FIRST LINE OF TITLE

SECOND LINE OF TITLE

by

Author

A dissertation submitted to the Faculty of the University of Delaware in partial fulfillment of the requirements for the degree of Doctor of Philosophy in Major

Semester Year

FIRST LINE OF TITLE

SECOND LINE OF TITLE

by

Author

Approved: _____

Xxxx Xxxx, Highest Degree
Chair of the Department of Xxxx

Approved: _____

Xxxx Xxxx, Highest Degree
Dean of the College of Xxxx

Approved: _____

James G. Richards, Ph.D.
Vice Provost for Graduate and Professional Education

I certify that I have read this dissertation and that in my opinion it meets the academic and professional standard required by the University as a dissertation for the degree of Doctor of Philosophy.

Signed: _____

Xxxx Xxxx, Highest Degree
Professor in charge of dissertation

I certify that I have read this dissertation and that in my opinion it meets the academic and professional standard required by the University as a dissertation for the degree of Doctor of Philosophy.

Signed: _____

Xxxx Xxxx, Highest Degree
Member of dissertation committee

I certify that I have read this dissertation and that in my opinion it meets the academic and professional standard required by the University as a dissertation for the degree of Doctor of Philosophy.

Signed: _____

Xxxx Xxxx, Highest Degree
Member of dissertation committee

I certify that I have read this dissertation and that in my opinion it meets the academic and professional standard required by the University as a dissertation for the degree of Doctor of Philosophy.

Signed: _____

Xxxx Xxxx, Highest Degree
Member of dissertation committee

ACKNOWLEDGEMENTS

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABSTRACT

Chapter 1

EIGEN-VALUE BASED SHAPE DESCRIPTORS

1.1 Introduction

Identification of objects is a challenging and tricky problem. Not all object recognition solutions are robust to noisy input data and variations due to environmental factors. One way to recognize objects is through shape identification [27]. Eigen-value based shape descriptors [25, 41], is a mathematical framework that can identify prespecified geometric shapes in images. This method can be used to detect artificially introduced man-made objects describable by a strict geometric shape amongst other naturally occurring objects in images. An application for such a shape identification method was identified in the Redbird reef site. In the Redbird reef site off the coast New York-New Jersey, subway cars were dropped into the sea to aid artificial reef development [31, 30]. When research studies were conducted to study the impact of these subway cars on the geologic features [32, 38], the need for an automated method to pinpoint the locations of artificial objects from sonar survey data became apparent. Eigen-value based shape descriptors were employed to solved this problem.

1.2 Background

Eigen-value based shape descriptors is an object recognition solution designed to identify objects characterized by a specific shape. The object to be detected should have a unique prespecified shape that contrasts with the other objects found in the environment. This requirement is readily met when trying to identify man-made objects form natural scenes. Man-made objects tend to have strict geometric shapes compared to naturally occurring objects which exhibit a wide variation in shape and

appearance. This suggests that eigen-value based shape descriptors may be a useful tool for recognizing specific objects from natural images.

In shape-based identification problems, even if prior knowledge is available about the shape of the objects, the position and orientation of objects could be unknown. The task of searching for the same shape over different orientations in an image makes it computationally intensive. It is more effective to have a solution that can identify shapes irrespective of variations in rotation, scaling and translation of objects or in other words use a shape descriptor that is Rotation, Scaling and Translation (RST) *invariant*. The RST-invariant nature of eigen-value shape descriptors make them a useful tool for object recognition.

"Can one *hear* the shape of the drum?" This famous question by Kac [21] laid the foundation for the research into eigen-value based shape descriptors. This question can be rephrased as "Can one determine the shape of the drum membrane from the principal modes of vibration of the sound it produces?". If the principle modes of vibration of each drum membrane shape were unique, then would be an appealing possibility of using the eigen modes as descriptors for shape of drum membranes. Through a later work, Gordan et.al.[18] proved that a pair of iso-spectral drums produce sound with same principal modes. The work by Gorden answers Kac's question by stating that the eigen modes are not sufficient to uniquely identify the shape of a drum. However two independent papers by Khabou and Zuliani [25, 41] show that eigen modes can still be used as shape descriptors for practical purposes.

Eigen-value based shape descriptor offers a RST invariant object recognition solution for identifying predefined shapes. This is useful in applications where the objects to be recognized can be distinguished from other background objects primarily using their shape. This property is especially useful for identifying artificial objects with a well defined shape from natural scenes.

1.3 Preliminaries

The mathematical formulation of identifying the bounded planar domain Ω from its eigen-values λ_i can be applied to object recognition problem as in [25, 41]. The λ_i's are the eigen-values of the helmholtz differential equation (1.1) with Dirichlet boundary condition $u = 0$ on the boundary of the domain $\delta\Omega$.

$$\Delta u + \lambda u = 0 \tag{1.1}$$

When cast as an object to be recognized, the domain Ω is a binary profile representation of a shape in an image. Let the sequence of computed eigenvalues λ_i be

$$0 < \lambda_1 \leq \lambda_2 \leq \lambda_3 \leq \cdots \leq \lambda_k \leq \cdots \to \infty. \tag{1.2}$$

A shape Ω can then represented as a finite n-element vector of eigen ratios, which we call as shape descriptor F. For this application, the length of shape descriptor was truncated to 17 ($n = 17$).

$$F(\Omega) = \left\{ \frac{\lambda_1}{\lambda_2}, \frac{\lambda_1}{\lambda_3}, \frac{\lambda_1}{\lambda_4}, \ldots, \frac{\lambda_1}{\lambda_n} \right\}. \tag{1.3}$$

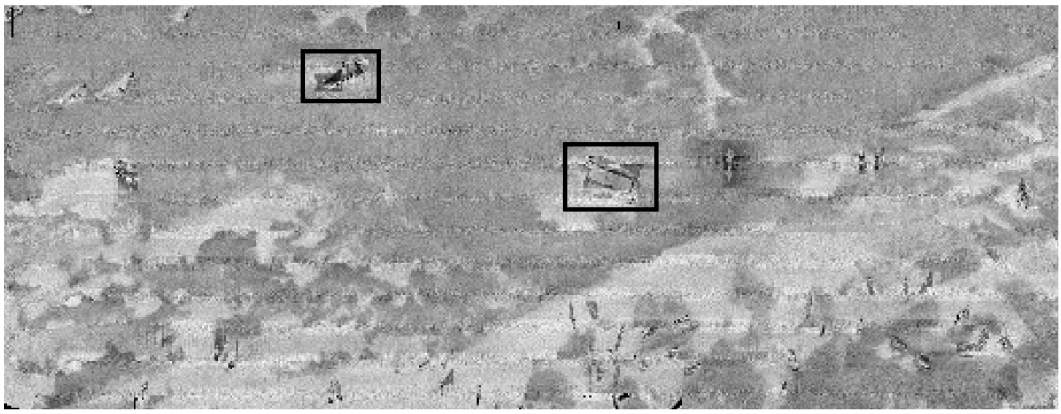
The angle Θ between two shape descriptor vectors $F(\Omega_1)$ and $F(\Omega_2)$ can be used as relative a measure of dissimilarity between the two shapes Ω_1 and Ω_2,

$$\Theta\left(\Omega_1, \Omega_2\right) = \cos^{-1}\left(\frac{\langle F(\Omega_1), F(\Omega_2) \rangle}{\|F(\Omega_1)\| \|F(\Omega_2)\|} \right). \tag{1.4}$$

Small values for the angle Θ would imply that the two shapes Ω_1 and Ω_2 are similar. For a pair of objects whose Θ value is less than a threshold, we can assume that the two objects are identical in terms of their shape.

1.4 Subway-car Detection from Seabed Images

The ability of eigen-value shape descriptors to match to certain shapes can be for object recognition in natural images, especially for identification of artificial objects from natural scenes. Artificial, or *man-made*, objects are often characterizable by a prespecified shape. On the other hand, naturally occurring objects can exhibit wide



(a) Backscatter image of sea-floor from Gavia AUV

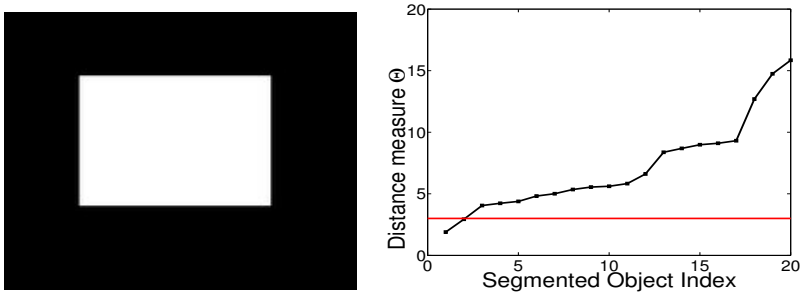
(b) Image segmentation using basic morphological operations and edge detection

Figure 1.1

intra-species and inter-species variation. For instance, it is easier to define the shape of an artificial object like a book compared to a natural object like a leaf. This enables eigen-value descriptors to pick out artificial objects from natural scenes based on their shape alone.

This problem of recognizing artificial objects from natural scenes was encountered while marine geologists were studying the Redbird artificial reef site [32, 38]. The focus of these studies were to observe geologic features around artificial objects like subway cars on the seabed. An automated method to detect subway cars would significantly facilitate such studies.

The sonar backscatter image of Redbird reef in Figure 1.1a shows the shape profiles of some sunken objects resting on the seabed. Two subway cars are marked using



(a) (b)

Figure 1.2: (a) Rectangular template used as reference shape (b) Plot of the weighted distance D between each segmented feature and the template rectangle. The threshold (D=3) is shown as a red line

black rectangles. If we look for rectangular objects like the rectangle in Figure 1.2a, it is likely that eigen-value descriptors will distinguis the profiles of subway-cars, as there are no other objects with such rectangular shape profile in the seabed image (Figure 1.1a).

In order to verify the ability of eigen-value descriptors to pick out the subway-cars in the Figure 1.1a, we follow a sequence of steps. The sonar backscatter image is first segmented to get a series of shape profiles using a grayscale threshold operation on the image. Figure 1.1b shows the different "blobs" obtained after thresholding. The blobs were then compared to the rectangular shape in Figure 1.2a using the angle Θ metric that was defined in (1.4). The Θ value of a blob is a direct indicator on how close in appearance it is to the reference rectangle in Figure 1.2a. The smaller the Θ value, the closer it is in appearance to the reference rectangle shape Ω_r. The $\Theta(\Omega_r, \Omega_j)$ value for each blob j is recorded and the Θ values are plotted in ascending order in Figure 1.2b.

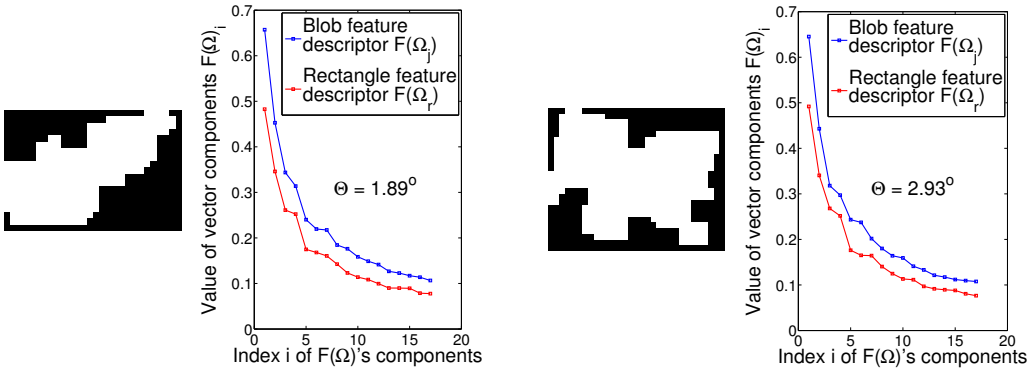
The two blobs that correspond to subway cars have the lowest Θ values. The blob with the lowest Θ value ($\Theta = 1.89°$) along with its plotted shape descriptor F is shown in Figure 1.3a. Similarly the blob with the second lowest Θ value is shown in Figure 1.3b($\Theta = 2.93°$). In Figures 1.3a and 1.3b, the blue line corresponds to the shape descriptor $F(\Omega_r)$ of reference rectangle Ω_r and the red line corresponds to the

shape descriptor $F(\Omega_j)$ of blob j. The $\Theta(\Omega_r, \Omega_j)$ is the computed angle between the shape descriptor vectors as discussed in (1.4). In contrast, the blob with the largest Θ value ($\Theta = 15.85°$) or in other words the blob that matches the least with the reference rectangle is shown in Figure 1.3c. In this case, if we set a threshold $\Theta_{thresh} = 3$ and only consider objects with $Theta < \Theta_{thresh}$ to be subway cars, then we have a mechanism to detect subway cars from other objects present in this sonar image. This effectively shows that eigen-value descriptors are in principle capable of picking up prespecified shapes from images.

1.5 Discussion

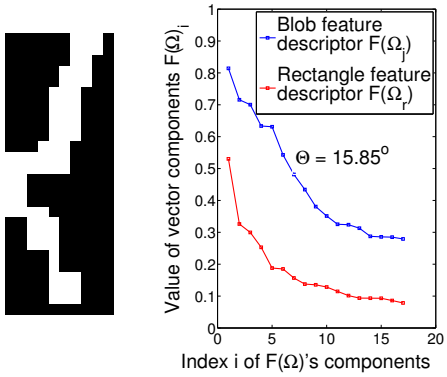
In Section 1.4, we saw how eigen-value based shape descriptors can be utilized to detect subway cars from other underwater objects. The eigen-value descriptors here were tuned to look for rectangular objects that match the profile in Figure 1.2a. The blobs obtained as matches to the reference rectangle through this eigen-value shape descriptor method complies with the ground truth information on the position of the subway cars in the sonar image (Figure 1.1a). Hence from this study, we see that eigen-value shape descriptor provide a viable mechanism to detect objects with specific shapes.

Since eigen-value shape descriptors purely rely on the shape of an object, two objects with identical shapes but significantly different textures cannot be differentiated using this method. In tests we performed, eigen-value descriptor did not perform well on objects whose shape is characterized by more complex contours. For instance when eigen-value shape descriptors were evaluated as a tool to recognize numbers between 0 and 9, 1 and 7 were often confused and wrongly classified. Similar misclassifications were also registered between 0,6,8 and 9. These errors can be attributed to the RST invariance property coupled with discretization errors while representing the numbers (0-9) in an image form. Even though RST invariance is helpful in some cases, it can be detrimental when the orientation of a shape can play a part in the recognition process.



(a)

(b)



(c)

Figure 1.3: Parts (a) and (b) correspond to the blobs that are closest to the reference rectangle in Figure 1.2a. The shape of the blob is shown on left and the plot of shape descriptor $F(\Omega_j)$ of blob j (red line) along with the plot of the reference rectangular profile (blue line) is shown on right. The distance measure Θ between the blob and the reference rectangle gives a picture of how close the given blob is to the reference rectangle. A contrasting view of the blob that matches the least with the reference rectangle is shown in (c).

The theoretical mechanism behind the eigen-value descriptors defines it over a continuous domain. When eigen-value descriptors were adopted as a shape identification tool on images, the domain needed to be discretized; images are nothing but discrete spatial arrangement of pixel values. When dealing with complex shapes, there can be significant discretization errors that can adversely affect the performance of eigen-value descriptors. High levels of noise and errors in segmentation are other significant factors that can lower the performance of eigen-value shape descriptors.

1.6 Conclusion

Eigen-value based shape identification is a tool that can be used to identify simple predefined shapes from natural images. This method was successful in identifying subway-cars from sonar images of the seabed under the assumption that subway cars are rectangular in shape. Apart from shape, the texture of an object plays a key role in determining the identity of an object. Since eigen-value shape descriptors discard textural information, they are not suited for applications where shape is insufficient to decipher the identity of an object. Additionally, while dealing with complex shapes, there can be significant discretization error and segmentation errors which affect the shape profile of an object. Such errors often diminish the performance of eigen-value shape descriptors. Hence eigen-value shape descriptors is a useful tool for identifying objects provided we can guarantee that the shape of the object can be described in a discrete domain with minimal error. Furthermore, for this method to work, the objects we are interested in should exhibit a shape profile that is significantly different from all other objects in the background. These special requirement limit eigen-value shape descriptors to few specialized object recognition applications only.

Chapter 2

MULTI-LAYERED SCALLOP RECOGNITION FRAMEWORK

2.1 Introduction

The sea scallop *(Placopecten magellanicus)* fishery in the US EEZ (Exclusive Economic Zone) of the northwest Atlantic Ocean has been, and still is, one of the most valuable fisheries in the United States. Historically, the inshore sea scallop fishing grounds in the New York Bight, i.e., Montauk Point, New York to Cape May, New Jersey, have provided a substantial amount of scallops [11, 35, 19, 28, 17]. These mid-Atlantic Bight "open access" grounds are especially important, not only for vessels fishing in the day boat category, which are usually smaller vessels with limited range opportunities, but also all the vessels that want to fish in near-shore "open access" areas to save fuel.[1] These areas offer high fish densities, but are at times rapidly depleted due to overfishing [33].

The 2011 Research Set-Aside (RSA) project (Titled: "A Demonstration Sea Scallop Survey of the Federal Inshore Areas of the New York Bight using a Camera Mounted Autonomous Underwater Vehicle") was a scallop survey effort undertaken to study the health of the scallop population along the coast of New York-New Jersey. As a part of this effort around a quarter million images of the ocean floor were recorded and a manual scallop enumeration was performed on these images. The considerable human effort involved for manual enumeration spawned the idea of building an automated species recognition system that can sift through millions of images and perform species enumeration with minimal to no human intervention. In response to this need for an automated scallop enumeration system, a multi-layered scallop recognition framework

[1] Based on personal communication with several limited access and day boat scallopers.

was proposed [22, 24, 23]. The workflow of this scallop recognition framework involves 4 processing layers: customized Top-Down Visual Attention (TDVA) pre-processing, robust image segmentation, and object classification and false positive filtering layers.

The value of the proposed approach in this disseration is primarily in providing a novel engineering solution to a real-world problem with economic and societal significance, which goes beyond the particular domain of scallop population assessment, and can possibly extend to other problems of environmental monitoring, or even defense (e.g. mine detection). Given the general unavailability of similar automation tools, the proposed one can have potential impact in the area of underwater automation. The multi-layered approach not only introduces several technical innovations at the implementation level, but also provides a specialized package for benthic habitat assessment. At a processing level, it provides the flexibility to re-task individual data processing layers for different detection applications. When viewed as a complete package, the approach offers an efficient tool to benthic habitat specialists for processing large image datasets.

In the this chapter, we discuss the details of the multi-layered scallop recognition system [22, 24, 23]. This chapter also lists information about the data collection effort that provided the scallop data for the scallop enumeration survey. Finally, an in depth comparison of the differences between this multi-layered framework and an earlier scallop recognition work [15] is discussed. For an overview of the existing literature on other scallop recognition work, refer to Chapter **??**.

2.2 Preliminaries

2.2.1 Visual Attention

Visual attention is a neuro-physiologically inspired machine learning method [26] that attempts to mimic the human brain function in its ability to rapidly single out objects that are different from their surroundings within imagery data. The method is based on the hypothesis that the human visual system first isolates points of interest in an image, and then sequentially processes these points based on the degree of interest

associated with each point. The degree of interest associated with a pixel is called *salience*, and points with the highest salience values are processed first. The method is used to pinpoint regions in an image where the value of some pixel attributes may be an indicator to its uniqueness relative to the rest of the image.

According to the visual attention hypothesis [26], in the human visual system the input video feed is split into several feature streams. Locations in these feature streams that are different from others in their neighborhood would generate peaks in the *center-surround* feature maps. The different center-surround feature maps can be combined to obtain a saliency *map*. Peaks in these resulting saliency maps, otherwise known as *fixations*, become points of interest, processed sequentially in descending order of their salience values.

Itti et al. [20] proposed a computational model for visual attention. According to this model, an image is first processed along three feature streams (color, intensity, and orientation). The color stream is further divided into two sub-streams (red-green and blue-yellow) and the orientation stream into four sub-streams ($\theta \in \{0°, 45°, 90°, 135°\}$). The image information in each sub-stream is further processes in 9 different scales. In each scale, the image is scaled down using a factor $\frac{1}{2^k}$ (where $k = 0, \ldots, 8$), resulting in some loss of information as scale increases. The resulting image data for each scale factor constitutes the *spatial scale* for the particular sub-stream.

The sub-stream feature maps are compared across different scales to expose differences in them. Through the spatial scales in each sub-stream feature map, the scaling factors change the information contained. Resizing these spatial scales to a common scale through interpolation, and then comparing them, brings out the mismatch between the scales. Let $\ominus$ be an pixel operator that takes pixel-wise differences between resized sub-streams. This function is called the *center-surround* operator, and codifies the mismatches in the differently scaled sub-streams in the form of another map: the center-surround feature map. In the case of the intensity stream, with $c \in \{2, 3, 4\}$ and $s = c + \delta$ for $\delta \in \{3, 4\}$ denoting the indices of two different spatial

scales, the center-surround feature map is given by

$$I(c,s) = |I(c) \ominus I(s)| \quad . \tag{2.1}$$

Similarly center-surround feature maps are computed for each sub-stream in color and orientation streams.

In this way, the seven sub-streams (two in color, one in intensity and four in orientation), yield a total of 42 center-surround feature maps. All center-surround feature maps in an original stream (color, intensity, and orientation) are then combined into a *conspicuity map* (CM): one for color $\bar{C}$, one for intensity $\bar{I}$, and one for orientation $\bar{O}$. Define the cross-scale operator $\oplus$ that adds up pixel values in different maps. Let w_{cs} be scalar weights associated with how much the combination of two different spatial scales c and s contributes to the resulting conspicuity map. If M is the global maximum over the map resulting from the $\oplus$ operation, and $\bar{m}$ is the mean over all local maxima present in the map, let $\mathcal{N}(\cdot)$ be a normalization operator that scales that map by a factor of $(M - \bar{m})^2$. For the case of intensity, this combined operation produces a conspicuity map based on the formula

$$\bar{I} = \bigoplus_{c=2}^{4} \bigoplus_{s=c+3}^{c+4} w_{cs}\, \mathcal{N}(I(c,s)) \quad . \tag{2.2}$$

The three conspicuity maps—for intensity, color and orientation—are combined to produce the *saliency map*. If scalar weights for each data stream are selected, say $w_{\bar{I}}$ for intensity, $w_{\bar{C}}$ for color, and $w_{\bar{O}}$ for orientation, the saliency map can be expressed mathematically as

$$S = w_{\bar{I}}\, \mathcal{N}(\bar{I}) + w_{\bar{C}}\, \mathcal{N}(\bar{C}) + w_{\bar{O}}\, \mathcal{N}(\bar{O}) \quad . \tag{2.3}$$

In a methodological variant of visual attention known as BUVA, all streams are weighted equally: w_{cs} is constant for all $c \in \{2,3,4\}$, $s = c + \delta$ ($\delta \in \{3,4\}$) and $w_{\bar{I}} = w_{\bar{C}} = w_{\bar{O}}$. A winner-takes-all neural network is typically used [20, 40] to compute the maxima, or fixations, on this map—other discrete optimization methods are of course possible. In the context of visual attention, fixations are the local maxima

of the saliency map. These fixations lead to shifts in *focus of attention*, or in other words, enables the human vision processing system to preferentially process regions around fixations in an image.

In a different variant of visual attention referred to as TDVA [29], the weights in (2.2) and (2.3) are selected judiciously to bias fixations toward particular attributes. There exists a method to select these weights in the general case when N_m maps are to be combined with those weights [29]. Let N be the number of images in the learning set, and N_{iT} and N_{iD} be the number of targets—in this case, scallops—and distractors (similar objects) in image i within the learning set. For image i, let P_{ijT_k} denote the local maximum of the numerical values of the map for feature j in the neighborhood of the target indexed k; similarly, let P_{ijD_r} be the local maximum of the numerical values of the map for feature j in the neighborhood of distractor indexed r. The weights for a combination of maps are determined by

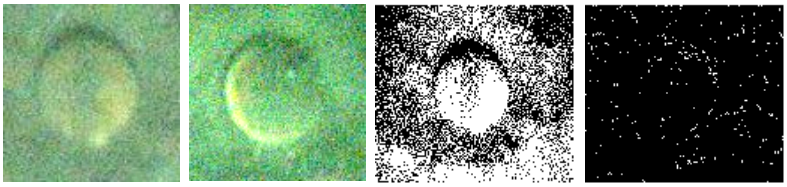
$$w'_j = \frac{\sum_{i=1}^{N} N_{iT}^{-1} \sum_{k=1}^{N_{iT}} P_{ijT_k}}{\sum_{i=1}^{N} N_{iD}^{-1} \sum_{r=1}^{N_{iD}} P_{ijD_r}}$$
$$w_j = \frac{w'_j}{\frac{1}{N_m} \sum_{j=1}^{N_m} w'_j}, \tag{2.4}$$

where $j \in \{1, \ldots, N_m\}$ is the index set of the different maps to be combined. Equations (2.4) are used for the selection of weights w_{cs} in (2.2), and $w_{\bar{I}}$, $w_{\bar{O}}$, $w_{\bar{C}}$ in (2.3).

2.3 Problem Statement

A visual scallop population assessment process involves identifying these animals in image datasets. A representative example of an image from the dataset we had to work with is shown in Figure 2.1 (scallops marked within red circles). A general solution to automated image annotation might not necessarily be effective for the dataset at hand. The need here is to identify algorithms and methods that will work best under *poor* lighting and imaging conditions, characteristic of this particular scallop counting application. The results from using elementary image processing methods like

Figure 2.1: [24] Seabed image with scallops shown in red circles



(a) (b) (c) (d)

Figure 2.2: [23] (a) Scallop with yellowish tinge and dark crescent; (b) Scallop with yellowish tinge and bright shell rim crescent; (c) Scallop with no prominent crescents and texturally identical to the background (d) Scallop sample after thresholding; (c) Scallop sample after edge detection.

thresholding and edge detection on the images (see Figure 2.2c and 2.2d) demonstrate the need for a more sophisticated approach (possibly a hybrid combination of several techniques).

Another challenge, related to the issue of low image resolution and high levels of speckle noise, is the selection of appropriate scallop features that would enable distinguishing between these organisms and other objects. In the particular dataset, one recurrent visual pattern is a dark crescent on the upper perimeter of the scallop shell, which is the shadow cast by the upper open scallop shell produced from the Autonomous Underwater Vehicle (AUV) strobe light (see Figure 2.2a). Another pattern that could serve as a feature in this dataset is a bright crescent on the periphery of the scallop, generally associated with the visible interior of the bottom half when the

scallop shell is partly open (see Figure 2.2b). A third pattern may be a yellowish tinge associated with the composition of the scallop image (see Figure 2.2b).

We have leveraged visual patterns [24] to develop a three-layered scallop counting framework that combines tools from computer vision and machine learning. This particular hybrid architecture uses top-down visual attention, graph-cut segmentation and template matching along with a range of other filtering and image processing techniques. Though this architecture offers a performance of over 63% true positive detection rate, it has a very large number of false positives. To mitigate this problem, we extend the framework [24] by adding a fourth, false-positives filtering layer [23].

2.4 Scallop Survey Procedure

The 2011 RSA project (Titled: "A Demonstration Sea Scallop Survey of the Federal Inshore Areas of the New York Bight using a Camera Mounted Autonomous Underwater Vehicle") was a proof-of-concept project that successfully used a digital, rapid-fire camera integrated to a Gavia AUV, to collect a continuous record of photographs for mosaicking, and subsequent scallop enumeration. In July 2011, transects were completed in the northwestern waters of the mid-Atlantic Bight at depths of 25-50 m. The AUV continuously photographed the seafloor along each transect at a constant distance of 2 m above the seafloor. Parallel sets of transects were spaced as close as 4 m. Georeferenced images were manually analyzed for the presence of sea scallops using position data logged (using Doppler Velocity Log (DVL) and Inertial Navigation System (INS)) with each image.

2.4.1 Field Survey Process

In the 2011 demonstration survey, the federal inshore scallop grounds from Shinnecock, New York to Ocean View, Delaware, was divided into eight blocks or strata (as shown in Figure 2.3). The *f/v Christian and Alexa* served as the surface support platform from which a Gavia AUV (see Figure 2.4) was deployed and recovered. The

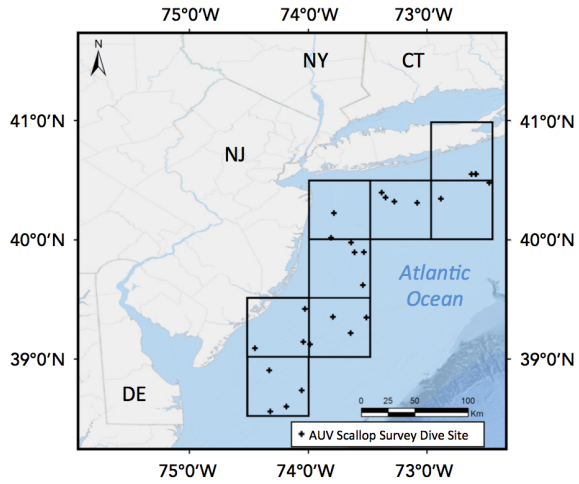


Figure 2.3: Map of the survey region from Shinnecock, New York to Cape May, New Jersey, divided into eight blocks or strata

AUV conducted photographic surveys of the seabed for a continuous duration of approximately 3 hours during each dive, repeated 3–4 times in each stratum, with each stratum involving roughly 10 hours of imaging and an area of about 45 000 m^2. The AUV collected altitude (height above the seabed) and attitude (heading, pitch, roll) data, allowing the georectification of each image into scaled images for size and counting measurements. During the 2011 pilot study survey season, over 250 000 images of the seabed were collected. These images were analyzed in the University of Delaware's Coastal Sediments, Hydrodynamics and Engineering Laboratory for estimates of scallop abundance and size distribution. The *f/v Christian and Alexa* provided surface support, and made tows along the AUV transect to ground-truth the presence of scallops and provide calibration for the size distribution. Abundance and sizing estimates were computed manually for each image using a GUI-based digital sizing software. Each image included embedded metadata that allowed it to be incorporated into existing benthic image classification systems (HabCam mip [15]).

During this proof of concept study, in each stratum the f/v *Christian and Alexa* made one 15-minute dredge tow along the AUV transect to ground-truth the presence of scallops and other fauna, and provide calibration for the size distribution. The

vessel was maintained on the dredge track by using Differential GPS. The tows were made with the starboard 15 ft (4.572 m) wide New Bedford style commercial dredge at the commercial dredge speed of 4.5–5.0 knots. The dredge was equipped with 4 inch (10.16 m) interlocking rings, an 11 inch (27.94 cm) twine mesh top, and turtle chains. After dredging, the catch was sorted, identified, and weighed. Length-frequency data were obtained for the caught scallops. This information was recorded onto data logs and then entered into a laptop computer database aboard ship for comparison to the camera image estimates.

The mobile platform of the AUV provided a more expansive and continuous coverage of the seabed compared to traditional fixed drop camera systems or towed camera systems. In a given day, the AUV surveys covered about 60 000 m^2 of seabed from an altitude of 2 m above the bed, simultaneously producing broad sonar swath coverage and measuring the salinity, temperature, dissolved oxygen, and chlorophyll-a in the water.

2.4.2 Sensors and Hardware

The University of Delaware AUV (Figure 2.4) was used to collect continuous images of the benthos, and simultaneously map the texture and topography of the seabed. Sensor systems associated with this vehicle include: (1): a 500 kHz GeoAcoustics GeoSwath Plus phase measuring bathymetric sonar; (2): a 900/1800 kHz Marine Sonic dual-frequency high-resolution side-scan sonar; (3): a Teledyne Rd Instruments 1200 kHz acoustic doppler velocity log (DVL)/Acoustic doppler current profiler (ADCP); (4): a Kearfott T-24 inertial navigation system; (5): an Ecopuck flntu combination fluorometer / turbidity sensor; (6): a Point Grey Scorpion model 20SO digital camera and LED strobe array; (7): an Aanderaa Optode dissolved oxygen sensor; (8): a temperature and density sensor; and, (9): an altimeter. Each sensor separately records time and spatially stamped data with frequency and spacing. The AUV is capable of very precise dynamic positioning, adjusting to the variable topography of the seabed while maintaining a constant commanded altitude offset.

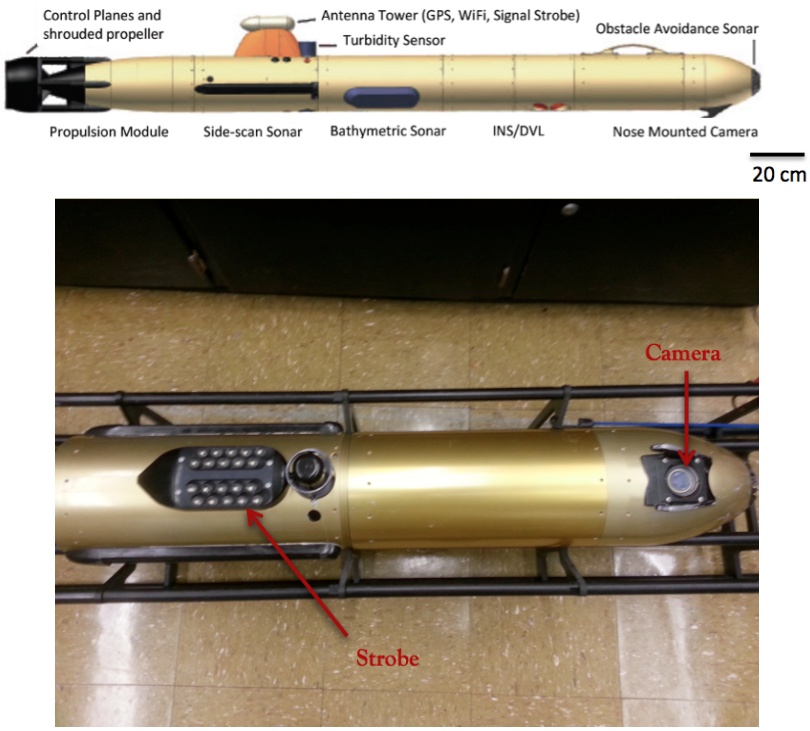


Figure 2.4: Schematics and image of the Gavia AUV

2.4.3 Data Collection

The data was collected over two separate five-day cruises in July 2011. In total, 27 missions were run using the AUV to photograph the seafloor (For list of missions see Table 2.1). Mission lengths were constrained by the 2.5 to 3.5 hour battery life of the AUV. During each mission, the AUV was instructed to follow a constant height of 2 m above the seafloor. In addition to the 250 000 images that were collected, the AUV also gathered data about water temperature, salinity, dissolved oxygen, geoswath bathymetry, and side-scan sonar of the seafloor.

The camera on the AUV, a Point Grey Scorpion model 20SO (for camera specifications see Table 2.2), was mounted inside the nose module of the vehicle. It was focused at 2 m, and captured images at a resolution of 800×600. The camera lens had a horizontal viewing angle of 44.65 degrees. Given the viewing angle and distance from the seafloor, the image footprint can be calculated as 1.86×1.40 m^2. Each image was saved in jpeg format, with metadata that included position information (including

latitude, longitude, depth, altitude, pitch, heading and roll) and the near-seafloor environmental conditions analyzed in this study. This information is stored in the header file, making the images readily comparable and able to be incorporated into existing RSA image databases, such as the HabCam database. A manual count of the number of scallops in each image was performed and used to obtain overall scallop abundance assessment. Scallops counted were articulated shells in life position (left valve up) [39].

2.5 Methodology

The multi-layered scallop counting framework that comprises four layers of processing on underwater images for the purpose of obtaining scallop counts is discussed in this section. The four layers involve the sequential application of Top-Down Visual Attention, Segmentation, Classification and False-Positive Filtering.

2.5.1 Layer I: Top-Down Visual Attention

2.5.1.1 Learning

A customized TDVA algorithm can be designed to sift automatically through the body of imagery data, and focus on regions of interest that are more likely to contain scallops. The process of designing the TDVA algorithm is described below.

The first step is a small-scale, bottom-up (BUVA) saliency computation. The saliency computation is performed on a collection of randomly selected 243 annotated images, collectively containing 300 scallops. This collection constitutes the *learning set*. Figure 2.5 represents graphically the flow of computation and shows the type of information in a typical image that visual attention tends to highlight.

A process of extremum seeking on the saliency map of each image identifies fixations in the associated image. If a 100×100 pixel window—corresponding to an approximately 23×23 cm^2 area on the seafloor—centered around a fixation point contained the center of a scallop, the corresponding fixation was labeled a *target*; otherwise, it is considered a *distractor*.

Mission	Number of images
LI1[1]	12 775
LI2	2 387
LI3	8 065
LI4	9 992
LI5	8 338
LI6	11 329
LI7	10 163
LI8	9 780
LI9	2 686
NYB1[2]	9 141
NYB2	9 523
NYB3	9 544
NYB4	9 074
NYB5	9 425
NYB6	9 281
NYB7	12 068
NYB8	9 527
NYB9	10 950
NYB10	9 170
NYB11	10 391
NYB12	7 345
NYB13	6 285
NYB14	9 437
NYB15	11 097
ET1[3]	9 255
ET2	12 035
ET3	10 474

[1] LI–Long Island
[2] NYB–New York Bight
[3] ET–Elephant Trunk

Table 2.1: List of missions and number of images collected

Attribute	Specs
Name	Point Grey Scorpion 20SO Low Light Research Camera
Image Sensor	8.923 mm Sony ccd
Horizontal Viewing Angle	44.65 degrees (underwater)
Mass	125 g
Frame rate	3.75 fps
Memory	Computer housed in AUV nose cone
Image Resolution	800 × 600
Georeferenced metadata	Latitude, longitude, altitude, depth
Image Format	jpeg

Table 2.2: Camera specifications

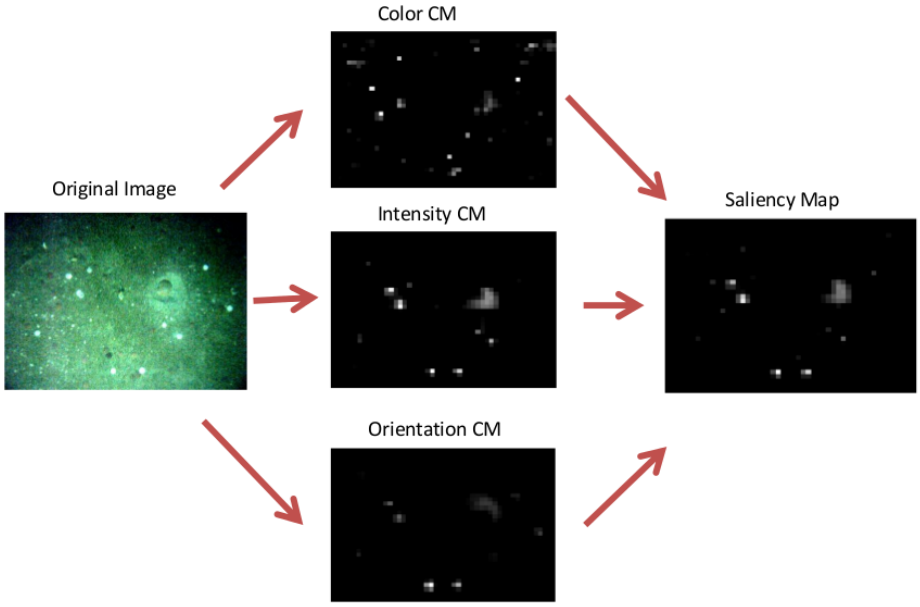


Figure 2.5: Illustration of computation flow for the construction of saliency maps

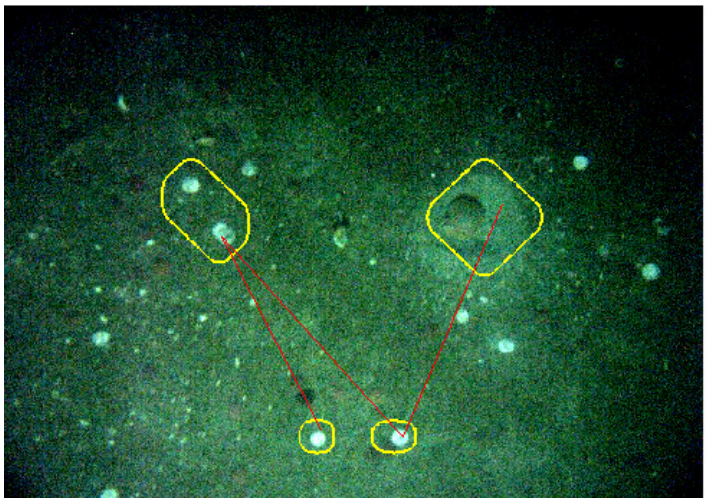


Figure 2.6: Illustration of fixations (marked by yellow boundaries): red lines indicate the order in which the fixations were detected with the lower-left fixation being the first.

The target and distractor regions are determined in all the feature and conspicuity maps for each one of these processed images in the learning set. This is done by adaptively thresholding and locally segmenting the points around the fixations with similar salience values in each map. Then the mean numerical value in neighborhoods around these target and distractor regions in the feature maps and conspicuity maps are computed. These values are used to populate the P_{ijT_k} and P_{ijD_r} variables in (2.4), and determine the top-down weights for feature maps and conspicuity maps.

For the conspicuity maps, the center-surround scale weights w_{cs} computed through (2.4) and consequently used in (2.2), are shown in Table 2.3. For the saliency map computation, the weights resulting from the application of (2.4) on the conspicuity maps are $w_{\bar{I}} = 1.1644$, $w_{\bar{C}} = 1.4354$ and $w_{\bar{O}} = 0.4001$. The symmetry of the scallop shell in our low-resolution dataset justifies the relatively small value of the orientation weight.

2.5.1.2 Implementation and Testing

To test the performance of the customized TDVA algorithm, it is applied on two image datasets, the size of which is shown in Table 2.5. In this application, the saliency maps are computed via the formulae (2.3) and (2.2), using the weights listed in Table

Table 2.3: Top-down weights for feature maps

		Center Surround Feature Scales					
		1	2	3	4	5	6
Color	red-green	0.8191	0.8031	0.9184	0.8213	0.8696	0.7076
	blue-yellow	1.1312	1.1369	1.3266	1.2030	1.2833	0.9799
Intensity	intensity	0.7485	0.8009	0.9063	1.0765	1.3111	1.1567
Orientation	0°	0.7408	0.2448	0.2410	0.2788	0.3767	2.6826
	45°	0.7379	0.4046	0.4767	0.3910	0.7125	2.2325
	90°	0.6184	0.5957	0.5406	1.2027	2.0312	2.1879
	135°	0.8041	0.6036	0.7420	1.5624	1.1956	2.3958

2.3. Convergence time of the winner-takes-all neural network that finds fixations in the saliency map of each image in the datasets of Table 2.5, is controlled using dynamic thresholding: It is highly unlikely that a fixation that contains an object of interest requires more than 10 000 iterations. If convergence to some fixation takes more than this number of iterations, then the search is terminated and no more fixations are sought in the image.

Given that an image in datasets of Table 2.5 contains two scallops on average, no more than ten fixations are sought in each image (The percentage of images in the datasets that contained more than 10 scallops was 0.002%). Since in the testing phase the whole scallop—not just the center—needs to be included in the fixation window, the size of this window is set at 270×270 pixels; more than 91% of the scallops are accommodated inside the window (Figure 2.7).

2.5.2 Layer II: Segmentation and shape extraction

This processing layer consists of three separate sub-layers: edge based segmentation (involves basic morphological operations like smoothing, adaptive thresholding and edge detection), graph-cut segmentation, and shape fitting. The flow of the segmentation process for a typical fixation window containing scallop is illustrated in

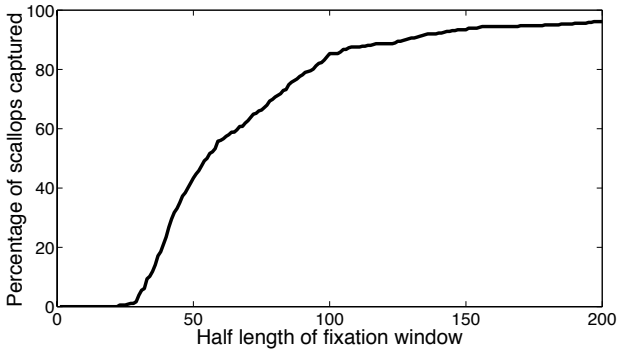
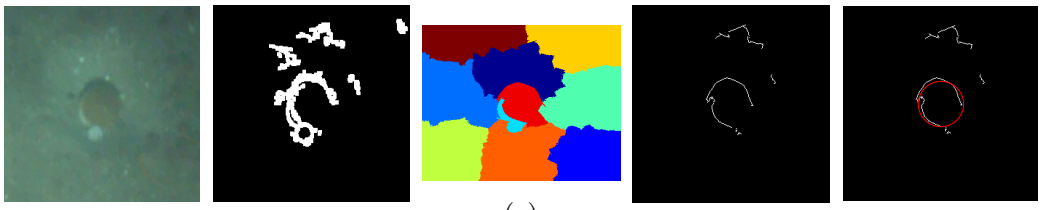


Figure 2.7: Percentage of scallops enclosed in the fixation window as a function of window half length (in pixels)



(a) (b) (c) (d) (e)

Figure 2.8: a Fixation window from layer I; b Edge segmented image; c graph-cut segmented image; d Region boundaries obtained when the edge segmented image is used as a mask over the graph-cut segmented image boundaries; e circle fitted on the extracted region boundaries.

Figure 2.8. Figure 2.8a shows a fixation window. Edge-based segmentation on this window yields the edge segmented image of Figure 2.8b. At the same time, graph-cut segmentation process [36] is applied on the fixation window to decompose it into 10 separate regions as seen in Figure 2.8c. The boundaries of these segments are matched with the edges in the edge segmented image. This leads to further filtering of the edges, and eventually to the region boundaries on Figure 2.8d. This is followed by fitting a circle to each of the contours in the filtered region boundaries (Figure 2.8d). Only circles with dimensions close to that of a scallop (diameter $20-70$ pixels) are retained (Figure 2.8e), which in turn helps in rejection of other non-scallop round objects.

The choice of the shape to be fitted is suggested by the geometry of the scallop's shell. Finding the circle that fits best to a given set of points is formulated as an optimization problem [37, 12].

Given a set of n points on a connected contour each with coordinates (x_i, y_i) $(i \in \{1, 2, \ldots, n\})$, define a function of four parameters A, B, C, and D:

$$F_2(A, B, C, D) = \frac{\sum_{i=1}^{n} [A(x_i^2 + y_i^2) + Bx_i + Cy_i + D]^2}{n^{-1} \sum_{i=1}^{n} [4A^2(x_i^2 + y_i^2) + 4ABx_i + 4ACy_i + B^2 + C^2]} \ . \tag{2.5}$$

It is shown [37] that minimizing (2.5) over these parameters yields the circle that fits best around the contour. The center (a, b) and the radius of this best-fit circle are given as a function of the parameters as follows:

$$a = -\frac{B}{2A} \ , \qquad b = -\frac{C}{2A} \ , \qquad R = \sqrt{\frac{B^2 + C^2 - 4AD}{4A^2}} \ . \tag{2.6}$$

For all annotated scallops in the testing image dataset, the quality of the fit is quantified by means of two scalar measures: the center error e_c, and the percent radius error e_r. An annotated scallop would be associated with a triple (a_g, b_g, R_g)—the coordinates of its center (a_g, b_g) and its radius R_g. Using the parameters of the fit in (2.6), the error measures are evaluated as follows, and are required to be below the thresholds specified on the right hand side in order for the scallop to be considered detected.

$$e_c = \sqrt{(a_g - a)^2 + (b_g - b)^2} \leq 12 \ \text{(pixels)} \qquad e_r = \frac{|R_g - R|}{R_g} \leq 0.3 \ .$$

These thresholds were set empirically, taking into account that radius measurements in manual counts used as ground truth [39] have a measurement error of 5–10%.

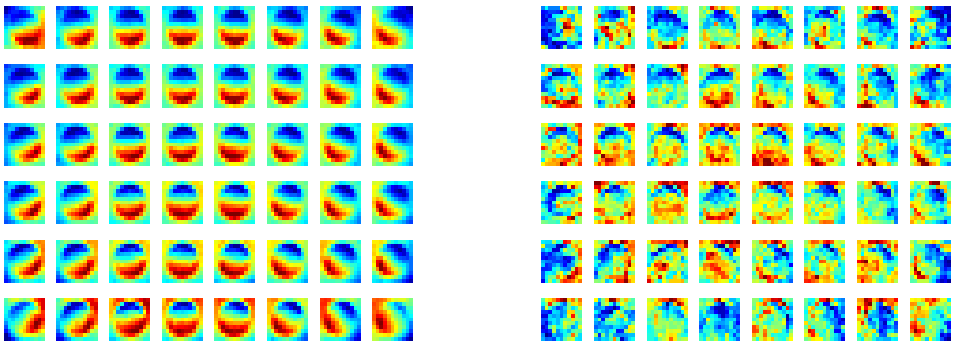
2.5.3 Layer III: Classification

The binary classification problem solved in this layer consists of identifying specific features in the images which mark the presence of scallops. These images are obtained by a using a camera at the nose of the AUV, illuminated by a strobe light close to its tail (mounted to the hull of the control module at an oblique angle to the camera). Our hypothesis is that due to this camera-light configuration, scallops appear in the images with a bright crescent at the lower part of its perimeter and a dark crescent at the top—a shadow. Though crescents appear in images of most scallops, their prominence and relative position with respect to the scallop varies considerably. The hypothesis regarding the origin of the light artifacts implies that the approximate profile and orientation of the crescents is a function of their location in the image.

2.5.3.1 Scallop Profile Hypothesis

A statistical analysis was performed on a dataset of 3 706 manually labeled scallops (each scallop is represented as (a, b, R) where a, b are the horizontal and vertical coordinates of the scallop center, and R is its radius). For this analysis, square windows of length $2.8 \times R$ centered on (a, b) were used to crop out regions from the images containing scallops.[2] Each cropped region was filtered in grayscale, contrast stretched, and then normalized by resizing to 11×11 dimension or 121 bins. To show the positional dependence of the scallop profiles, the image plane is discretized into 48 regions (6×8 grid). Scallops whose centers lie within each grid square are segregated. The mean (Figure 2.9a) and standard deviation (Figure 2.9b) of the 11×11 scallop profiles of all scallops per grid square over the whole dataset of 3 706 images was recorded. The

[2] Using a slightly larger window size ($> 2 \times R$, the size of the scallop) includes a neighborhood of pixels just outside the scallop which is where crescents are expected. This also improves the performance of local contrast enhancement, leading to better edge detection.



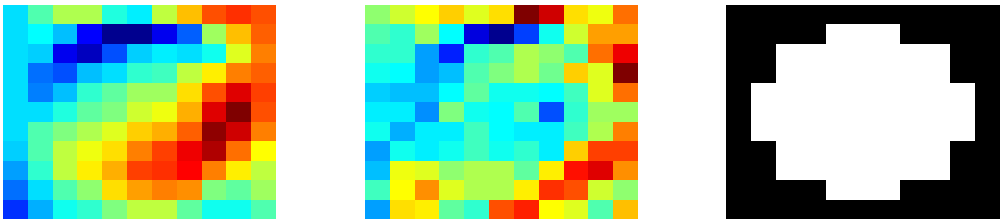
(a) (b)

Figure 2.9: a Mean map of scallops in each quadrant b Standard deviation map of scallops in each quadrant. Red corresponds to higher numeric values and blue correspond to lower numeric values.

lower standard deviation found in the intensity maps of the crescents on the side of the scallop facing away from the camera reveal that these artifacts are more consistent as markers compared to the ones closer to the lens.

2.5.3.2 Scallop Profile Learning

The statistics of the dataset of 3 706 images used to produce Figure 2.9 form a look-up table that represents reference scallop profile (mean and standard deviation maps) as a function of scallop center pixel location. To obtain the reference profile for a pixel location, the statistics from all the scallops whose centers lie inside a 40×40 window centered on the pixel is used. This look-up table can be compressed; it turns out that not all of the 121 bins (11×11) within each map is equally informative, because bins close to the boundary are more likely to include a significant number of background pixels. For this reason, a circular mask with a radius covering 4 bins is applied to each map (Figure 2.10), thus reducing the number of bins that are candidates as features for identification to 61. Out of these 61 bins, 15 additional bins having the highest standard deviation are ignored, leading to a final set of 46 bins. The value in the selected 46 bins from mean map forms a 46-dimensional feature vector associated



(a) (b) (c)

Figure 2.10: Intensity statistics and mask for a region centered at a pixel with coordinates $470, 63$) in the image a Map of mean intensity; b Map of intensity standard deviation; c Mask applied to remove background points.

with that region. The corresponding 46 bins from the standard deviation map are also recorded, and are used to weight the features (as seen later in (2.7)).

2.5.3.3 Scallop Template Matching

With this look-up table that codes the reference scallop profile for every scallop center pixel location, the resemblance of any segmented object to a scallop can now be assessed. The metric used for this comparison is a weighted distance function between the elements of the feature vector for the region corresponding to the segmented object, and that coming from the look-up table, depending on the location of the object in the image being processed. If this distance metric is below a certain threshold D_{thresh}, the object is classified a scallop. Technically, let $X^o = (X^o_1, X^o_2, \ldots, X^o_{46})$ denote the feature vector computed for the segmented object, and $X^s = (X^s_1, \ldots, X^s_{46})$ the reference feature vector. Every component of the X^s vector is a reference mean intensity value for a particular bin, and is associated with a standard deviation σ_k from the reference standard deviation map. To compute the distance metric, first normalize X^o to produce vector $X^{\bar{o}}$ with components

$$X^{\bar{o}}_p = \min_k X^s_k + \left(\frac{\max_k X^s_k - \min_k X^s_k}{\max_k X^o_k - \min_k X^o_k} \right) \left[X^o_p - \min_k X^o_k \right] \text{ for } p = 1, \ldots, 46 \ ,$$

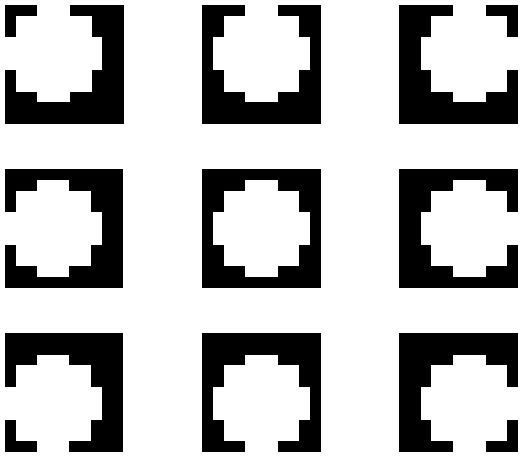


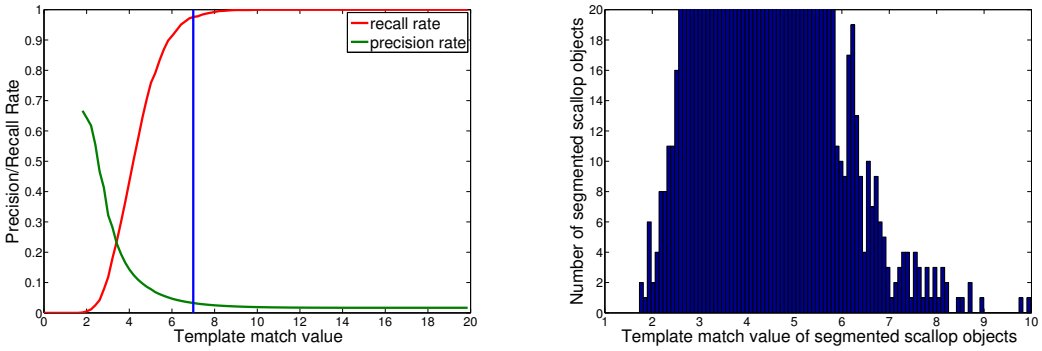
Figure 2.11: Nine different masks slightly offset from the center used to make the classification layer robust to errors in segmentation

and then evaluate the distance metric D_t quantifying the dissimilarity between the normalized object vector $X^{\bar{o}}$ and the reference feature vector X^s as

$$D_t = \sqrt{\sum_{k=1}^{n} \frac{\|X_k^{\bar{o}} - X_k^s\|^2}{\sigma_k}} \quad . \tag{2.7}$$

Small variations in segmentation can produce notable deviations in the computed distance metric (2.7). To alleviate this effect, the mask of Figure 2.10c was slightly shifted in different directions and the best match in terms of the distance was identified. This process enhanced the robustness of the classification layer with respect to small segmentation errors. Specifically, nine slightly shifted masks were used (shown in Figure 2.11). Out of the nine resulting distance metrics $D_t^{o_1} \ldots D_t^{o_9}$, the smallest $D_{\mathsf{obj}} = \min_{p\in\{1,\ldots,9\}} D_t^{o_p}$ is found and used for classification. If $D_{\mathsf{obj}} < D_{\mathsf{thresh}}$, the corresponding object is classified as a scallop. Based on Figures 2.12a–2.12b, the threshold value was chosen at $D_{\mathsf{thresh}} = 7$ to give a recall[3] rate of 97%. Evident in Figure 2.12a

[3] *Recall* refers to the fraction of relevant instances identified: fraction of scallops detected over all ground truth scallops; *precision* is the fraction of the instances returned that are really relevant compared to all instances returned: fraction of true scallops over all objects identified as scallops.



(a) (b)

Figure 2.12: a Precision-Recall curve with D_{thresh} shown as a vertical line; b Histogram of template match of segmented scallop objects.

is the natural trade-off between increasing recall rates and keeping the number of false positives low.

2.5.4 Layer IV: False Positives Filter

To decrease the false positives that are produced in the classification layer, two methods are evaluated as possible candidates: a high-dimensional wctm technique and a hog method. The main objective here is to find a method that will retain a high percentage of true positive scallop and at the same time eliminate as many false positives from the classification layer as possible.

2.5.4.1 High-dimensional weighted correlation template matching (WCTM)

In this method, the templates used are generated from scallop images that are *not* preprocessed, i.e., images that are not median-filtered, unlike the images that were processed by the first three layers. The intuition behind this is that although median filtering reduces speckle noise and may improve the performance of segmentation, it also weakens the edges and gradients in an image. Avoiding median filtering helps to generate templates that are more accurate than the ones already used in the classification layer.

Based on the observation that the scallop templates are dependent on their position in the image (Figure 2.9), a new scallop template is generated for each object that is classified as a scallop in Layer III. As indicated before, such an object would be represented by a triplet (a_o, b_o, R_o), where a_o and b_o represent the spatial Cartesian coordinates of object's geometric center, and R_o gives its radius. The representative scallop template is now generated from all scallops in the learning set (containing 3 706 scallops), of which the center is within a 40×40 window in the neighborhood of the object center (a_o, b_o). Each of these scallops is then extracted using a window of size $2.5R \times 2.5R$ where R is the scallop radius. Since these scallops in the learning set can be of different dimensions, it is resized (scaled) to a window of size $2.5R_o \times 2.5R_o$. All these scallop instances in the learning set are finally combined through a pixel-wise mean to obtain the mean representative template. Similarly, a standard deviation map that captures the standard deviation of each pixel in the mean template is also obtained. The templates produced here are of larger size compared to the templates in Layer III (recall that a Layer III template was of size 11×11). The inclusion of slightly more information contributes to these new larger templates being more accurate.

In a fashion similar to the analysis in Layer III, the templates and object pixels first undergo normalization and mean subtraction. Then they are compared. Let $v = (2.5R_o)^2$ be the total number of pixels in both the template and the object, and let the new reference scallop feature (template) and the object be represented by vectors $X^t = (X_1^t, X_2^t, \ldots, X_v^t)$ and $X^u = (X_1^u, \ldots, X_v^u)$, respectively. In addition, let σ be the standard deviation vector associated with X^t. Then the reference scallop feature vector X^t would first be normalized as follows:

$$X_p^{t'} = \min_k X_k^u + \left(\frac{\max_k X_k^u - \min_k X_k^u}{\max_k X_k^t - \min_k X_k^t} \right) \left[X_p^t - \min_k X_k^t \right] ,$$

where p denotes the position of component X_p^t in vector X^t. Normalization is followed by mean subtraction, this time both for the template and for the object. The resulting,

mean-subtracted reference scallop feature $X^{\bar{t}}$, and object $X^{\bar{u}}$ are computed as

$$X_p^{\bar{t}} = X_p^{t'} - \frac{1}{v}\sum_{k=1}^{v} X_k^{t'} \ , \qquad\qquad X_p^{\bar{u}} = X_p^{u} - \frac{1}{v}\sum_{k=1}^{v} X_k^{u} \ .$$

Now the standard deviation vector is normalized:

$$\bar{\sigma}_p = \frac{\sigma_p}{\sum_{k=1}^{v} \sigma_k} \ .$$

At this point, a metric that expresses the correlation between the mean-subtracted template and the object can be computed. This metric is weighted by the (normalized) variance of each feature. In general, the higher the value of this metric, the better the match between the object and the template. The wctm similarity metric is given by

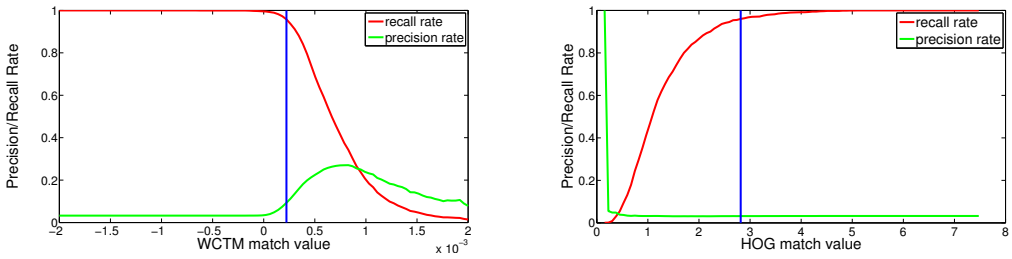
$$D_{\mathsf{wctm}} = \sum_{k=1}^{v} \frac{X_k^{\bar{t}} X_k^{\bar{u}}}{\bar{\sigma}_k} \ .$$

The threshold set for the weighted correlation metric D_{wctm}, in order to distinguish between likely true and false positives is at 0.0002222, i.e., any object with a similarity score lower than this threshold is rejected. This threshold value is justified from the precision-recall curves (see Figure 2.13a) of the weighted correlation metric values for the objects filtering down from the classification layer. The threshold shown by the blue line corresponds to 96% recall rate, i.e., 96% of the true positive scallops from the classification layer pass through wctm. At the same time, wctm decreases the false positives by over 63%.

2.5.4.2 Histogram of Gradients (HOG)

The hog feature descriptor encodes an object by capturing a series of local gradients in neighborhood of the object pixels. These gradients are then transformed into a histogram after discretization and normalization. There are several variants of hog feature descriptors. The R-hog used for human detection in [13] was tested here as a possible Layer IV candidate.

To produce R-hog, the image is first tiled into a series of 8×8 pixel groups referred to here as cells (the image dimensions need to be multiples of 8). The cells are



(a) (b)

Figure 2.13: Precision recall curve for Layer IV candidate methods (a) wctm and (b) hog. The blue line marks thresholds $D_{\mathsf{wctm}} = 0.0002222$ and $D_{\mathsf{hog}} = 2.816$. It is important to note that wctm is a similarity measure and hog is a dissimilarity measure. This implies that only instances below the indicated threshold D_{wctm} in wctm, and likewise instances above the threshold D_{hog} in hog, are rejected as false positives.

further divided into a series of overlapping blocks each containing groups of 2×2 cells. For each cell a set of 64 gradient vectors (one per pixel) is computed. Each gradient vector contains a direction and magnitude component. In the gradient directions, the sign is ignored reducing the range of angles from 0–360 down to 0–180. The gradient vectors are then binned into a 9–bin histogram ranging from 0–180 degrees with a bin width of 20 degrees. The contribution of each gradient vector is computed as half its gradient magnitude. The other half of the gradient magnitude is split between the two neighboring bins (in case of boundary bins, the neighbors are determined by wrapping around the histogram). The histograms from the 4 cells in each block is then concatenated to get vector v of 36 values (9 per cell). These vectors from each block are then normalized using their L_2-norm; for a vector v this normalization would be expressed as

$$\bar{v} = \frac{v}{\sqrt{\|v\|_2^2 + \epsilon^2}}$$

where ϵ is a small constant (here $\epsilon = 0.01$). The normalized vector $\bar{v}$ from each block is concatenated into a single feature vector F to get the hog descriptor for the input image.

Since this method imposes a constraint on the image dimensions being multiples of 8, the learning samples (each cropped using a square window of size of $3 \times$ radius)

are resized to 24×24. Here, we have to use both positive and negative object samples, the latter being objects other than scallops picked up in the segmentation layer. A hog feature vector F of length 144 (4 blocks $\times$ 4 cells $\times$ 9 values) is computed for each object instance obtained from the classification layer.

Now several different machine learning methods can be applied, using the positive and negative object instances as learning samples. As per the original implementation of the R-hog method [13], an Support Vector Machine (SVM) is used here. It turns out that the SVM learning algorithm fails to converge even after a large number of iterations. This could be attributed to the fact that the scallop profiles vary significantly based on their position in the image. To overcome this limitation, a lookup table similar to the one used to learn the scallop profiles in the classification layer is generated. The only difference here is that instead of saving a reference scallop template vector, a reference hog vector for only positive scallop instances from the learning set is recorded. The reference hog descriptor for a pixel coordinate in the image is taken to be the mean of all the hog descriptors of scallop instances inside a 40×40 window around the point.

For each instance classified as a scallop from the classification layer, its hog descriptor is compared with its corresponding learned reference hog descriptor from the lookup table. Since hog feature vectors are essentially histograms, the Earth Mover's Distance (emd) metric [34] is used to measure the dissimilarity between feature and object histograms. Let A and B be two histograms, and let m and n be the number of bins in A and B, respectively. Denote d_{ij} the spatial (integer) distance between bin i in A and bin j in B, and f_{ij} the smaller number of items that can be moved between bins i and j to ultimately make both histograms match (this is known as the *optimal flow* and can be found through a process of solving a linear program [34]). Then the emd metric D_{emd} that quantifies dissimilarity between two histograms A and B would be expressed as

$$D_{\mathsf{emd}}(A, B) = \frac{\sum_{i=1}^{m} \sum_{j=1}^{n} d_{ij} f_{ij}}{\sum_{i=1}^{m} \sum_{j=1}^{n} f_{ij}} \quad .$$

	hog		wctm	
	Dataset 1	Dataset 2	Dataset 1	Dataset 2
TP[1] from Classification Layer	183	1 759	183	1759
FP[2] from Classification Layer	7 970	52 456	7 970	52 456
TP after Layer IV	179	1 689	176	1 685
FP after Layer IV	7 752	51 329	2 924	16 407
Decrease in TP after Layer IV	4 (2.2%)	70 (4%)	7 (3.8%)	74(4.2%)
Decrease in FP after Layer IV	218 (2.7%)	1 127 (2.1%)	5 046 (63.3%)	36 049 (68.7%)

[1] TP–True Positives
[2] FP–False Positives

Table 2.4: Comparison of tested false positive filter layer methods

A precision-recall curve (shown in Figure 2.13b) with the classification threshold set as 2.816 (which corresponds to 96% recall rate, same rate used to set the wctm threshold). Any object with emd distance value less than this threshold is considered as a scallop. Though this threshold can capture 96% of the scallops, very few false positives actually get eliminated (less than 3%).

2.6 Results

The multi-layered detection approach is tested on two separate datasets containing 1 299 and 8 049 images, respectively. Among the two candidate methods tested for the fourth layer, wctm was chosen over hog due to it superior performance in terms of eliminating false positives. The difference in performance between hog and wctm is given in Table 2.4 for both datasets. Rows 1 and 2 in Table 2.4 show the true positives and false positives, respectively, that are filtered down from the initial 3 layers (Layers I-III). With these values as baseline, the thresholds for both hog and wctm were chosen to retain a high recall rate of close to 96%. This ensures that very few true positives are lost and their performance is primarily assessed through the reduction in false positives (row 6 of Table 2.4).

Since the thresholds are set such that the recall rate is high in both methods, the decrease in true positives is less than 5% in both hog and wctm. However there is a significant reduction in false positives (63.3% for dataset 1 and 68.7% for dataset 2) due

to wctm. On the other hand, the decrease in false positives is relatively small (less than 3%) for hog. It is not clear at this point why the hog filter fails to remove false positives. One reason could be that the hog filter derived from its native implementation for human detection in [13] might need further customization and even weighting through standard deviation weights like in wctm. Further study and detailed analysis is required to investigate and possibly improve its performance. In any case, the results support the inclusion of wctm as the false positive filter layer in the multi-layer scallop detection and counting process pipeline.

The overall performance of the four-layer pipeline is shown in Table 2.5. The results are compared to manually labeled ground truth. Only a subset of the available scallops—scallops at least 80 pixels horizontally and 60 pixels vertically away from the image boundaries—were used as ground truth. This was done to leave out scallops near the boundaries that were affected by severe vignetting effects. Such scallops were often too dark (see Figure 2.1) and very difficult to correct using standard vignetting correction algorithms. Furthermore, the scallop templates for scallops near the boundaries are such that their prime feature, the dark crescents, blend into the dark borders of the image (see Figure 2.9a). Inclusion of the boundaries would cause almost any objects near the boundary to be classified as scallops, resulting in a large number of false positives. It is also interesting to note that scallops only partially visible near the image boundaries were excluded in the manual counts performed [39].

Table 2.5 shows the results of the 3-layer pipeline along with the improvements in terms of the reduction in false positives as a result of introducing the fourth processing layer. The true positive percentages shown are computed with reference to the valid ground truth scallops (row 3 of table 2.5), i.e., scallops away from image boundaries. In dataset 1, which contains 1 299 images, the four-layer filtering results in a 70.4% overall recall rate, while in dataset 2 that contains 8 049 images the overall recall rate is 60.6%. Though the addition of the fourth false positive layer results in a small drop of 2.6% in recall rate, it eliminates over 63% of the false positives in both datasets. There is no clear reason for the better performance of this pipeline on dataset 2 both

Table 2.5: Results of multi-layer scallop classification

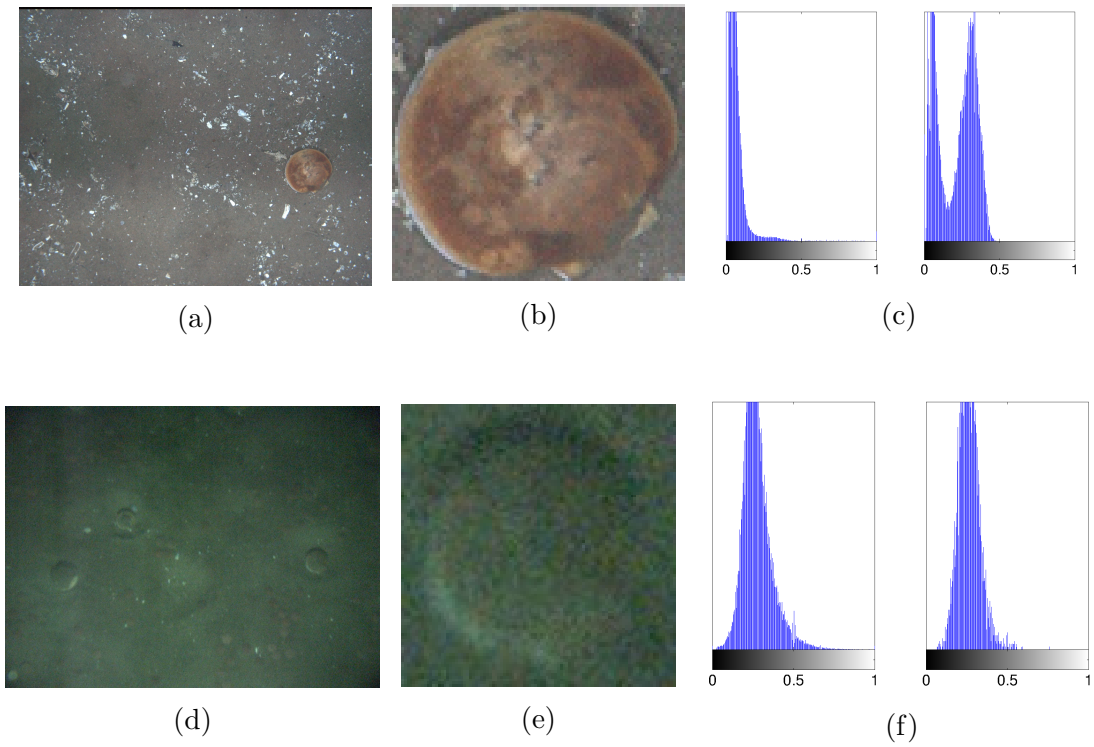
	Dataset 1	Dataset 2
Number of images	1,299	8,049
Ground Truth Scallops	363	3,698
Valid Ground Truth Scallops	250	2,781
True positives after Visual Attention Layer	231 (92.4%)	2,397 (86.2%)
True positives after Segmentation Layer	185 (74%)	1,807 (64%)
True positives after Classification Layer	183 (73%)	1,759 (63.2%)
True positives after False Positive Filter Layer	176 (70.4%)	1,685 (60.6%)
False positives after Classification Layer	7,970	52,456
False positives after False Positives Filter Layer (wctm)	2,924	16,407
Decrease in false positives (due to wctm)	63.3%	68.7%

in terms of recall rate and decrease in false positives compared to dataset 1.

2.7 Discussion

The four-layer automated scallop detection approach discussed here works on feature-poor, low-light imagery and yields overall detection rates in the range of 60–75%. Related work on scallop detection using underwater imaging [14, 15], reported higher detection rates, but the quality of the images used was visibly better. Specifically, the datasets on which the alternative algorithms [15] operated on, exhibit much more uniform lighting conditions, higher resolution, brightness, contrast, and color variance between scallops and background (see Figure 2.14). Evidence of this can be seen in Figure 2.14: the color variation between scallops and background data is reflected in the saturation histogram of Figure 2.14. While the histograms of scallop regions in the datasets of Table 2.5 is often identical to the global histogram of the image, the histograms of the Woods Hole data used by the alternative algorithms [15] present a bimodal saturation histogram (Figure 2.14c), from which foreground and background are easily separable.

Compared to another alternative approach that uses a series of bounding boxes



(a) (b) (c)

(d) (e) (f)

Figure 2.14: Representative samples of different imagery data on which scallop detection algorithms may be called to operate on. Figures 2.14a and 2.14d, show an image containing a single scallop from the dataset used by Dawkins et al.[15] (used with permission from the authors) and the datasets used in this paper respectively. A magnified view of a scallop cropped from Figure 2.14a and 2.14d can be seen in Figures 2.14b and 2.14e respectively. Figure 2.14c gives the saturation histogram of background or the complete image in Figure 2.14a to left and saturation histogram of Figure 2.14b to the right. Similarly, Figure 2.14f gives the saturation histogram of Figure 2.14d to the left and saturation histogram of Figure 2.14e to the right. The bimodal nature of the scallop histogram in Figure 2.14c derived from the dataset used in Dawkins et al.[15], clearly portrays the distinguishing appearance of the scallop pixels from the rest of the image, making it easily identifiable. The datasets we used did not exhibit any such characteristics (as seen in Figure 2.14f) to aid the identification of scallops.

to cover the entire image [16], the one reported here employs only ten windows per image, scanning the images at a much faster rate. Additionally, the detection rates there [16] were based on a dataset of just 20 images; statistically significant differences in performance rates between that approach and the one reported here would need much larger image samples.

2.8 Conclusions and Futurework

With the increasing use of underwater robotic platforms, terrabytes of imagery datasets featuring millions of images are becoming commonplace. The current practice of manual processing of these underwater images introduces a bottleneck. In the spirit of this scallop counting work, designing better and faster automated tools to characterize animals and other natural underwater phenomenon from images is imperative for future marine environmental studies.

This work is a step toward the development of an automated procedure for scallop detection, classification and counting, based on low-resolution imagery data obtained in the organisms' natural environment. The uniqueness of the reported method lies in its ability to handle poor lighting and low-contrast imaging conditions. A large natural datasets of over 8000 images have been used to validate this four-layered framework. Augmenting a previously developed three-layer scallop counting framework with a dedicated false-positive filtering layer has a drastic effect in terms of reducing the number of false positives. The study noted that a filter based on a custom wctm method outperforms hog in this specific application context. The multilayer framework reported is verified to be modular, and it allows easy adaptation of different layers for various applications like counting other sea organisms. Designing such tools with further improvements in form of higher detection rates and lower false positives is required to help advance future marine animal studies.

2.9 Future Work

One future direction would be to further reduce false positives by enhancing the false positives filter layer by using multiple scallop reference templates for each pixel location. These new templates could be designed to capture the bright crescents that sometimes appear due to the visible interior of the lower valve of a scallop when the scallop shell is partly open. As this crescent appearance is only dependent on the relative scallop orientation with respect to the camera, it can occur at any point in the periphery of a scallop. If these bright crescents were to be used in conjunction with dark crescents multiple templates will be required to model scallops at each pixel location. This idea is supported by inspection of recently collected high-resolution scallop data, which indicate additional definitive features connecting the position of the bright and dark crescents along with their relative intensities. We believe that even without major changes to the current framework, testing on higher resolution images could produce much better performance outcomes (both in terms of detection and false positive rates). The unavailability of ground truth for the new datasets makes it hard to provide evidence of any performance at this point. It is also expected that using more targeted color and light correction methods [15] as a part of image preprocessing will improve results.

Chapter 3

COOPROV: LOW COST UNDERWATER REMOTELY OPERATED VEHICLE

3.1 Introduction

Field experiments related to underwater object recognition require a submersible robot with appropriate sensing and control payload to navigate through points of interest and capture necessary data through sensors. Depending on the level of manual intervention involved to guide the vehicle, an underwater vehicle, can be classified as an AUV, a Remotely Operated Vehicle (ROV) or a Human Operated Vehicle (HOV). AUVs navigate autonomously with minimal to no manual intervention. Both ROVs and HOVs are piloted by humans, with the difference being that the human operator is within the vehicle in an HOV, whereas in the ROV the human operator controls the vehicle from a remote location. Commercially available underwater vehicles come with wide range of capabilities in terms of sensing and operational depths. The sensing package or the type of vehicle required is dictated by the objective of the science experiment and its requirements. For validation and fine-tuning of the underwater object recognition algorithms for multi-layered scallop recognition 2 and the multi-image object recognition **??** developed in this dissertation, an underwater vehicle that can navigate to specific points and collect image data of underwater targets is needed. With these requirements in mind, a low-cost underwater research vehicle named CoopROV was developed. CoopROV is a ROV-class vehicle that is equipped with several sensors for data collection, and also has the capability to incorporate any customized controller.

3.2 Commercial Submersible Robots

Commercially available robots are expensive and often provide minimal access to their legacy controllers and software. Most commercial ROVs come with a graphical interface and joystick. AUVs come with an interface to allow the user to predefine the robot trajectory through a series of geo-tagged waypoints before deployment. Once deployed, the AUV tracks the specified trajectory. Such manufacturer interfaces do not provide considerable freedom to modify the core software running on the robot. For instance if someone wants to use their own controller, it becomes problematic to implement it without manufacturer supported software Application Program Interface (API)s. Cost is another factor that limits accessibility to commercial underwater robots. A relatively small ROV like the videoray [10] with a footprint of $14.75 \times 11.4 \times 8.75$ ($L \times W \times H$ in inches) and a sensor package comprising a camera, gyros, accelerometers, compass, depth and temperature sensors can cost over 25 000 dollars. Bigger ROVs like the Outland ROV [7] can cost upwards of 75 000 dollars with the cost here being chiefly determined by the sophistication of the on board sensing capabilities. Despite the high cost and minimal flexibility for customer modifications, the primary advantage of these commercial solutions is their well tested hardware that is resilient to harsh conditions that are characteristic to the marine environments.

The other league of underwater robotic solutions come in form of low-cost Do-It-Yourself (DYI) style robots like the OpenROV [6] and the Fathom One [2]. Such solutions offer a lot of potential for custom hardware and software modifications. However being small-scale development projects, the associated hardware in these cases is not well tested and can be prone to failures (e.g. water leaks). The capability of these systems to withstand the harsh conditions of a marine environment may be limited. In terms of software, the source code is often available and comes with an open-source license (as in the case of OpenROV). However such software often does not come with APIs to encourage users to extend or modify the capabilities of the robot. Additionally, the limited documentation available makes the task of directly modifying the source code cumbersome. Since the focus of most of these DYI projects are to offer low-cost

options that enable primarily hobbyists to explore underwater, a teleoperation interface and access to a camera feed from the robot typically suffices. For researchers that may have very specialized interests, an ideal robotic solution should offer finer control of the hardware and software with options to easily extend the base capabilities of the system.

There are two primary requirements for a professionally developed robotic solution that is intended for research. The first is the need for a tough exterior shell that can hold up against high values of water pressure which is common with deep sea operation. This exterior shell should also be resilient to water leaks and at the same time offer the ability to retrofit sensors. The second requirement is for the robotic system to offer a well built software interface with APIs that allows researchers to extend the capabilities of the system to match their needs. It will be beneficial if the software APIs adhere to popular robotics research architectures like the ROS [9] or Mission Oriented Operating Suite (MOOS) [4]. A robot with ROS-support will greatly enhance the versatility of the robot by allowing access to a plethora of existing software tools. Furthermore, software development based on open-source tools like ROS can be supported an active base of researchers that can provide feedback, tips and advice. Currently the underwater research community lacks a variety of low-cost submersible robotic platforms with resilient hardware design and well-engineered software.

3.3 CoopROV

The chief requirements that led to the design of CoopROV was the need for a low-cost submersible robotic platform that can serve as a test-bed for computer vision and control algorithms. Before the effort to build CoopROV was initiated, existing solutions were evaluated. Commercial robotic solutions like the Videoray were clearly outside the price range of the budget of this research project. OpenROV 2.5 [6], the first version of OpenROV, was a good candidate primarily due to it sub-1000 dollar price tag. The original OpenROV design came with a beaglebone and browser based graphical user interface that used node.js [5] in the backend to drive the robot.

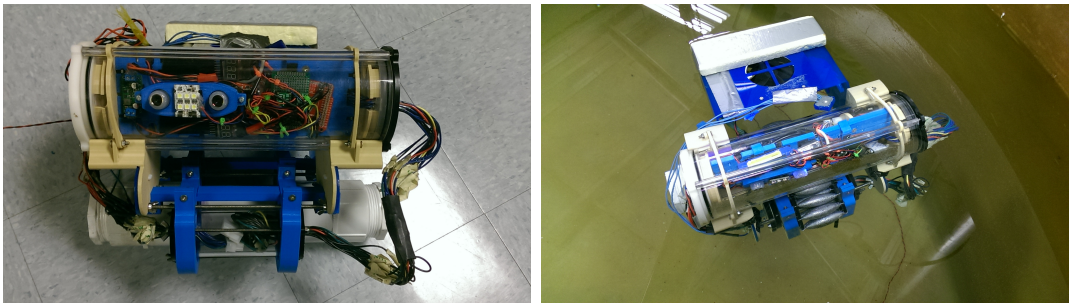


Figure 3.1: CoopROV shown in two different views

However, multiple tests revealed that the water proofing design and electronics package had several glitches making it unsuitable for deployment in its original configuration. Furthermore, most of the algorithm development till that point was in ROS, hence ROS support was one of the preferred requirements for the submersible robot. The initial evaluation experiments of OpenROV provided useful insights into the core components and associated problems that need to be tackled to design a submersible robot. The design of CoopROV was initiated to address the shortcomings of OpenROV's hardware in terms of waterproofing and electronics. Additional features of the target design was to allow easy interfacing of new sensors along with full ROS software support for sensors and actuators.

In the current stage of development, CoopROV shows some resemblance to OpenROV as both of them share the same frame and actuators. However, all electronics, power-supply and sensors in CoopROV have diverged completely from OpenROV 2.5. Two views of CoopROV a frontal view and its appearance when deployed in a test tank are shown in Figure 3.1.

3.3.1 System Overview

CoopROV is a ROV-class submersible that is connected to a surface station through a tether. The surface station is a laptop computer. CoopROV and its surface station are connected by an ethernet interface and run a distributed ROS system. In teleoperation mode, a joystick controller connected to the surface station can drive the

Weight	7 Kg
Dimensions (L x W x H)	0.30 x 0.28 x 0.39 m
Run time	1 hour
Sensors	Minoru stereo 3D webcam
	LIS3MDL 3-axis magnetometer
	LSM6DS33 3-axis accelerometer and gyro
	MS5803 pressure and temperature sensor
	LiPo tester voltage sensor

Table 3.1: CoopROV Specifications

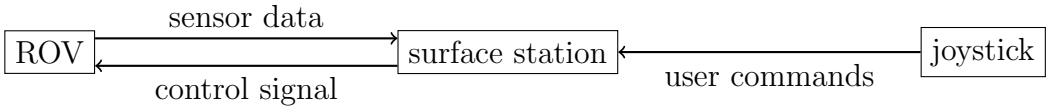


Figure 3.2: Block diagram of high-level data flow between different components in CoopROV

robot. The joystick controller can actuate the 3 thrusters (two at the back and one on top) offering the robot a 6-Degree of Freedom (DOF) motion capability. This data flow between the different system components is graphically illustrated in Figure 3.2. The computing on the robot is handled by a RaspberryPi [8] and Arduino Mega[1]. The sensor package includes a stereo camera, a 9-DOF Inertial Measurement Unit (IMU), depth, temperature and voltage sensors. The robot is powered by two sets of Lithium Polymer (LiPo) batteries which can provide up to an hour of run time. For a brief list of CoopROV specifications see Table 3.1. A more detailed account of different subsystems comprising hardware, electronics, software, sensors and power supply is given in the following sections.

3.3.2 Hardware

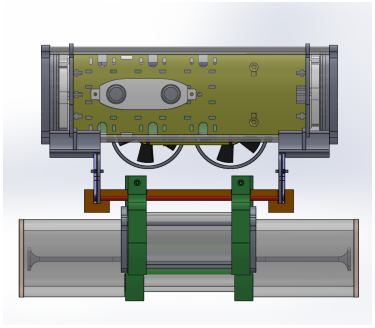
The exterior shell of CoopROV is composed of an acrylic frame that holds all housing tubes together. There are 3 housing tubes in total: an electronics tube, a connector tube and a battery tube. A CAD model of CoopROV in Figure 3.3 illustrates the position of the housing tubes on the CoopROV frame. The electronics tube is a long transparent acrylic tube that holds all sensors and most electronics of the robot.

The connector housing tube is a relatively short transparent acrylic tube with a molex connector plug inside it. This connector tube serves as a water proof connector plug that allows to change the sensor package by swapping out the electronics tube. The battery tube is a PVC pipe that holds a 11.1 V Li Ion batteries. The clips and clamps that hold the tubes in place were made out of 3D-printed plastic parts. One big design improvement over OpenROV is the water proof seals that comprise a delrin end cap with an O-ring slot along with 3 threaded rods that maintain the tension between the end cap and the acrylic tube.

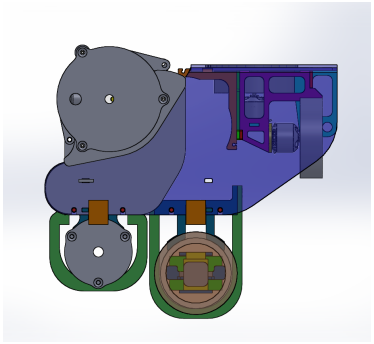
3.3.3 Electronics

The electronic components inside the electronics tube are responsible for controlling the thrusters and relaying the sensor data back to the surface station. The RaspberryPi and the surface station are components of a networked ROS system. The RaspberryPi connects to the surface station through Transmission Control Protocol (TCP) connection via the tether cable. The Arduino is connected to the RaspberryPi via the serial bridge and behaves like a node of the ROS system on the RaspberryPi. Data logging from the depth sensor and the IMU along with sending the motor control commands to the motor drivers is handled by the Arduino. The stereo camera is directly connected to RaspberryPi and the stereo logging node on the RaspberryPi handles the image data acquisition.

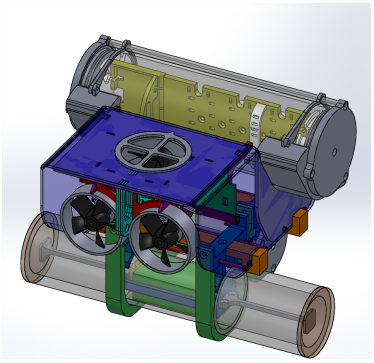
There are two parallel data streams. One transfers the control commands that flow through the system and other sends the sensor data being logged from the various sensors. The motion control commands that come from the surface station are fed by the RaspberryPi into the Arduino before being translated into motor commands and sent to the motor driver, which in turn sets the appropriate voltage to run the motors connected to the propellers. The sensor data is logged by both the Arduino and RaspberryPi and served to any other node on the ROS system on request. The block diagram in Figure 3.4 provides a graphical overview of the connections between the major electronic components on CoopROV. It its important to note that only one



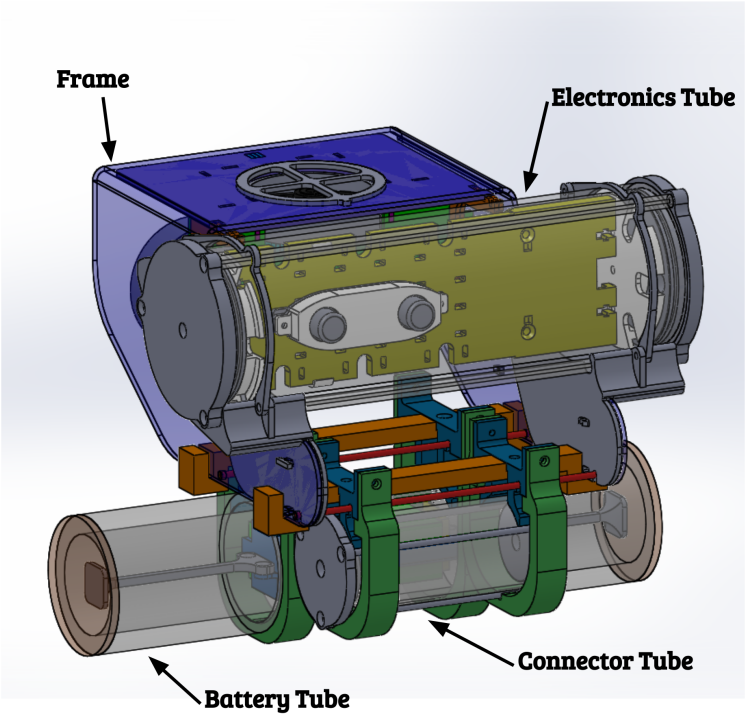
(a) Front view



(b) Side view



(c) Isometric view (back)



(d) Isometric view (front)

Figure 3.3: Different views of the CoopROV CAD model with some labeled parts

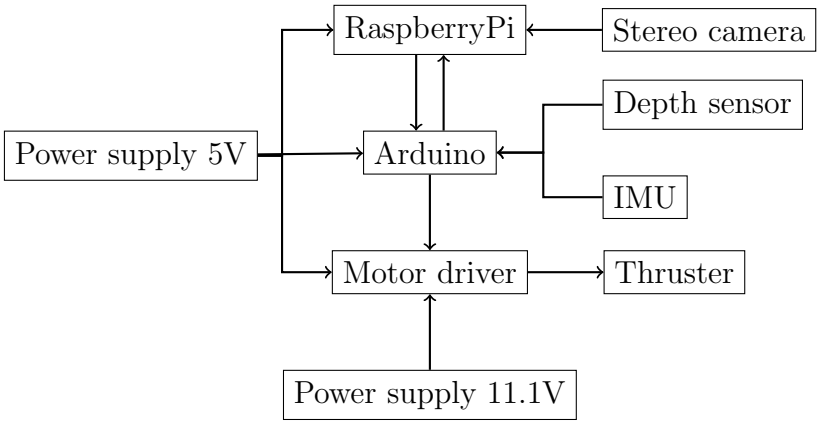


Figure 3.4: Block diagram depicting the connectivity between major components in the electronics schematics

motor and motor driver are shown in Figure 3.4, but in the actual system, there are 3 motors with dedicated motor drivers. Figure 3.4 also shows the two power supplies in the system. A detailed exposition on power supply components can be found in Section 3.3.4.

3.3.4 Power Supply

Power supply is a critical part of any robotic system. Almost all robotic systems that are mobile require batteries on board. However tethered systems like the ROVs can have an alternative resource in form of power supply through the tether. Running power through a tether comes with its own set of challenges. Direct Current (DC) transmitted at low voltages result in significant voltage drop due to the impedance of the long tether cables. Conversely using AC power at high voltages circumvents this problem but then brings the need for on board transformer/rectifier to transform the input Alternating Current (AC) power into usable DC form. The other key requirement is designing power supply systems to satisfy the loads and transients that the system can be subjected to during operation.

While designing CoopROV, several power supply systems were tested. Four different battery types were evaluated: alkaline C, lithium ion, lithium phosphate and Lithium Polymer (LiPo). Tests revealed that the high transient current requirements

Table 3.2: CoopROV Battery Specifications

Power Source	Battery Specification
11.1V	LiPo 3-cell 11.1V 5400mAh
5V	LiPo 2-cell 7.4V 800mAh

(>20A) of CoopROV were only met by the LiPo composition. Furthermore, two independent sets of LiPo batteries were used to power motors, and electronics, respectively. This physical isolation of the power supplies improved the stability of the robot's power supply system and prevented electronic components from being affected by the occasional erratic power draws of the motors.

There are thus two independent power sources in the system. One 5V power source drives all the on board electronics. The higher power demands of the motors are met by a separate 11.1V power source. The 5V power supply is regulated to protect the electronic circuits against spikes or voltage drops. There are two independent sets of LiPo batteries powering the two supplies. The 5V power supply is powered by a 7.4V battery which is stepped-down and regulated to 5V. The 11.1V power supply comes directly (unregulated) from LiPo batteries and is connected to the motor drivers. Figure 3.4 represents the different power supplies and their connection to the different components that depend on them. Table 3.2 provides information about the batteries used for these power supplies. This setup allows for adding an additional battery in parallel for both the power supplies effectively doubling the run time. The approximate running time of the ROV is 1 hour with two sets of LiPo batteries connected to both sources.

3.3.5 Sensors

CoopROV comes with 4 sensorsp; namely, stereo cameras, IMU, depth sensor and voltage sensors. A low cost off-the-shelf Minoru 3D webcam [3] was used to capture stereo images at 5 Frames Per Second (FPS) with a resolution of 320×240. The 9-DOF IMU on the robot provides instantaneous acceleration in the x,y and z directions

along with roll, pitch and yaw. The depth sensor measures the water pressure and temperature to compute the depth at which the robot is operating on. The voltage sensors are connected on to the LiPo batteries to measure and report the instantaneous battery voltage. They also double as a protection to the LiPo batteries, as they signal an alarm if the batteries drop below a preset critical voltage.

3.3.6 Software

The software of CoopROV is structured using a ROS framework. The different critical functions of the robots are sub-divided into independent modules or ROS-nodes. Each ROS-node has a specific task and can communicate with the other ROS-nodes. In CoopROV, the setup comprises a distributed ROS framework where both, the ROS-nodes running on the RaspberryPi and the surface station are connected together through a TCP link. ROS framework allows ROS-nodes either running remotely on a different system or locally on the parent system to coexist and communicate seamlessly, agnostic to where they are running. This ability of the distributed ROS system was exploited to build CoopROV's software stack. The different nodes running on the robot and the surface station along with their functionality is explained in the system diagram in Figure 3.5.

3.4 Test Infrastructure

Initial waterproofing tests were performed in the University of Delaware indoor swimming and diving pools. Later integration tests for software and electronics along with some experimental runs were performed in a circular tank in the Robotics Discovery Lab at University of Delaware.

3.5 Localization Experiments

Preliminary localization experiments were performed by placing an Augmented Reality (AR) tag on the bottom of the water tank. Images from the stereo camera was used to detect and track the AR tag. Further experiments were performed by fusing the the 6-DOF position obtained from the AR tag localization and IMU values. Since

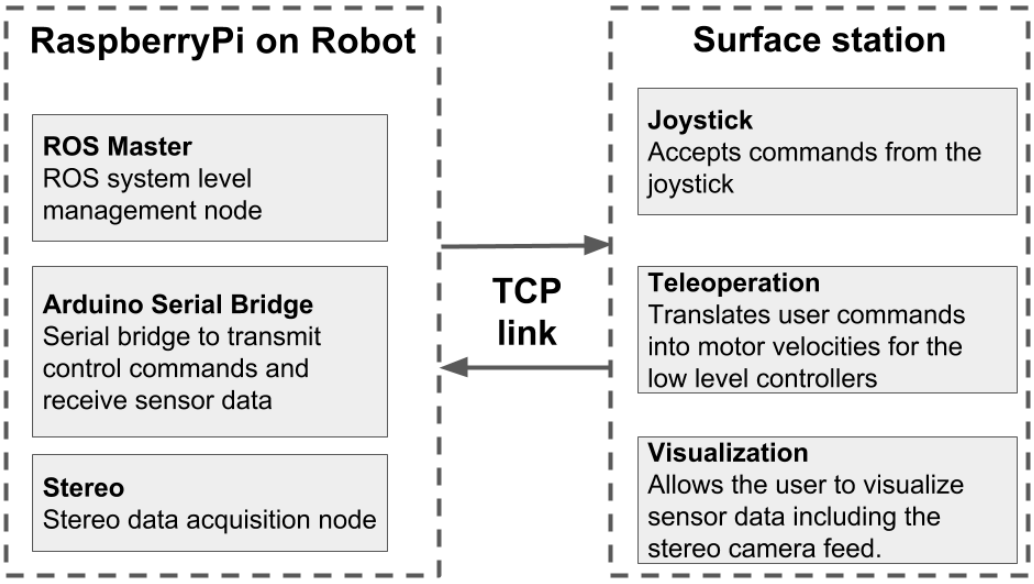


Figure 3.5: CoopROV software architecture showing all the ROS-nodes along with their functionality explained below.

there was no reliable reference measure to compare the reported position of the ROV, the quality of the results were not verified.

3.6 Conclusion

CoopROV was built as a research platform to conduct underwater object recognition and control experiments. It offers a low-cost solution with a easily reconfigurable hardware and extendable ROS-software interface. The ROS-software support allows the robot to use open source perception and control algorithms available as a part of ROS software infrastructure. The sensor package comprising a stereo camera, IMU and depth sensor can be leveraged for object recognition and motion control experiments. The capabilities of the CoopROV system can be easily expanded to suit any research experiment like the multi-layered scallop recognition 2 and the multi-image object recognition **??** discussed in this dissertation.

BIBLIOGRAPHY

[1] Arduino Mega. https://www.arduino.cc/en/Main/arduinoBoardMega. Accessed: 2016-09-20.

[2] Fathom One ROV. http://fathomdrone.com. Accessed: 2016-09-17.

[3] Minoru 3d webcam. http://www.minoru3d.com/. Accessed: 2016-09-20.

[4] Mission oriented operating suite (moos). http://www.robots.ox.ac.uk/~mobile/MOOS/wiki/pmwiki.php/Main/HomePage. Accessed: 2016-10-03.

[5] node.js. https://nodejs.org/. Accessed: 2016-09-17.

[6] OpenROV. http://www.openrov.com. Accessed: 2016-09-17.

[7] Outland ROV. http://www.outlandtech.com/product_info.php. Accessed: 2016-09-17.

[8] RaspberryPi. https://www.raspberrypi.org/. Accessed: 2016-09-20.

[9] Robot Operating System (ROS). http://www.ros.org. Accessed: 2016-09-17.

[10] VideoRay Pro 4 standard base datasheet. http://www.videoray.com/images/specsheets/2016/2016_P4STANDARDBASE_FINAL.pdf. Accessed: 2016-09-17.

[11] JF Caddy. Spatial model for an exploited shellfish population, and its application to the georges bank scallop fishery. *Journal of the Fisheries Board of Canada*, 32(8):1305–1328, 1975.

[12] N. Chernov. *Circular and linear regression: Fitting circles and lines by least squares*. Taylor and Francis, 2010.

[13] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 1, pages 886–893. IEEE, 2005.

[14] M. Dawkins. Scallop detection in multiple maritime environments. Master's thesis, Rensselaer Polytechnic Institute, 2011.

[15] M. Dawkins, C. Stewart, S. Gallager, and A. York. Automatic scallop detection in benthic environments. In *IEEE Workshop on Applications of Computer Vision*, pages 160–167, 2013.

[16] EO Guòmundsson. Detecting scallops in images from an AUV. Master's thesis, University of Iceland, 2012.

[17] Fisheries of the United States. Fisheries of the United States, Silver Spring, MD. Technical report, National Marine Fisheries Service Office of Science and Technology, 2012.

[18] C. Gordon, DL Webb, and S. Wolpert. One cannot hear the shape of a drum. *Bull.Amer.Math.Soc*, 27(1):134–138, 1992.

[19] DR Hart and PJ Rago. Long-term dynamics of us atlantic sea scallop placopecten magellanicus populations. *North American Journal of Fisheries Management*, 26(2):490–501, 2006.

[20] L. Itti, C. Koch, and E. Niebur. A model of saliency-based visual attention for rapid scene analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(11):1254–1259, 1998.

[21] M. Kac. Can one hear the shape of a drum? *American Mathematical Monthly*, pages 1–23, 1966.

[22] P. Kannappan and HG Tanner. Automated detection of scallops in their natural environment. In *21st Mediterranean Conference on Control & Automation, 2013*, pages 1350–1355. IEEE, 2013.

[23] P. Kannappan, HG Tanner, AC Trembanis, and JH Walker. Machine learning for detecting scallops in AUV benthic images: Targeting false positives. *Computer Vision and Pattern Recognition in Environmental Informatics*, page 22, 2015.

[24] P. Kannappan, JH Walker, AC Trembanis, and HG Tanner. Identifying sea scallops from benthic camera images. *Limnology and Oceanography: Methods*, 12(10):680–693, 2014.

[25] MA Khabou, L. Hermi, and MBH Rhouma. Shape recognition using eigenvalues of the dirichlet laplacian. *Pattern Recognition*, 40(1):141–153, 2007.

[26] C. Koch and S. Ullman. Shifts in selective visual attention: towards the underlying neural circuitry. *Human Neurobiology*, 4(4):219–227, 1985.

[27] S. Loncaric. A survey of shape analysis techniques. *Pattern Recognition*, 31(8):983–1001, 1998.

[28] KS Naidu and G. Robert. Fisheries sea scallop placopecten magellanicus. *Developments in Aquaculture and Fisheries Science*, 35:869–905, 2006.

[29] V. Navalpakkam and L. Itti. An integrated model of top-down and bottom-up attention for optimizing detection speed. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 2, pages 2049–2056. IEEE, 2006.

[30] DNREC State of Delaware Department of Natural Resources and Environmental Control. 44 more subway cars sunk at redbird reef. *News of the Delaware Department of Natural Resources and Environmental Control*, 39(182), 2009.

[31] DNREC State of Delaware Department of Natural Resources and Environmental Control. *Delaware Reef Guide*, 4050203 FW FISHERIES, 2009-2010.

[32] NA Raineault, AC Trembanis, DC Miller, and V. Capone. Interannual changes in seafloor surficial geology at an artificial reef site on the inner continental shelf. *Continental Shelf Research*, 58:67–78, 2013.

[33] AA Rosenberg. Managing to the margins: the overexploitation of fisheries. *Frontiers in Ecology and the Environment*, 1(2):102–106, 2003.

[34] Y. Rubner, C. Tomasi, and LJ Guibas. The earth mover's distance as a metric for image retrieval. *International Journal of Computer Vision*, 40(2):99–121, 2000.

[35] FM Serchuk, PW Wood, JA Posgay, and BE Brown. Assessment and status of sea scallop (placopecten magellanicus) populations off the northeast coast of the united states. In *Proceedings of the National Shellfisheries Association*, volume 69, pages 161–191, 1979.

[36] J. Shi and J. Malik. Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8):888–905, 2000.

[37] G. Taubin. Estimation of planar curves, surfaces, and nonplanar space curves defined by implicit equations with applications to edge and range image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(11):1115–1138, 1991.

[38] AC Trembanis, C. DuVal, J. Beaudoin, V. Schmidt, D. Miller, and L. Mayer. A detailed seabed signature from hurricane sandy revealed in bedforms and scour. *Geochemistry, Geophysics, Geosystems*, 14(10):4334–4340, 2013.

[39] JH Walker. Abundance and size of the sea scallop population in the mid-atlantic bight. Master's thesis, University of Delaware, 2013.

[40] D. Walther and C. Koch. Modeling attention to salient proto-objects. *Neural Networks*, 19(9):1395–1407, 2006.

[41] M. Zuliani, L. Bertelli, CS Kenney, S. Chandrasekaran, and BS Manjunath. Drums, curve descriptors and affine invariant region matching. *Image and Vision Computing*, 26(3):347–360, 2008.

Acronyms

AC Alternating Current. 48

API Application Program Interface. 42, 43

AR Augmented Reality. 50

AUV Autonomous Underwater Vehicle. 14, 41, 42

CAD Computer Aided Design. x, 45, 47

DC Direct Current. 48

DOF Degree of Freedom. 45, 49, 50

DVL Doppler Velocity Log. 15

DYI Do-It-Yourself. 42

emd Earth Mover's Distance. 34, 35

FPS Frames Per Second. 49

hog Histogram of Gradients. ix, 30, 32–36, 39

HOV Human Operated Vehicle. 41

IMU Inertial Measurement Unit. 45, 46, 49–51

INS Inertial Navigation System. 15

LiPo Lithium Polymer. 45, 48–50

MOOS Mission Oriented Operating Suite. 43

ROS Robot Operating System. x, 43, 44, 46, 50, 51

ROV Remotely Operated Vehicle. 41, 42, 44, 48, 49, 51

RSA Research Set-Aside. 9, 15, 19

RST Rotation, Scaling and Translation. 2, 6

SVM Support Vector Machine. 34

TCP Transmission Control Protocol. 46, 50

TDVA Top-Down Visual Attention. 10, 13, 19, 22

wctm Weighted Correlation Template Matching. ix, 30, 32, 33, 35–37, 39