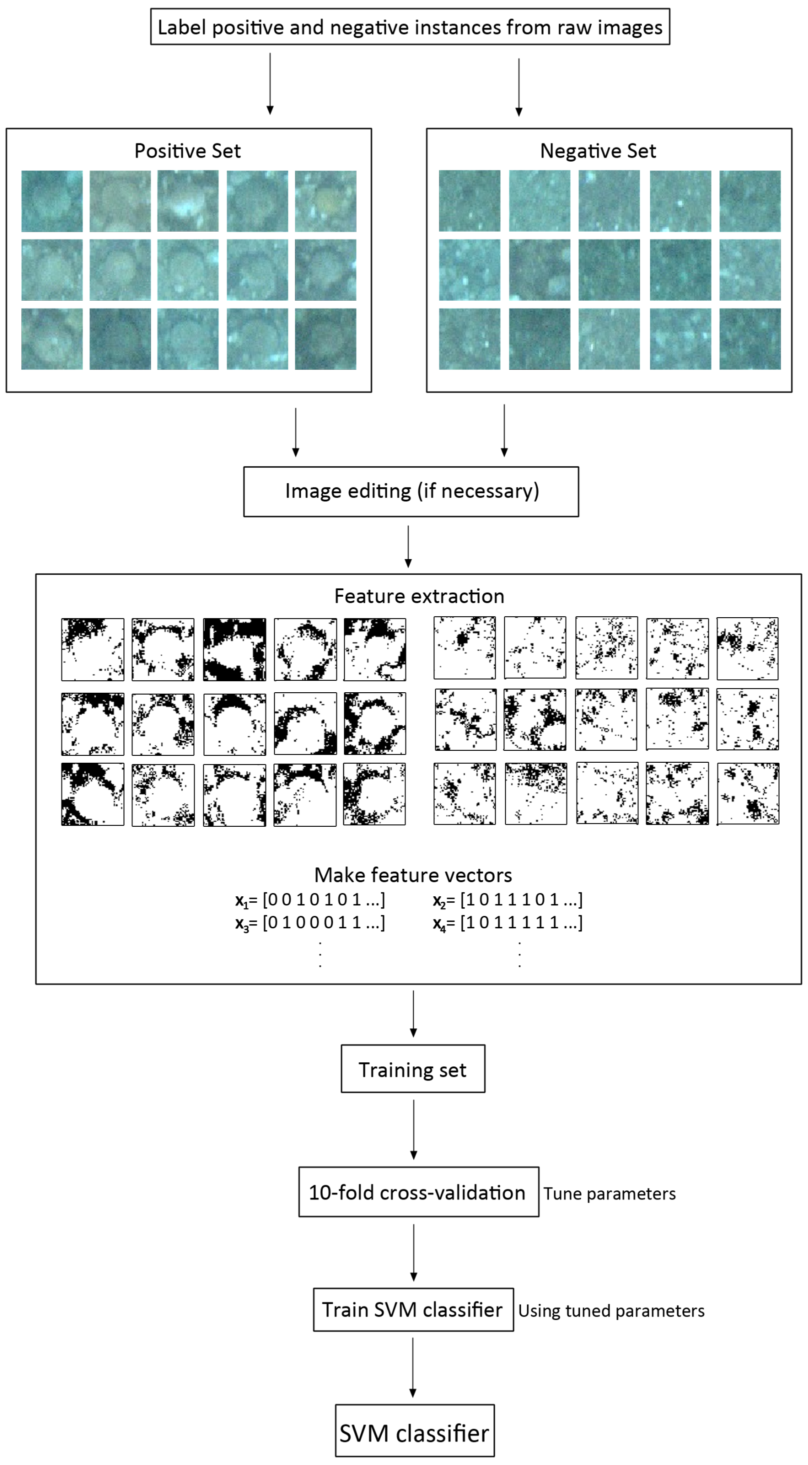


Figure 4.1: The training phase of an SVM scallop classifier (only gray-level threshold feature extraction shown).

4.6. Scallop Detection

The classifiers, which have been described in the previous sections, will predict if a detection window includes a scallop or not. Finding scallops, in images of the seabed, therefore requires that the images be scanned and then the classifier is used to predict if windows from the scan include scallops or not. One problem that often arises with image scanning techniques is overlapping detections (Hjelmas and Low, 2001). This can be dealt with by using a voting method that eliminates overlapped detections.

4.6.1. Image Scanning

Images of the seabed are scanned with detection windows (e.g. 50×50 pixels of size), which overlap according to the size of the jumps between detection windows. On image edges the center of the window has to be at least half the size of the detection window away from the edge (25 pixels in the case of a 50×50 pixel window), otherwise a part of the detection window would lie outside the image. This means that scallops which lie on the image edges will not be detected.

Figure (4.2) shows an example where the centers of the detection windows (indicated by red dots) would be situated in an image scan. Every second horizontal scan is shifted by half the length of the jump of the detection window (in pixels). For example, if the jump were set at 8 pixels then the x-coordinates of the detection window centers would switch between $(25, 33, 41, \dots)$ and $(29, 37, 45, \dots)$ in every second line. This is done to ensure greater scanning coverage of the image. This also ensures that true centers of scallops are never further than 4 pixels away (in case of 8 pixel jumps) from the center of a detection window.

The total number of detection windows for each 800×600 pixel image increases rapidly when the length of the jumps decrease. For example, with 8 pixel jumps the total number of scanning windows would be 6,489 and with 6 pixel jumps the total number would be 11,592. Therefore, when the jump is decreased, the computational effort needed to detect scallops will increase drastically.

4.6.2. Voting

Figure (4.3) shows an image of the seabed acquired from the AUV and centers of detection windows, indicated by green circles, that are classified as positive (using an SVM classifier). The green circles have a greater diameter for larger decision

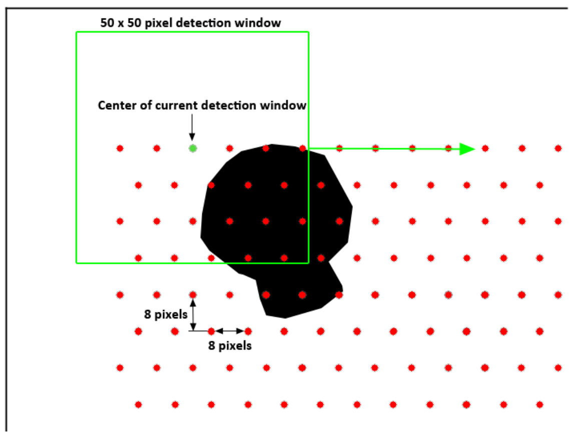

50 x 50 pixel detection window
Center of current detection window
8 pixels
8 pixels

Figure 4.2: Example of how image scanning is performed (showing 50×50 *pixel detection window with 8 pixel jumps). Centers of detection windows are indicated by red dots. Note how the dots are shifted by 4 pixels in every second horizontal line.*

values. The figure shows the problem of overlapping detections that can arise when image scanning is used. It also shows a number of positively classified detection windows that have low decision values, scattered around the image, which are in most cases false positives. In Hjelmas and Low (2001) a method (for face detection) is presented that deals with the problem of overlapping detections by using a thresholding heuristic. The method counts the number of detections in a small neighborhood surrounding the current location, and if the number is above a certain threshold a face is said to be present at the location. The *voting method* used in the scallop detection is based on this idea, with the addition of elimination of probable false positives by using a threshold on the decision values as well.

The voting method is shown under Algorithm 2. Firstly, all the decision values of the detection windows are calculated, to find which detection windows are classified as positive. Then, for each positive instance, the number of neighboring windows that are *also* classified as positive is found. Neighboring windows are the windows that are closest to the current window, which can be at most six in number. If the number of neighboring windows is equal to or above a threshold, K, and the decision value of the current window is above a threshold, α_1, then the coordinate of the current window is stored and a scallop is said to be at that location. Also if the number of neighboring windows is below K but the decision value is above another threshold, α_2, then the coordinate of the current window is as well stored. This is done to include windows that have relatively high decision values but not many neighbors (note that $\alpha_2 > \alpha_1$). Finally, if any of the stored windows are still cluttered inside a circle of 25 pixel radius then the window with the highest decision value is chosen and the other windows are discarded.

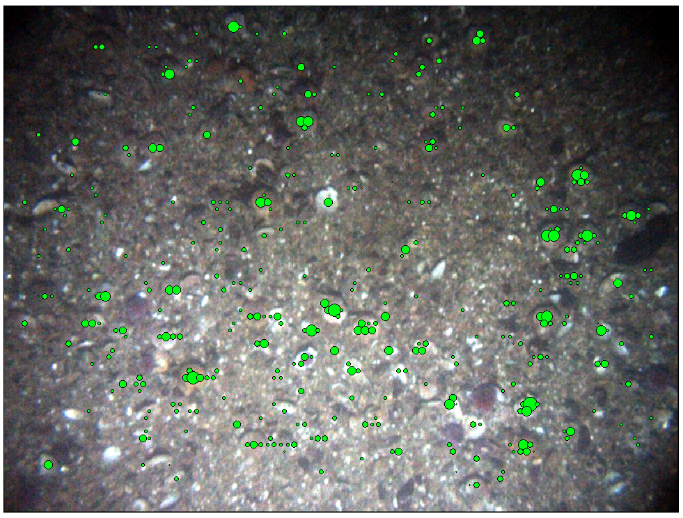


Figure 4.3: Image from the AUV of the seabed (intensity adjusted) and positive detection window centers, indicated by green circles.

The threshold values K, α_1 and α_2 are chosen by the user and can vary between different images of the seabed. In the case when $K = \alpha_1 = \alpha_2 = 0$ then all windows that are classified as positive are said to include a scallop.

Algorithm 2 Voting algorithm

```
1. Choose the following three thresholding parameters:
    • $\alpha_1$: Threshold on decision value.
    • $\alpha_2$: Threshold on decision value.
    • $K \leq 6$: Threshold on number of neighbors.
2. Calculate decision values for all detection windows: $f(\mathbf{x}_i)$, $i = 1, 2, \ldots, N$
   where $N$ is the total number of detection windows per image.
3. For all detection windows that are classified as positive ($f(\mathbf{x}_j)$, $\{j \in N | f(\mathbf{x}_j) > 0\}$) do the following:
    • Find the number of positively classified neighboring windows, $k$.
    • If $f(\mathbf{x}_j) > \alpha_1$ and $k \geq K$ then store the coordinate of the current window.
    • If $f(\mathbf{x}_j) > \alpha_2$ also store the coordinate of the current window.
4. If any of the stored coordinates are in a cluster (of 25 pixel radius) choose the
   window with largest decision value.
5. Discard all other detection windows that are classified as positive.
```

The complete scallop detection process, shown in Figure (4.4), is then:

1. Scan image of the seabed with a detection window where the jump of the window is set by the user.
2. Calculate decision values for all detection windows with a selected classifier.
3. Use the voting method to eliminate overlapping detections and probable false positives.
4. Show detected scallops.

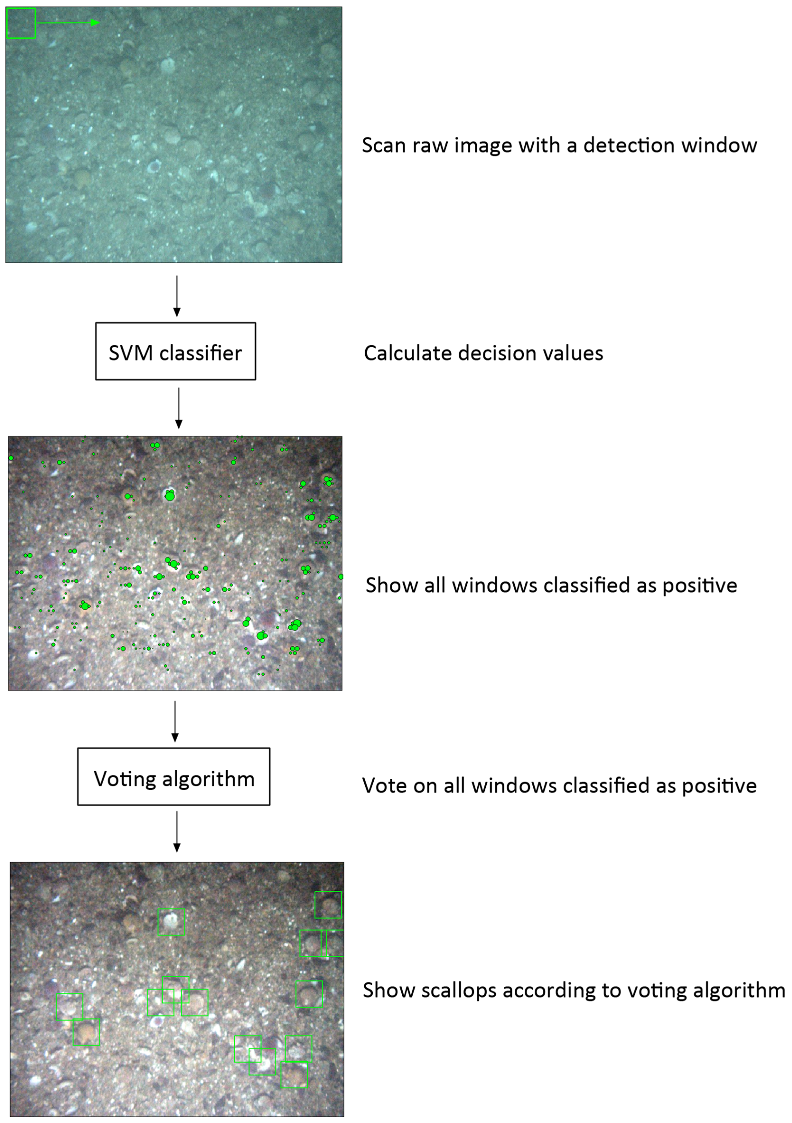

Scan raw image with a detection window
SVM classifier
Calculate decision values
Show all windows classified as positive
Voting algorithm
Vote on all windows classified as positive
Show scallops according to voting algorithm

Figure 4.4: The complete scallop detection process (shown for an SVM classifier).

4.6.3. Combination of Gray-level Threshold and Color Histogram Classifiers

The scallop detection process shown in Figure (4.4) only uses one classifier to calculate decision values for the detection windows. Figure (4.5) shows the detection process when a combination of color histogram and gray-level threshold classifiers is used. The seabed image is first scanned with a detection window and then the color histogram classifier is used to calculate the decision values of all the detection windows. The gray-level threshold classifier is then used on all detection windows that are classified as positive (areas of interest) by the color histogram classifier. Finally, the voting algorithm is used to eliminate all overlapping detections and probable false positives and the detected scallops are shown. Using the combination of the two classifiers could reduce the number of falsely detected scallops since the color histogram eliminates a large part of the image for the gray-level threshold classifier.

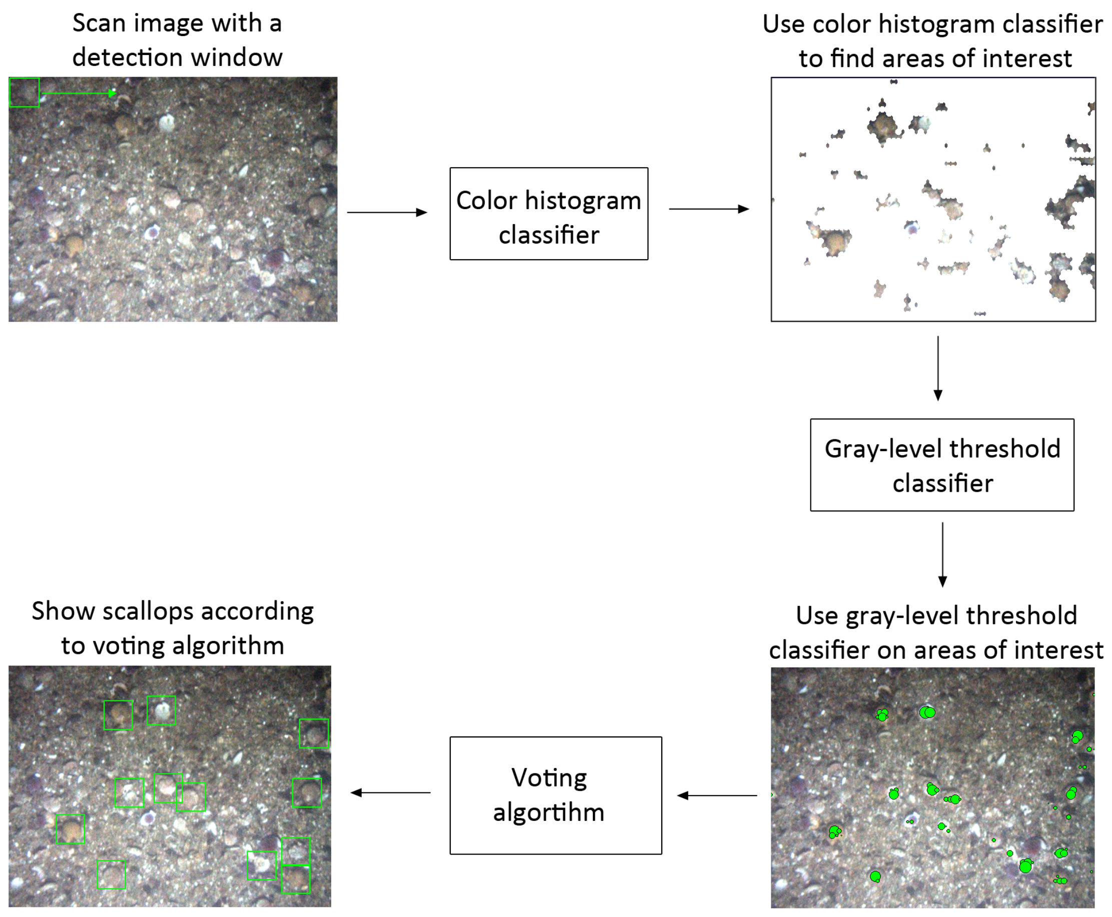

Scan image with a detection window
Color histogram classifier
Use color histogram classifier to find areas of interest
Gray-level threshold classifier
Use gray-level threshold classifier on areas of interest
Voting algortihm
Show scallops according to voting algorithm

Figure 4.5: The complete scallop detection process for the combination of gray-level threshold and color histogram classifiers.

4.7. Summary

This chapter has described the classification methods which are used for scallop detection and will later be used in Chapter 5. Two simple classification methods were first introduced, k-nn algorithm and the distance to the average feature image. Then the SVM classification method was described as were two common kernel functions. The idea of the ROC curve, how the ROC is calculated for the different classifiers, and the AUC were also discussed as well as how the AUC can be used to compare the classifiers' performance. The scallop detection process was shown, first by how the images of the seabed are scanned and then how a voting algorithm is used to eliminate overlapping detections and probable false positives. Finally, a scallop detection method that uses both gray-level threshold and color histogram classifiers was introduced.

5. Experiments

The methods described in the earlier chapters are the basis for the experiments. Firstly a comparison of the classifiers is conducted by using the gray-level feature extraction method on *raw* images, without any image editing. Then two other classification methods are tried: downsizing the gray-level feature image and the color-histogram feature extraction method. Finally the classifiers that show the best performance are used for testing to see how well they perform in scallop detection.

The experiments were conducted on an Apple computer with Dual Core 2.2 GHz Intel Pentium processor and 2GB RAM. MATLAB was used for all calculations. The LIBSVM (v3.11) toolbox for SVMs was used for the training of SVM classifiers.

5.1. Comparison of Classifiers for Gray-Level Threshold Feature

All classification methods used for comparison are trained using the feature image found with Otsu's method for finding a gray-level threshold. Performance of each classifier is then measured by how well Warsha's scallops are classified and by the AUC. The AUC is found by the method described in Section (4.5).

Figure (5.1) shows the ROC curves for the nearest neighbor and distance to the average feature image classifiers. The distance to the average feature image classifier uses the images shown in Figure (3.10). The nearest neighbor classifier only classifies based on the nearest neighbor ($k = 1$). It is interesting that the distance to the average feature classifier performs better than the nearest neighbor classifier in terms of cross validation accuracy, the AUC and accuracy of Warsha's scallops (results are shown in Table (5.1)). The distance to the average feature image classifier's accuracy of Warsha's scallops shows almost a 20% point improvement over the nearest neighbor classifier. This indicates that the distance to a centroid in feature space (the average feature image) is a better predictor than the distance to the nearest feature image. Another advantage of the distance to the average feature image classifier is that the computational time needed is much lower than for the nearest neighbor classifier since it only needs to calculate distance to the two average feature images whilst the nearest neighbor algorithm needs to calculate distance to all of the (23,681) training images.

Figure (5.2) shows the ROC curves for the SVM classifiers, a linear discriminant found with a linear kernel SVM, a 3rd degree polynomial kernel SVM and an RBF kernel SVM. All of the SVM classifiers outperform the nearest neighbor and distance to the average feature image classifiers in terms of all of the performance measurers. The 3rd degree polynomial SVM shows the highest accuracy using Warsha's scallops, 82.36%, while the AUC is the same for the RBF SVM and the 3rd degree polynomial SVM, 0.993. The cross validation accuracy is also very similar for the 3rd degree polynomial SVM and the RBF SVM, or around 96.4%. The high AUC and cross validation accuracy, for the 3rd degree polynomial and RBF SVM classifiers, suggest that the gray-level threshold feature is an acceptable descriptor of scallops and thus usable for scallop detection.

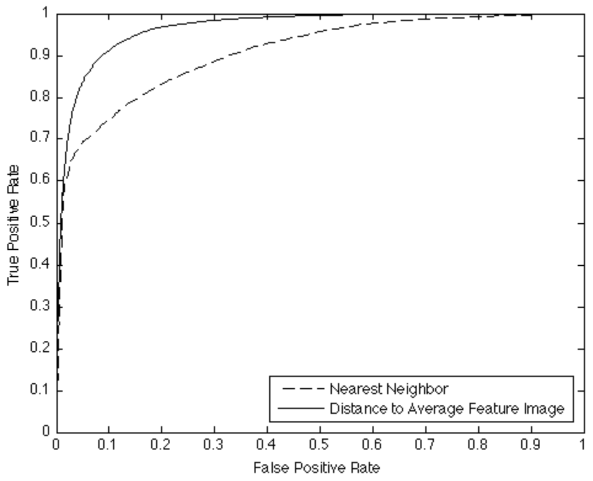


Figure 5.1: ROC curves for the nearest neighbor and distance to the average feature image classifiers.

Table 5.1: Performance of all gray-level thresholding classifiers on the 23,681 training instances and the 1,111 instances from Warsha.

Classifier	Warsha's Data Acc.	CV Acc.	CV Std.	AUC	#SVs
Nearest Neighbor	51.94%	87.10%	2.09%	0.908	-
Distance to the ave. feat. image	71.75%	91.10%	2.29%	0.967	-
Linear discriminant	75.88%	92.97%	0.60%	0.978	2148
SVM - 3rd degree polyn. kernel	82.36%	96.40%	0.84%	0.993	3036
SVM - RBF kernel	80.02%	96.44%	0.81%	0.993	3195

Figure (5.3) shows six scallops from Warsha's data that all classifiers labeled as negative and their gray-level threshold feature images. The feature images of these scallops could explain why about 20% of Warsha's scallops are labeled as negative

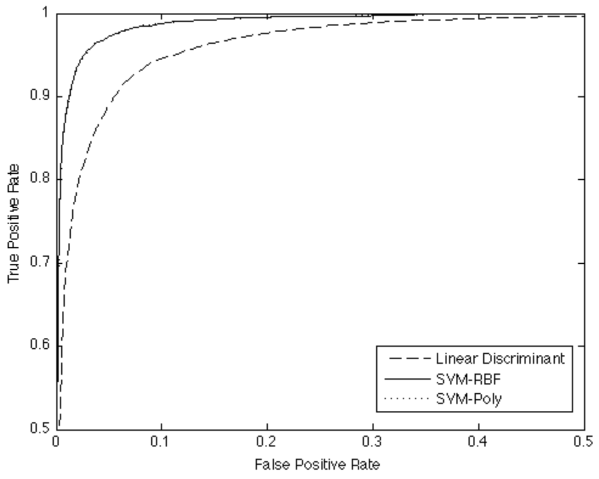
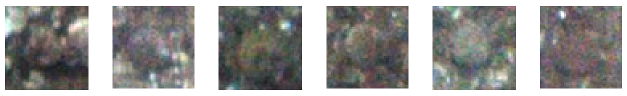


Figure 5.2: ROC curves for the two SVM classifiers and the linear discriminant. The 3rd degree polynomial kernel SVM and RBF kernel SVM show very similar curves.

by the best classifiers. The explanation as to why the AUC is very high while the accuracy of Warsha's instances is only about 80% could be that her instances of scallops are smaller, further away from the center of the bounding box and are darker than the training data. The second and fifth images (from the left) show scallops where the center of the bounding box is some distance from the center. The first feature image does not show the circular shape of the scallop, most likely because the brightness is too low. The fourth and the last images both show scallops that have color that blends with the background color, and hence the feature image does not show the circular shape.

(a)



(b)

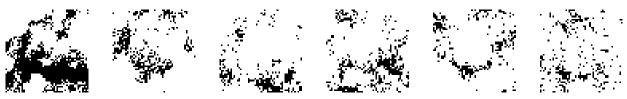


Figure 5.3: Example of scallops from Warsha's data (intensity adjusted) that all classifiers label as negative (a) and their gray-level threshold feature images (b).

In Figure (5.4) the image position of all true positives and true negatives using the

RBF kernel SVM classifier on Warsha's scallops is shown (the 3rd degree polynomial kernel SVM produces a very similar image). The green circles indicate true positives and the red circles the false negatives. The size of the circles is controlled by the decision values, large green circles have high positive decision values (correct classification) and the large red circles have high negative decision values (false classification). The figure shows that the false negatives are evenly spread around the image. This was expected since the gray-level threshold is calculated based on the gray-level intensity *only* inside the 50×50 pixel bounding window. Therefore it is independent of the brightness difference between different sections of the AUV's images.

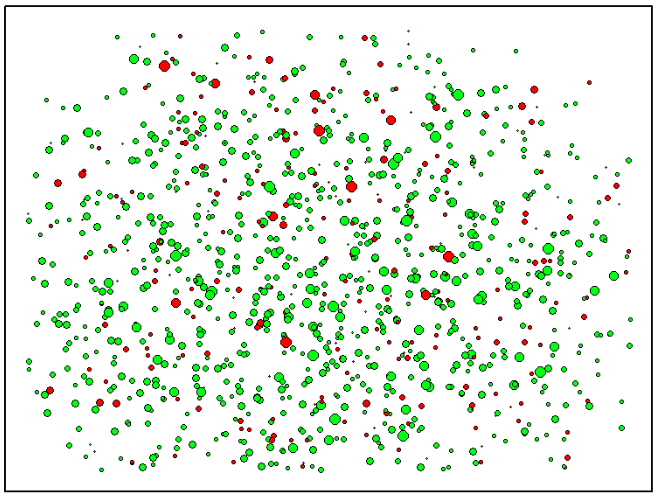


Figure 5.4: Image position of all true positives (green circles) and false negatives (red circles) using the RBF kernel SVM classifier on Warsha's scallops.

5.2. Enhanced Features

Until this point, only the gray-level feature found with Otsu's method has been considered. Other features will now be used to see if any improvement can be achieved, both in terms of performance of detecting scallops and computational time.

5.2.1. Downsize Image

The length of the feature vector made from the 50×50 pixel bounding window is 2,500. Since the number of support vectors for the SVM classifiers is quite high (see Table (5.1)) the computational effort needed when predicting if an instance is a scallop or not is also high. One way to make the feature vector smaller is to downsize the gray-level feature image and see if it affects the prediction rate of the classifier. For example, if the feature image is downsized to 25×25 pixels, the length of the feature vector for the downsized image would be 625, as shown in Figure (5.5).

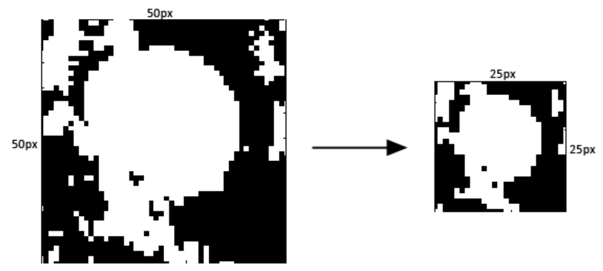


Figure 5.5: The feature image is downsized to 25×25 *pixels.*

Figure (5.6) shows the ROC curves for the SVM classifiers made with 25×25 pixel and 35×35 pixel feature images (3rd degree polynomial kernel), with AUC $= 0.767$ and AUC $= 0.737$ respectively. The classifiers' accuracy of Warsha's data is 80.65% and 82.99% respectively, which is surprisingly good. However, the ROC curves show that the false positive rates are already high for a rather low true positive rate for both of the classifiers. This could explain the high accuracy of Warsha's data (which only has positive instances) as the classifiers tend to label instances as positive when they are in fact negative. It is surprising that the classifier using the 25×25 pixel feature images performs slightly better than the 35×35 pixel classifier. The reason could be that the downsizing to 35×35 pixels (which is 70% of 50) loses more information of the original image than downsizing to 25×25 pixels (which is $1/2$ of 50).

The original image resolution is probably too low for making downsizing practical, as the ROC curves imply. However, if the image resolution were higher, then downsizing the gray-level feature image could be an effective way to save computational time without compromising the classifier's performance.

5.2.2. Color Histogram

The color histograms are made by using a 30×30 pixel window instead of the 50×50 pixel window, see Figure (5.7). This ensures that most pixels inside the window belong to scallops (in the case of positive instances) when the color histograms are constructed. When a 50×50 pixel window is used noise from background objects (such as white shell fragments) seems to affect the color histogram for positive instances and the performance is worse than for the 30×30 pixel window.

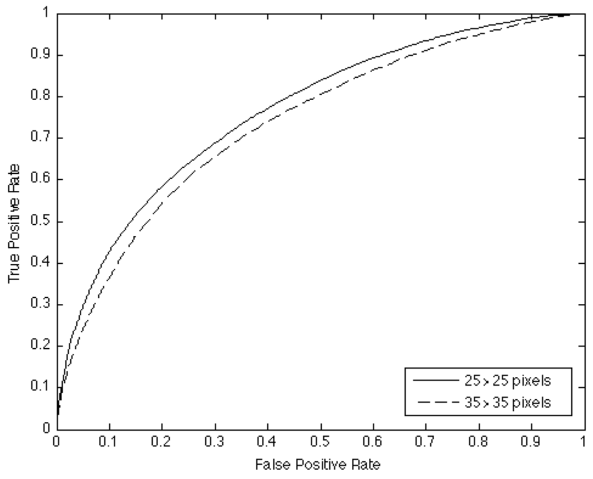


Figure 5.6: ROC curve for the 3rd degree polynomial kernel SVM for the downsized feature images.

An RBF kernel SVM was trained using the three color spaces in Section (3.4.2). The histogram bin size is set to 10, for the RGB and HSV color spaces, which results in a feature vector of a length of 1000. A larger bin size did not improve performance while a smaller bin size decreased performance. For the CIE-Lab color space only the color channels (a and b) are used and the bin size is set to 20, resulting in a feature vector of length 400.

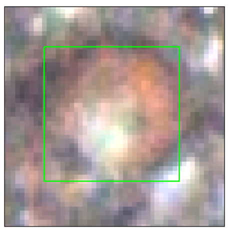


Figure 5.7: The window used for color histogram is 30×30 *pixels (green window in image).*

Both raw images and enhanced preprocessed images (with intensity adjustment and an average filter with $n = 2$) were tried for the color histogram. The color histograms made with enhanced images showed a better performance and only their results are shown. The reason as to why the enhanced images showed better performance is because the raw images have a compressed intensity range (see Section (2.4.1)) which results in most of the image pixels ending up in few bins of the color histogram.

Figure (5.8) shows the ROC curves for the color histogram classifiers made from the three color spaces. The ROC curves for the RGB and HSV classifiers are similar, which fits with Chapelle et al. (1999) results showing that the selection of color space does not seem to affect the performance greatly. The CIE-Lab classifier, only using the color channels, shows considerably lower AUC. The accuracy of all the color histogram classifiers using Warsha's data is low compared to the gray-level feature classifiers (see Table (5.2)) and the number of support vectors is much higher. The cross validation accuracy is similar for the RGB and HSV classifiers but lower for the CIE-Lab classifier.

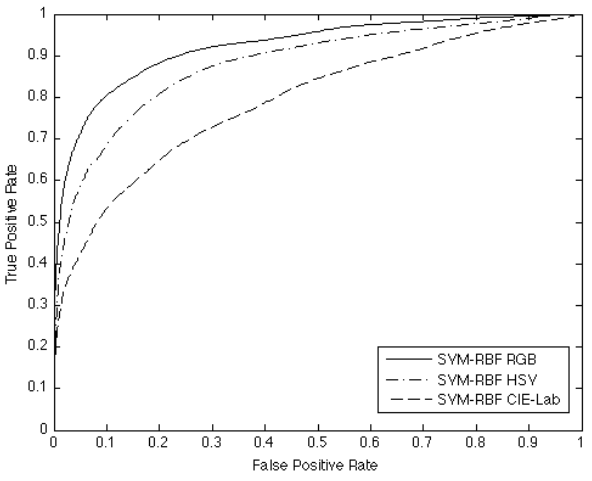


Figure 5.8: ROC curves for the color histogram classifiers.

Table 5.2: Performance of the color histogram classifiers on the 23,681 training instances and the 1,111 instances from Warsha.

Color Space	Warsha's Data Acc.	CV Acc.	CV Std.	AUC	#SVs
RGB	64.09%	86.38%	1.01%	0.923	10081
HSV	57.16%	82.37%	1.41%	0.881	10907
CIE-Lab	48.60%	76.25%	2.64%	0.793	14798

Figure (5.9) shows examples of scallops from Warsha's data which all of the color histogram classifiers classified as positive (upper row) and negative (lower row). All of the scallops which are classified correctly are light brown, red or orange while the scallops which are classified incorrectly are dark brownish which is similar to the background color.

In Figure (5.10) the RGB color histogram classifier is used to show all true positives and false negatives using Warsha's scallops. Compared to Figure (5.4), where the false negatives are evenly spread around the image, the density of the false negatives is higher on the image edges when the RGB color histogram classifier is used. Inside the blue box, in Figure (5.10), the accuracy is 74.8% whilst the accuracy outside the box is much lower, 45.9%. This indicates that the color histogram is not independent

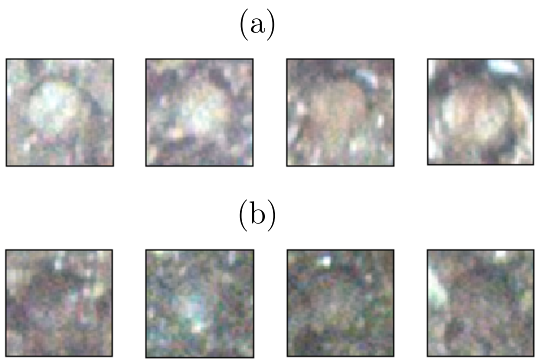


Figure 5.9: Examples of instances from Warsha's data (intensity adjusted) which all color histogram classifiers classified as positive (a), and negative (b).

of the image position (as expected) since different brightness affects the RGB values and thus the construction of the color histogram. The difference of the accuracy inside and outside the blue box could be explained by the following two factors: firstly the brightness inside the blue box is more or less even for all of the AUV's images (the AUV's strobe is effective inside the box) and secondly most of the training data is obtained from the same area of the images as the blue box, see Figure (3.5).

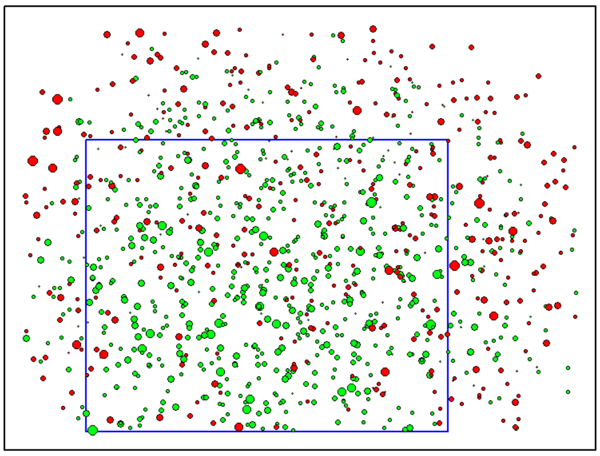


Figure 5.10: Image position of all true positives (green circles) and false negatives (red circles) using the RGB color-histogram classifier on Warsha's scallops.

Based on the results, shown in Figure (5.10), it is natural to ask if the performance of the classifier can be improved by using more than one color histogram classifier for different sections of the image. Figure (5.11) shows the average brightness of all training images (the average of the L channel from the CIE-Lab color space.) The effect of the strobe on the images can clearly be seen, with the highest brightness closest to the strobe. The two boxes (black and red) show the two image sections from where the training data is used to make two separate color histogram classifiers. The black box includes training instances that have the highest brightness and the red box training instances with lower brightness. All training instances outside the two boxes are omitted since the brightness is too low (88.8% of the positive training instances fit inside the two boxes).

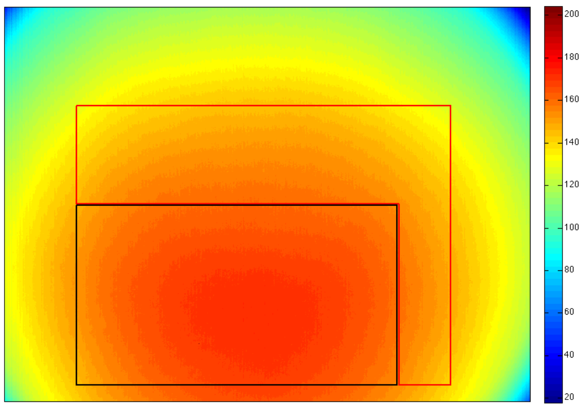


Figure 5.11: The average brightness of all training images and the boxes for the new color histogram classifiers.

Table (5.3) shows the performance of the two new color histogram classifiers (both RBF kernel SVMs). Both classifiers show higher cross validation accuracy and AUC than the original RGB color histogram classifier trained with the whole training set. The number of support vectors is also substantially lower for the new classifiers, which will improve the detection speed of the classifiers. The number of classifiers could even be increased such that the image would be divided into smaller sections for each classifier. This could further improve the classifiers' performance. However, this was not done since the number of positive training instances from many sections of the images is most likely too low for that to be viable.

Table 5.3: Performance of the new color histogram classifiers.

Color Space	**CV Acc.**	**CV Std.**	**AUC**	**#SVs**	**#Tr. inst.**
RGB (black box)	92.01%	1.26%	0.951	2375	11686
RGB (red box)	89.68%	0.91%	0.935	1911	8371

5.3. Tuning of Voting Parameters

Tuning of the voting parameters is conducted on 12 images from the AUV that were not used in training, but from the same data survey. The images all have an altitude below 2.2 m and different types of seabed, some with few scallops whilst others show benthos that are cluttered with objects. For the tuning three different scallop detectors are tried: two detectors that use the gray-level threshold feature RBF kernel SVM classifier from Section (5.1) and one detector that uses the combination of the gray-level threshold classifier and a color histogram feature RBF kernel SVM. Description of how the detectors work is shown in Section (4.6).

The gray-level feature RBF SVM is chosen because it showed the best performance in the earlier experiments (along with the 3rd degree polynomial SVM). One of the two detectors is set to have 8 pixel jumps (between detection windows) whilst the other has 6 pixel jumps and the size of the detection window is 50×50 pixels.

The combination detector uses the two color-histogram classifiers described in the pervious section. The color histogram classifier (made with the training instances which fit inside the black box in Figure (5.11)) is used on that same section of each tuning image. The same applies to the color histogram classifier made with the training instances from the red box. The size of the detection window is 30×30 pixels and the jump is set to 8 pixels. The gray-level classifier is then used on all the detection windows that are classified as positive by the color histogram classifiers (with a 50×50 pixel detection window).

The only part of each tuning image that is considered is the one which fits inside the two boxes in Figure (5.11). This is done for a fair comparison between the gray-level feature detectors and the combination detector. The voting parameters α_1 and α_2 are set such that they are a percentage of the maximum decision value for each image. Figure (5.12) explains the reason for this. Figure (5.12a) shows the tuning image with the lowest maximum decision value, $f(\mathbf{x}) = 1.2$. The visibility in the image is low and it is out of focus. Figure (5.12b) shows the tuning image with the highest maximum decision value, $f(\mathbf{x}) = 4.4$. The visibility is much higher than on the other image and it is easier to recognize scallops, which the decision values reflect. Detected scallops in Figure (5.12a) would therefore require lower decision values than scallops in Figure (5.12b).

The voting parameter K (number of positively classified neighboring detection windows) is fixed for each detector except when the number of positive detection windows is very high - then K is incremented by one. Figure (5.13) shows two tuning images with big difference in numbers of positive detection windows. The one on the left has only three positive detection windows but the one on the right has 151 positive detection windows (using the combination detector). The image on the

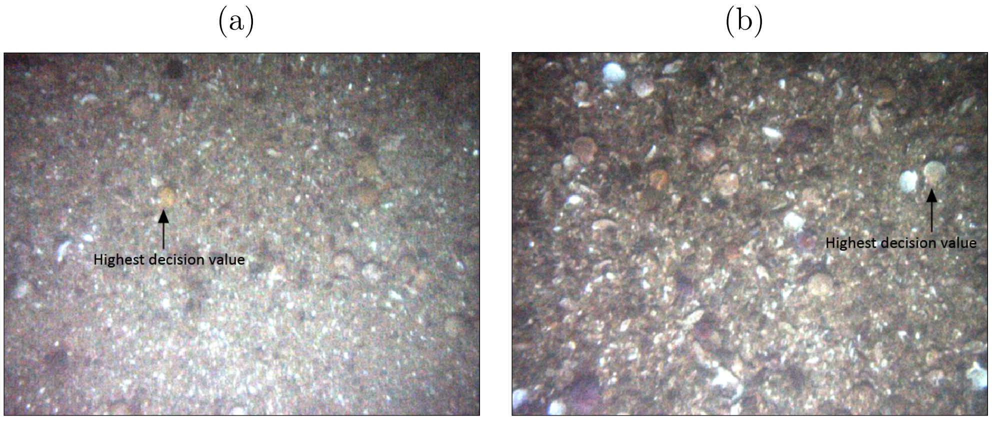


Figure 5.12: The tuning image with the lowest maximum decision value (a) and with the highest maximum decision value (b).

right will therefore need one more positively classified neighbor than the one on the left for a detected scallop by the voting algorithm.

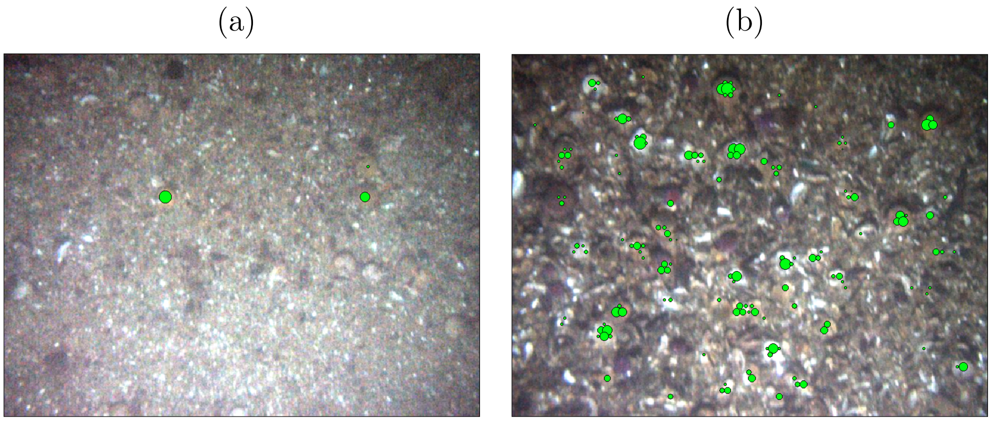


Figure 5.13: The tuning image with the fewest positive detection windows (a) and with the most positive detection windows (b).

The voting parameters, the percentage of α_1 and α_2, and K, are then tuned for each detector such that the number of scallops found is the highest whilst the number of falsely detected scallops is the lowest. This will result in a number of undetected scallops even though the classifiers classify them as positive. The voting parameters can be changed to include most (if not all) of the scallops in the tuning images but that would in turn produce greater number of falsely detected scallops.

Table (5.4) shows the results of the three scallop detectors used on the tuning images. All three detectors found a very similar number of scallops (TD), 77 - 79. However, the number of falsely detected scallops (FD) differs between the detectors. Interestingly the gray-level 6 pixel jump classifier has the highest number of false detections, 25, or 8 more than the gray-level 8 pixel jump classifier. A higher resolution than 8 pixel jumps in the scan therefore does not seem to improve the detection of scallops for the gray-level detectors. The combination detector had the lowest number of false detections, 7, which results in 0.50 false detections per image (for the 12 tuning images). This is explained by the fact that the color histogram classifier, used in the combination detector, decreases the area of the image that the gray-level classifier is then used on and thus decreases the possibility of false detections.

The average detection time for each image is also the lowest for the combination detector, 40 seconds, compared to 175 seconds for the 8 pixel jump gray-level detector and 290 seconds for the 6 pixel jump gray-level detector. The time saving of first using the color histogram classifier (which has fewer support vectors and a smaller feature vector then the gray-level classifier) and then the gray-level classifier is therefore significant. This is important when the detector is used in practice, which usually includes a large number of images to scan.

Table 5.4: Comparison of the three scallop detectors used in the tuning of the voting parameters.

Detector	TD	FD	FD/Image	FD/Scallop	Ave. det. time
Gray-level 6 px jumps	79	25	2.08	0.317	290 sec
Gray-level 8 px jumps	78	17	1.42	0.218	175 sec
Combination 8 px jumps	77	6	0.50	0.078	40 sec

The combination detector is therefore the winner of the three detectors, both in terms of the number of false detections and the average detection time. The detection results of the 12 tuning images using the combination detector are only shown and are found in Appendix B.

5.3.1. True and False Detections in Tuning Images

Table (5.4) shows that the number of true detections is very similar for all three detectors. However, this does not mean that they all detect the same scallops in the same image. Figure (5.14) shows detected scallops for two of the tuning images according to all three detectors. The most obvious scallops are detected by all three detectors but other scallops (or false detections) are divided by the detectors. The scallops that are detected by the gray-level detectors but not the combination detector have been eliminated by the color histogram classifier of the combination detector. The scallops that only the combination detector detects are the result of the lower voting parameters of the combination detector, which are allowed by the elimination of unwanted area by the color histogram classifier.

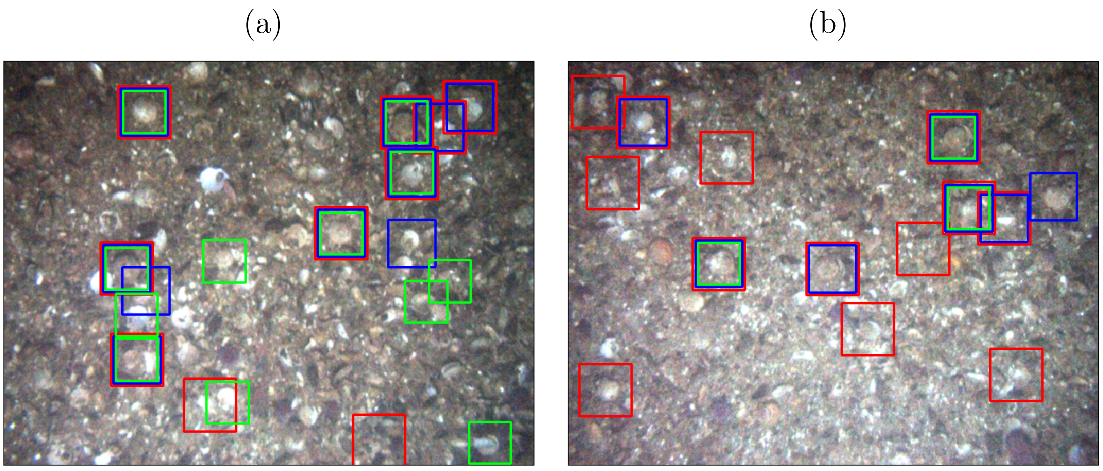


Figure 5.14: Detected scallops by the three detectors for two of the tuning images. Red boxes belong to the 6 pixel gray-level detector, blue boxes to the 8 pixel gray-level detector and green boxes to the combination detector.

Figure (5.15) shows the six falsely detected scallops, using the combination detector, and their gray-level feature images. The images could explain why they are detected as scallops. The feature images show a great resemblance to the feature images of the positive set that was used to train the gray-level classifier. The color of the objects in the images is also similar to the color of many of the scallops used for training. Therefore, the detectors that were used in tuning will always result in some false positives. The low quality of the images from the AUV and the complexity of the objects in the images may contribute to this fact.

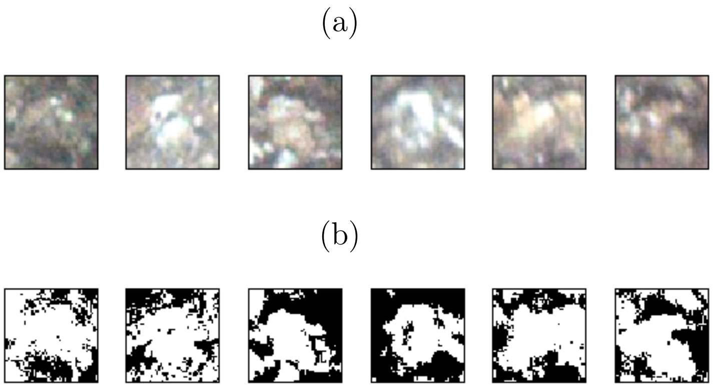


Figure 5.15: False detections from the tuning images (a), and their gray-level feature images (b).

5.3.2. Undetected Scallops in Tuning Images

There are few obvious instances of scallops in the tuning images that none of the detectors detected as scallops. They are all classified as positive by the gray-level classifier but the voting algorithm discards them. The first three scallops in Figure (5.16) are not detected by any of the three detectors. The feature image of the first two does not show the obvious scallop shape. The shape is clearer in the third image but the decision value is rather low. The last image is detected by the two gray-level detectors but the color histogram classifier of the combination detector classifies it as negative.

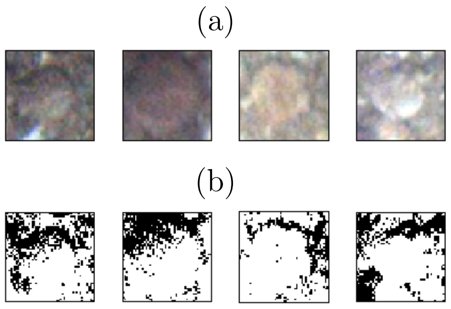


Figure 5.16: Undetected scallops in the tuning images (a), and their gray-level feature images (b).

5.4. Testing of The Combination Detector

The combination detector showed the best performance of the detectors used on the tuning images. However, since the tuning images were used to tune the voting parameters of the detector, a test set is made to confirm the performance of the detector. For this test 20 images (not used in training) are selected randomly from the data survey, with an altitude of < 2.20 m and the combination detector is used with the same voting parameters as in the tuning. Images with very low visibility were not used.

The results of the test images are shown in Appendix C. The performance is not as good as in the tuning, with 81 true detections of scallops and 15 false detections, or 1 falsely detected scallop for every 5.4 that are correctly detected. For some of the detections it was hard to decide if they were in fact scallops or not, as can be seen in Figure (5.17). The number of true detections of scallops therefore only includes definite detections. Figure (5.17) shows all of the false detections from the testing and their gray-level feature images. As in the tuning most of the gray-level feature images of the false detections bear resemblance to the gray-level feature images of the positive set that was used in training.

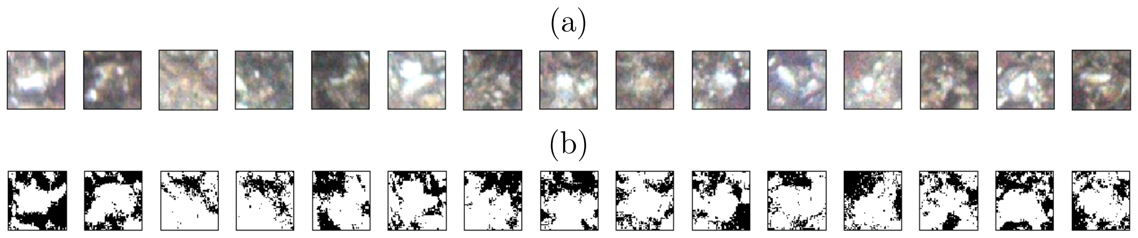


Figure 5.17: False detections from the second test images (a), and their gray-level feature images (b).

There are also some obvious undetected scallops which are, as in the tuning, classified as positive by the gray level classifier but either discarded by the voting algorithm or by the color histogram classifier.

There are many possible reasons as to why the detector performs worse on the test images than in the tuning. A higher number of images could have been needed in the tuning of the voting parameters to get a better generalization of the detector since the variability (in terms of altitude, visibility and brightness) of the AUV's images is quite high. The quality of the images in the testing was also lower than for the images in the tuning, which could have affected the detector's performance.

Based on the results of the testing, the combination detector achieves about an 80% success rate in scallop detection, without counting the undetected scallops. Even by using the combination of the color histogram and gray-level classifiers, a lower rate of false detections was not achieved. Most of the obvious scallop instances were, however, found by the detector which is promising. This is perhaps an acceptable performance bearing in mind the difficulty of the problem because of the low quality of the AUV's images and the variable background objects on the seabed which resemble scallops both in terms of shape and color.

5.5. Summary

This chapter started with the performance comparison of the scallop classifiers that were described in earlier chapters. Firstly the gray-level classifiers were tested and the RBF kernel SVM and 3rd degree polynomial kernel SVM classifiers showed the best performance with about 80% accuracy of Warsha's data. The downsizing of the gray-level feature image was then tested but the increase in calculation speed did not compensate for the decrease in performance. The color histogram classifier was also tested and the results showed that the performance was dependent of the image position because of the brightness variability of the AUV's images. Two new color histogram classifiers were then made with training data from different parts of the training images based on brightness intensity which showed a better performance. The tuning of voting parameters was then performed by three detectors using the most prominent classifiers. The combination detector, that uses the combination of the gray-level classifier and the two color histogram classifiers, showed the best results with about 0.5 false detections per image. Finally, testing was conducted on 20 test images using the combination detector. The results were not as good as in the tuning with about 80% detection accuracy of scallops, but most of the obvious scallops were detected.

6. Conclusions

The combination detector, using the mixture of the gray-level and color histogram classifiers, finds most of the obvious instances of scallops. However, about 1 in 6 detections are false detections based on the results of the test images. There are also few undetected scallops that are discarded either by the color histogram classifier or by the voting algorithm. The results are thought to be satisfactory since finding scallops in the low quality images from the AUV is a demanding problem.

The detection method described in this thesis will be useful in practice. It will not eliminate the need for a user to scan the images, as the user will have to confirm the classifier results, remove false detections and find the few undetected scallops. But the method will save time because most scallops will be highlighted with a bounding window. When the training set was constructed it was very difficult to find even the most obvious instances of scallops. Multiple observations of every image were needed to be sure of having found all scallops in the images.

The camera calibration provides an easy method to estimate scallop sizes, with lens distortion accounted for. The lens distortion of the AUV's camera is minimal, which is not surprising since it is manufactured for underwater use. The size estimation was validated with an image of a pattern with known lengths. However, size estimation has to be done manually. When scallops have been found by the detection method, with false detections and undetected scallops accounted for, the user has to manually select the two pixel points in the image on the top and bottom of the scallop and then use the size estimation method to convert the pixel distance to metric distance.

The University of Iceland has already bought a substantially more sophisticated camera for the AUV, which has greater resolution, and intends to add a more powerful light source to improve brightness in the images. The methods in this thesis should provide a good framework for the detection of scallops and size estimation in images from the new camera, especially if the quality and brightness of images will be greater.

6.1. Further Improvement and Methods

The scallop detection process that was described in this thesis is by no means the only possible one. In every step of the process a different method could have been used. However, the methods chosen are sufficiently accurate to be useful. There is also room for improvement of the combination detector that was used in the final testing. Some proposals for improvement and other methods that could have been effective are presented below.

Improvements of the combination detector:

- The making of the training set could be done in a more sophisticated way. For example, the training set could be sorted into groups based on color type, size, orientation and/or position in the image (for the color histogram classifier) and a separate classifier made for every group.

- Image editing could be used for preprocessing to improve the gray-level thresholding.

- In Figure (5.14) it can be seen that some of detected scallops are in fact dead (white shells). Another color histogram classifier could be made to separate dead scallops from alive ones.

- The calculation speed could be increased since a big part of the bounding window is probably not important for the classifier (the center of the bounding window matters the most).

- The tuning of voting parameters could be done on a greater number of images. A cost function could also be made to do the tuning automatically, which penalizes for false positive detections and rewards for true detections.

- Ideas from active learning could be used to reduce false positives and negatives, see Settles (2009). The classifiers made from the labeled training instances are first used on unlabeled detection windows. The windows that are most difficult to classify (closest to the OSH) are then fed to an "oracle" (e.g. human annotator) and labeled accordingly. Finally, these new labeled instances are added to the training set and the process repeated until a satisfactory performance is achieved.

Other methods that could be effective:

- Other feature extraction methods, such as Haar-wavelets and gradient methods, are widely used in face detection literature (see Hjelmas and Low (2001)) and could prove to be effective in finding scallops.

- A huge number of classification methods exist that have been proven to perform in face detection. Neural networks, statistical approaches and linear subspace models to name a few (Hjelmas and Low, 2001).

The method used in Algorithm 1 to estimate "real" size in images could rather easily be expanded to include the pitch and roll of the AUV, which would improve the size estimation (especially if the AUV is tilted by many degrees). An automatic method of finding the diameter of scallops, such as Hough transformation for finding circles, could also be used to make the size estimation process entirely automatic (provided that the scallop detection is effective).

Bibliography

Alpaydin, E. (2010). *Introduction to Machine Learning*, volume 56 of *Adaptive Computation and Machine Learning.* MIT Press.

Bouguet, J.-Y. and Perona, P. (1998). Closed-Form Camera Calibration in Dual-Space Geometry. *European Conference on Computer Vision (ECCV).*

Chapelle, O., Haffner, P., and Vapnik, V. (1999). SVMs for Histogram-Based Image Classification. Technical Report 3.

Farid, H. (2001). Blind Inverse Gamma Correction. *IEEE Transactions on Image Processing*, 10(10):1428–1433.

Gallager, S. M., York, A. D., Taylor, R., Vine, N., Howland, J., Lerner, S., and Bolles, K. (2009). Summary of HabCam Activities Related to the NMFS/NEFSC Sea Scallop Survey of the Nantucket Lightship Closed Area and Yellowtail Flounder Observations in 2009. Technical report, HabCam Group.

Gavia, T. (2011). *Gavia AUV User Manual.* Teledyne Gavia, Vesturvör 29, 200 Kópavogur, Iceland.

Heikkila, J. and Silven, O. (1997). A Four-step Camera Calibration Procedure with Implicit Image Correction. *Computer Vision and Pattern Recognition, IEEE Computer Society Conference on*, 0:1106.

Hjelmas, E. and Low, B. K. (2001). Face Detection: A Survey. *Computer Vision and Image Understanding*, pages 236–274.

Jónasson, J. P. (2007). Hörpudiskurinn í Breiðafirði - Rannsóknir og ástand stofnsins. Technical report, Háskólasetur Snæfellsnes.

Liu, Y., Wu, Y., Wu, M., and Hu, X. (2004). Planar Vanishing Points Based Camera Calibration. In *Proceedings of the Third International Conference on Image and Graphics*, ICIG '04, pages 460–463, Washington, DC, USA. IEEE Computer Society.

Otsu, N. (1979). A Threshold Selection Method From Gray-level Histograms. *Ieee Transactions On Systems Man And Cybernetics*, 9(1):62–66.

Papageorgiou, C. and Poggio, T. (2000). A Trainable System for Object Detection. *International Journal of Computer Vision*, 38(1):15–33.

Phung, S. L., Bouzerdoum, A., and Chai, D. (2005). Skin Segmentation Using Color Pixel Classification: Analysis and Comparison. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(1):148–154.

Research, P. G. (2005). *PGR Scorpion Technical Reference Manual.* Point Grey Research Inc., 8866 Hudson Street, Vancuver, BC, Canada.

Rosin, P. and Ellis, T. (1995). Image Difference Threshold Strategies and Shadow Detection. In *in Proc. British Machine Vision Conf*, pages 347–356. BMVA Press.

Settles, B. (2009). Active learning literature survey. Computer Sciences Technical Report 1648, University of Wisconsin–Madison.

Sezgin, M. and Sankur, B. (2004). Survey Over Image Thresholding Techniques and Quantitative Performance Evaluation. *Journal of Electronic Imaging*, 13(1):146–168.

Weng, J., Cohen, P., and Herniou, M. (1992). Camera Calibration With Distortion Models and Accuracy evaluation. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 14(10):965–980.

Zhang, Z. (2000). A Flexible New Technique for Camera Calibration. *IEEE Trans. Pattern Anal. Mach. Intell.*, 22:1330–1334.

Zhao, T. and Nevatia, R. (2003). Car Detection in Low Resolution Aerial Image. *Proceedings Eighth IEEE International Conference on Computer Vision ICCV 2001*, 21(8):693–703.

A. Support Vector Machines

A.1. Optimal Separating Hyperplane and The Kernel Trick

Let $(\mathbf{x}_i, y_i)_{1\leq i\leq N}$ denote a set of training examples, $\mathbf{x}_i \in \mathbb{R}^d$, where d is the dimension of the input space. Each sample belongs to class $\mathcal{C}_1$ or $\mathcal{C}_2$ where $\mathbf{x}_i \in \mathcal{C}_1$ if $y_i = 1$ and $\mathbf{x}_i \in \mathcal{C}_2$ if $y_i = -1$. The objective is to define a hyperplane that separates the set of examples such that all the points belonging to the same class are on the same side of the hyperplane. This is equivalent to finding the vector $\mathbf{w}$ and scalar b so that

$$y_i(\mathbf{w} \cdot \mathbf{x}_i + b) > 1, \;\; i = 1, \ldots, N. \tag{A.1}$$

The closest point to the hyperplane then has a distance of $1/||\mathbf{w}||$. There is a countless number of hyperplanes but the one with the largest distance to the closest point (input instance) is called the *optimal separating hyperplane* (OSH). Finding the OSH is equivalent to minimizing $||\mathbf{w}||^2$ under the constraint in Equation (A.1).

If the data are not linearly separable then Equation (A.1) will not work. In this case the hyperplane that incurs the least error is found (Alpaydin, 2010). Let us define slack variables $(\xi_1, \ldots, \xi_N)$ with $\xi_i \geq 0$ such that

$$y_i(\mathbf{w} \cdot \mathbf{x}_i + b) \geq 1 - \xi_i, \; i = 1, \ldots, N \tag{A.2}$$

to relax the Equation in (A.1). When $\xi_i = 0$ there is no problem with $\mathbf{x}_i$, it is correctly classified. If $0 < \xi_i < 1$, then $\mathbf{x}_i$ is correctly classified but in the margin and if $\xi_i \geq 1$ then $\mathbf{x}_i$ is misclassified. The generalized OSH is then the solution of

$$\min \left(\frac{1}{2}||\mathbf{w}||^2 + C \sum_{i=1}^{N} \xi_i \right) \tag{A.3}$$

subject to the constraints in Equation (A.2) and $\xi_i \geq 0$. The first term is the same as in the separable case, to control the learning capacity. The second term is a penalty term that penalizes for misclassified points. The parameter C is a penalty factor chosen by the user. The larger C is, the more penalty is assigned to errors. Chapelle et al. (1999) note that when dealing with images the dimension of the input space is most often large compared to the number of training instances so the training data is usually linearly separable. In these cases the value of the penalty parameter C will have little impact on the performance of the classifier.

Assuming that $||\mathbf{w}||^2$ is convex it is possible to minimize it under the linear constraints in Equation (A.2) using Lagrange multipliers. Let $\boldsymbol{\alpha} = (\alpha_1, \dots \alpha_N)$ be the N non negative Lagrange multipliers associated with the constraints in Equation (A.2). Then the optimization problem is equivalent to maximizing

$$W(\boldsymbol{\alpha}) = \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i,j}^{N} \alpha_i \alpha_j y_i y_j \mathbf{x}_i \cdot \mathbf{x}_j \tag{A.4}$$

subject to the constraints

$$\alpha_i \geq 0, \ \forall i$$
$$\sum_{i=1}^{N} y_i \alpha_i = 0.$$

This problem can be solved using a standard quadratic optimization method (Alpaydin, 2010). If the solution vector $\boldsymbol{\alpha}^0 = (\alpha_1^0, \dots \alpha_N^0)$ of the maximization problem in Equation (A.4) has been found then the OSH $(\mathbf{w}_0, b_0)$ can be expressed as

$$\mathbf{w}_0 = \sum_{i=1}^{N} \alpha_i^0 y_i \mathbf{x}_i. \tag{A.5}$$

The points for which $\alpha_i^0 > 0$ satisfy Equation (A.1) with equality are called the *support vectors*. The OSH is constructed in a high-dimensional *feature-space*, where the input data is mapped into this feature-space through some nonlinear mapping. If $\mathbf{x}$ is replaced by its mapping in the feature space, $\Phi(\mathbf{x})$, then it is possible to write Equation (A.4) as

$$W(\boldsymbol{\alpha}) = \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{N} \alpha_i \alpha_j y_i y_j \Phi(\mathbf{x}_i) \cdot \Phi(\mathbf{x}_j).$$

The dot product of the mapping into feature space can be replaced by a *kernel function*, $K(\mathbf{x}_i, \mathbf{x}_j) = \Phi(\mathbf{x}_i) \cdot \Phi(\mathbf{x}_j)$. Then only K is needed in the training algorithm and the mapping Φ is never explicitly used. This is called the *kernel trick* (Alpaydin, 2010). The training algorithm in Equation (A.4) is then replaced by

$$W(\boldsymbol{\alpha}) = \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i,j}^{N} \alpha_i \alpha_j y_i y_j K(\mathbf{x}_i, \mathbf{x}_j). \tag{A.6}$$

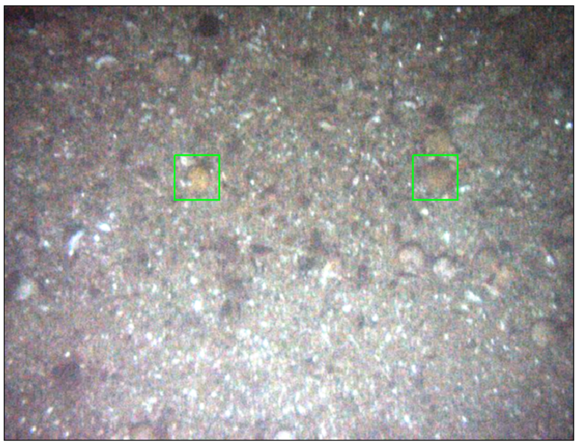
The hyperplane *decision function* can then be written as

$$f(\mathbf{x}) \quad = \quad sgn\left(\sum_{i=1}^{N} \alpha_i y_i K(\mathbf{x}_i, \mathbf{x}_j) + b \right). \tag{A.7}$$

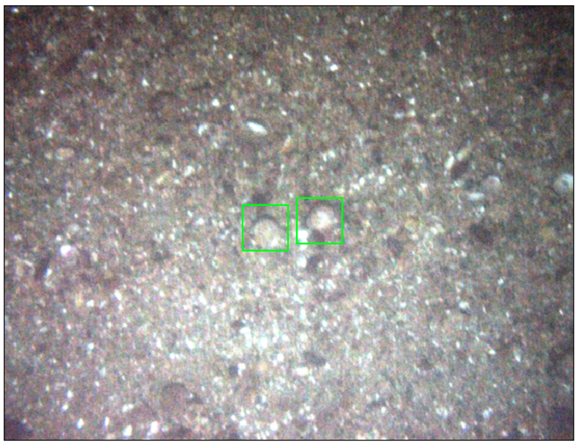
B. Results of The Tuning Images

Results are only shown for the combination detector. All images are intensity adjusted. True detections have green boxes and false detections have red boxes. Undetected scallops are left to the reader to determine.

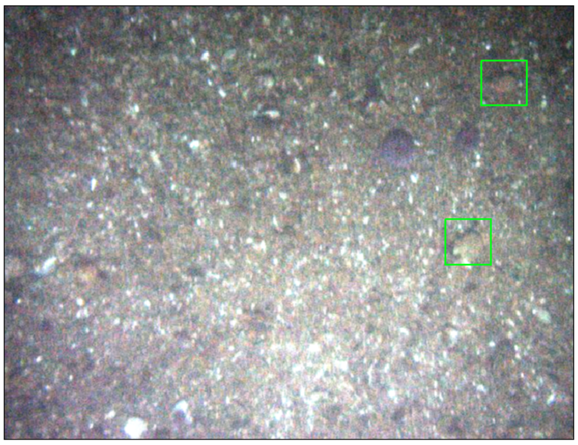
Altitude 1.99 m



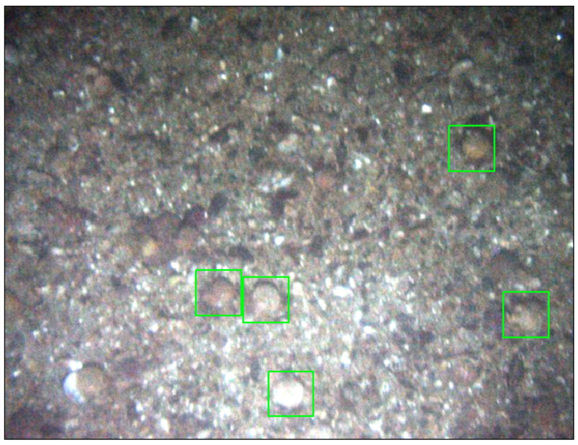
Altitude 1.98 m



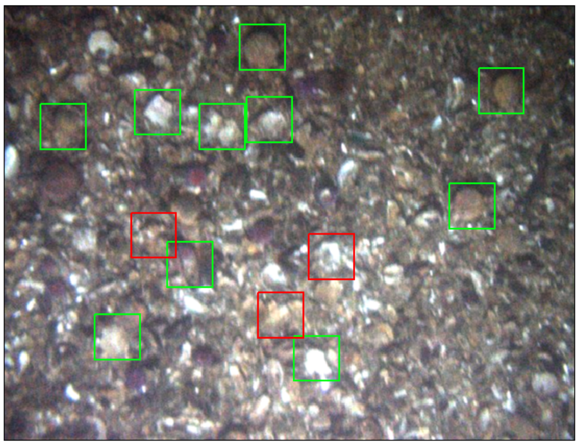
Altitude 1.95 m



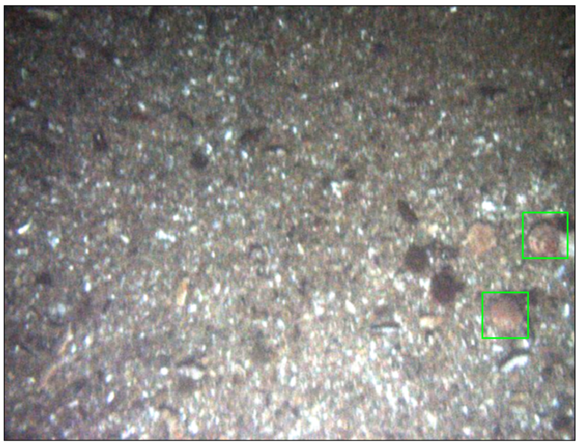
Altitude 1.86 m



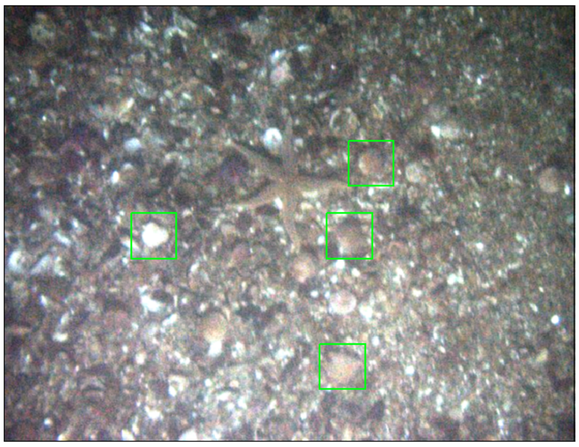
Altitude 1.62 m



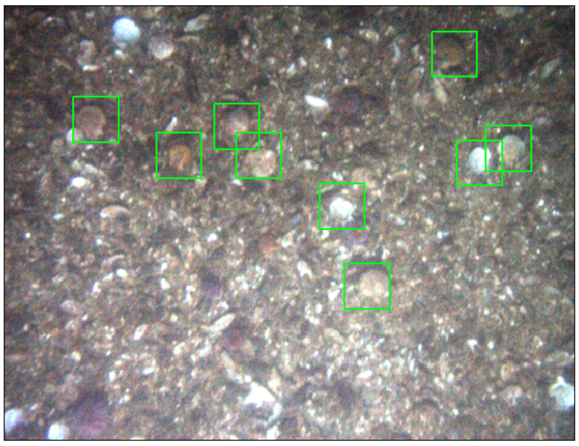
Altitude 1.61 m



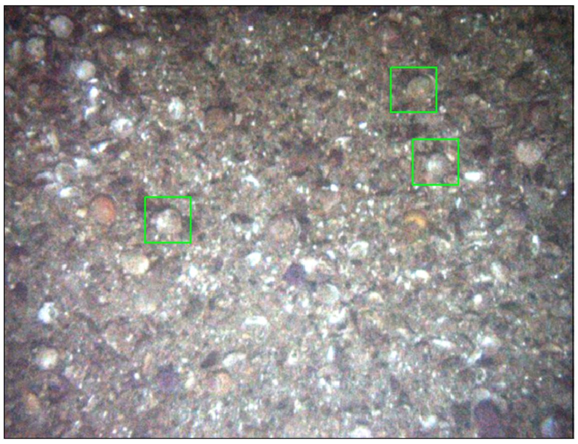
Altitude 1.89 m



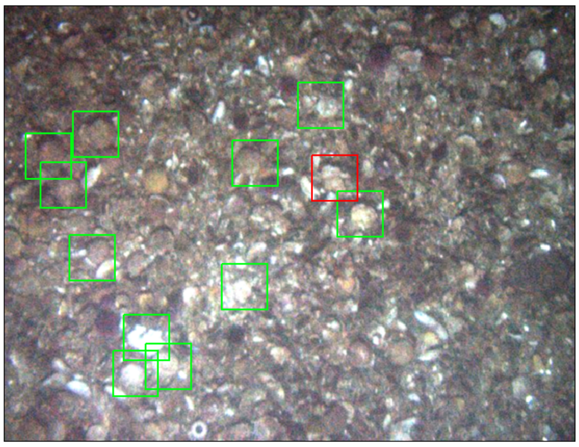
Altitude 1.78 m



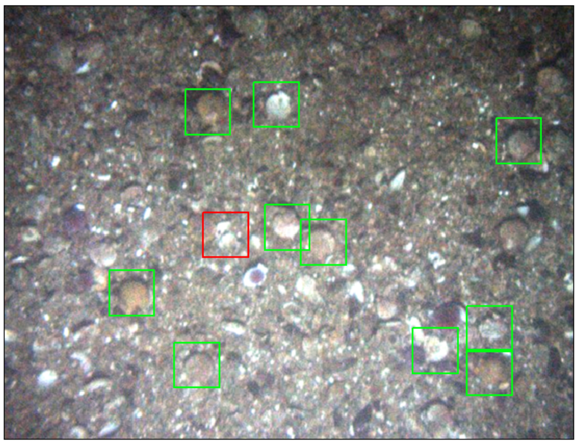
Altitude 2.04 m



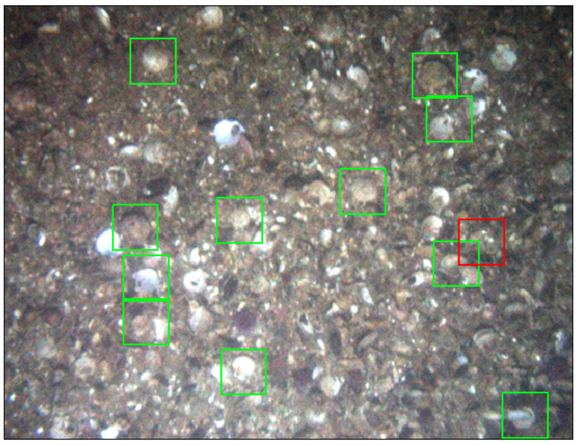
Altitude 2.09 m



Altitude 1.87 m

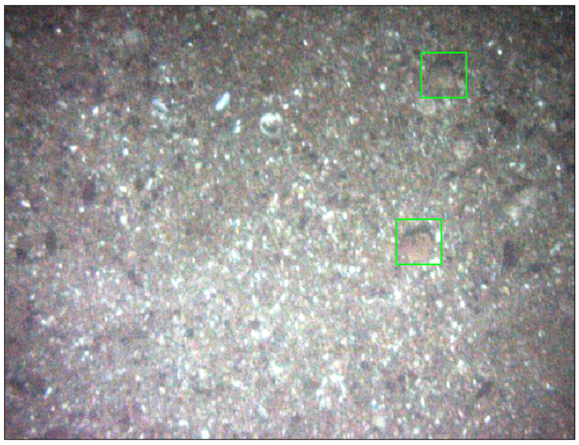


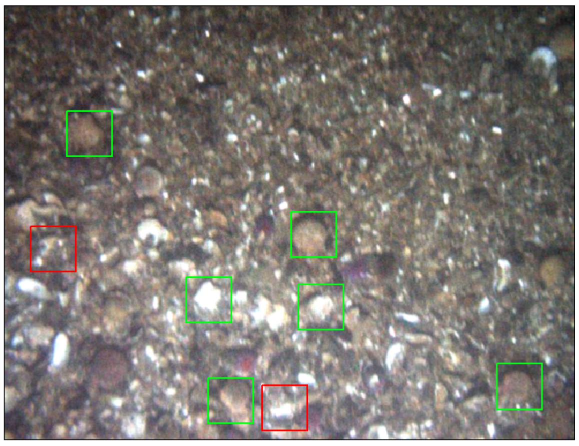
Altitude 1.90 m

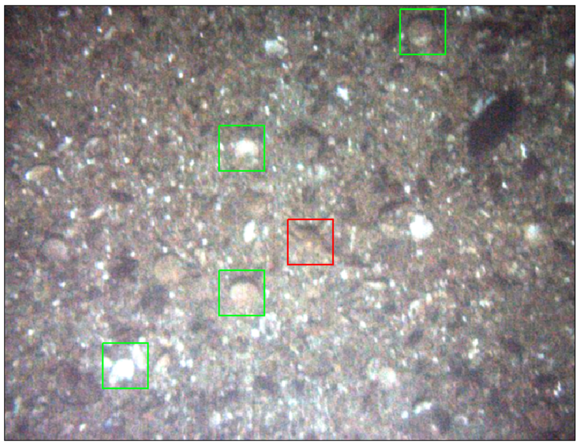


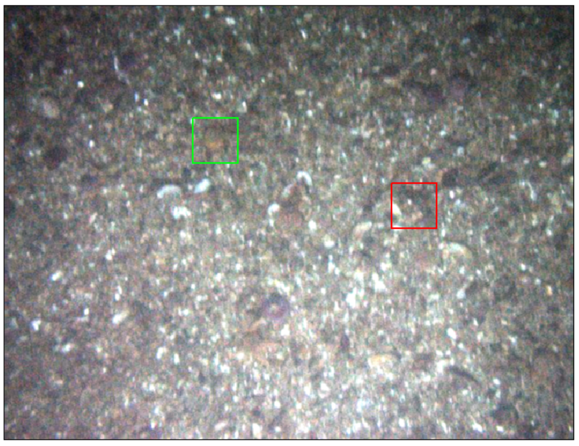
C. Results of The Test Images

True detections have green boxes and false detections have red boxes. Undetected scallops are left to the reader to determine.

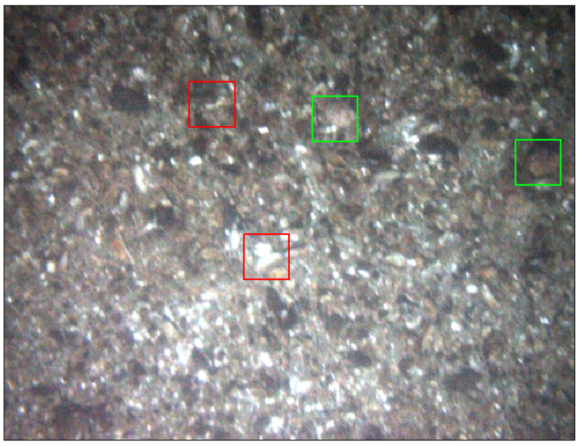

Altitude 2.00 m


Altitude 1.54 m

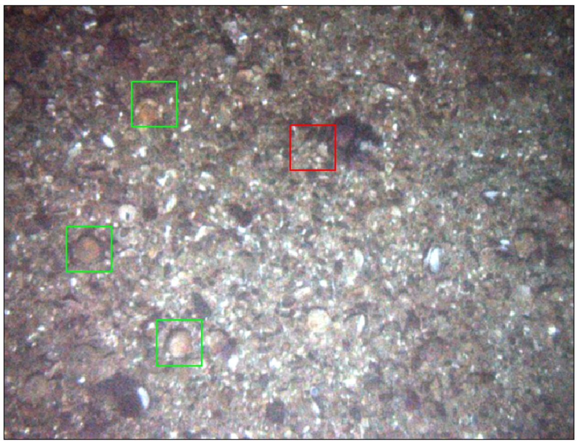

Altitude 1.93 m


Altitude 1.87 m

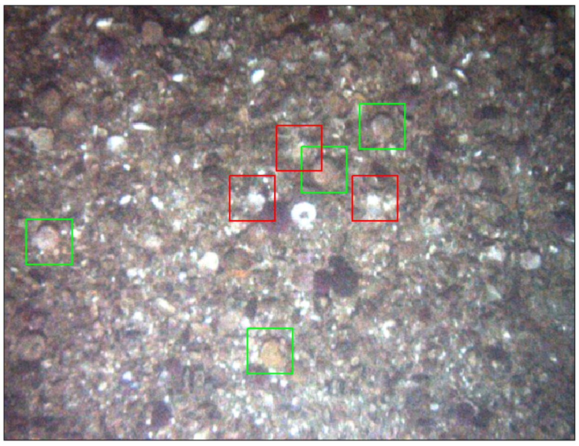
Altitude 1.70 m



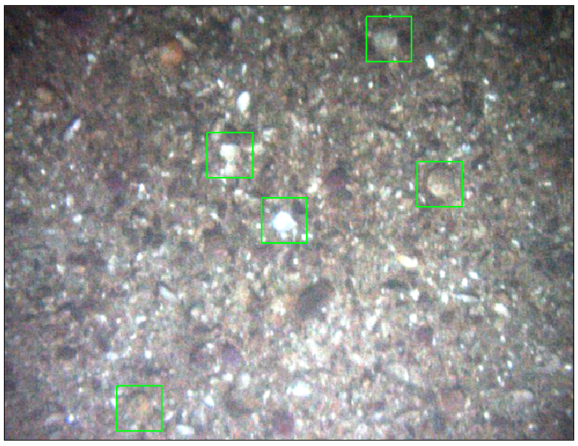
Altitude 2,07 m



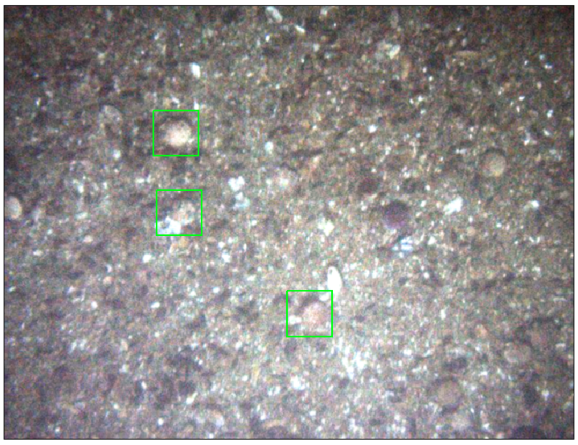
Altitude 2.06 m



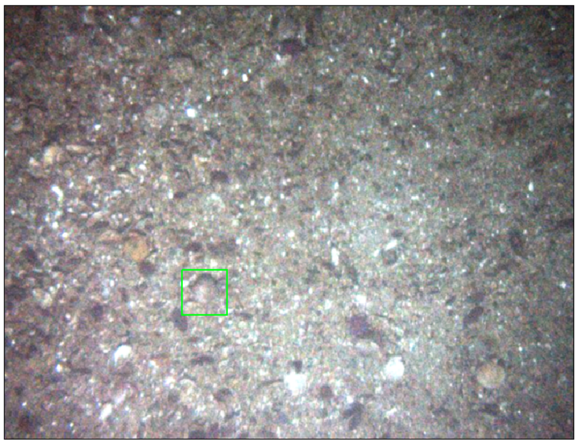
Altitude 1.88 m



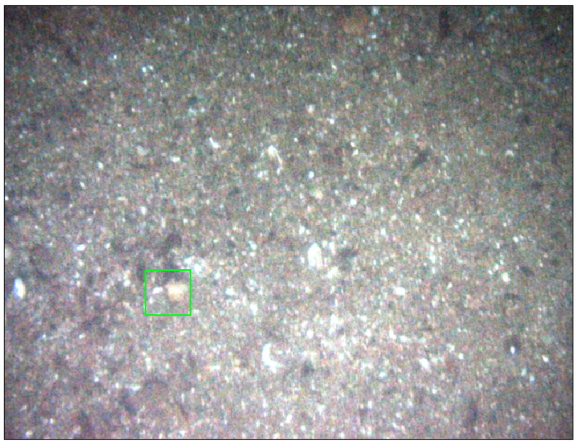
Altitude 1.93 m



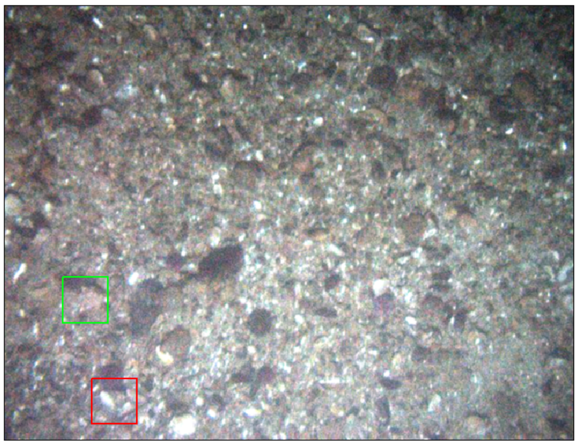
Altitude 1.93 m



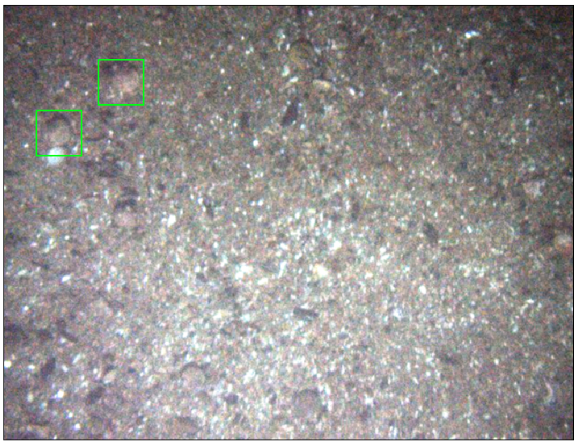
Altitude 2.00 m



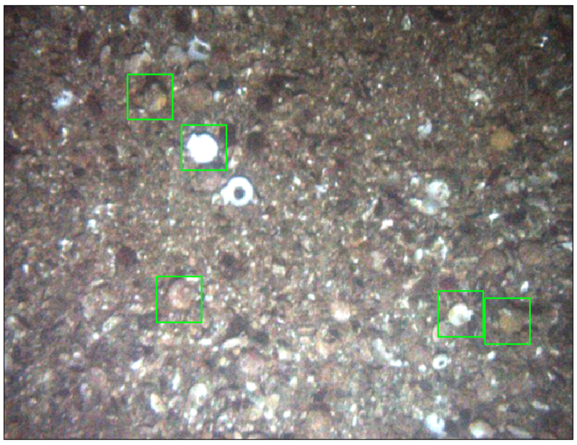
Altitude 1.98 m



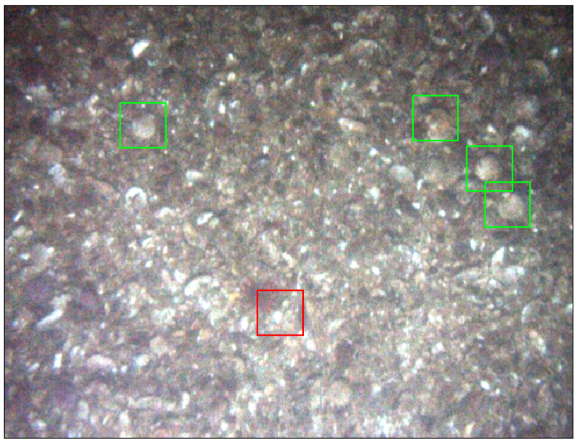
Altitude 1.92 m



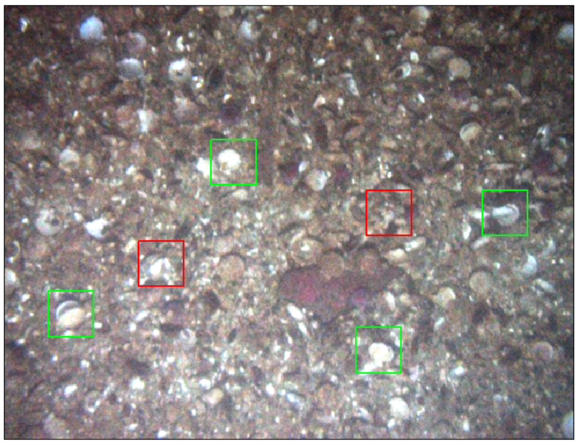
Altitude 1.86 m



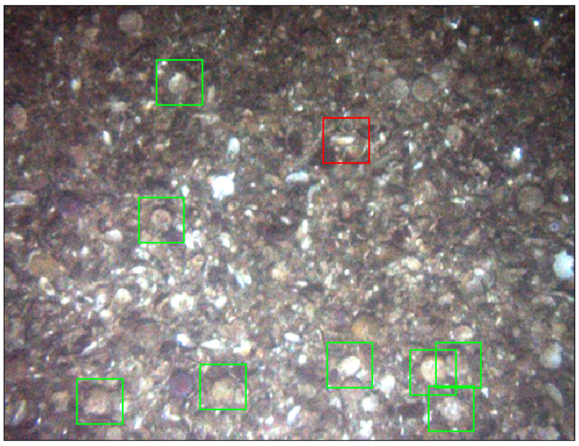
Altitude 1.87 m



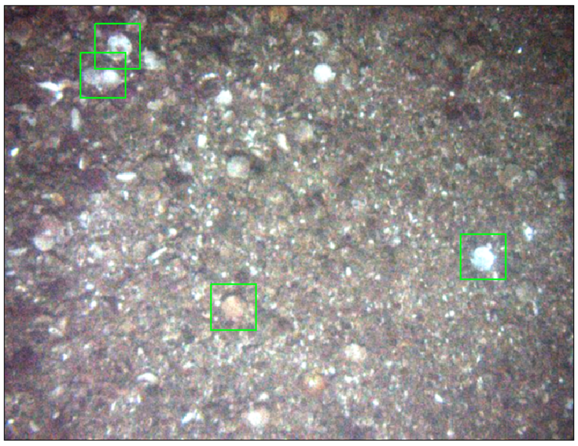
Altitude 1.93 m



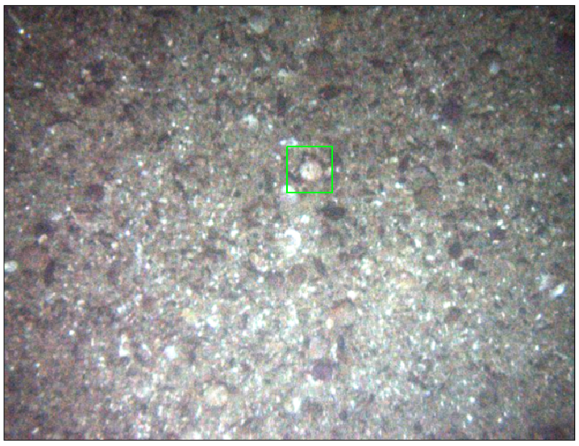
Altitude 1.94 m



Altitude 2.01 m



Altitude 2.0 m



Altitude 1.81 m