OBJECT RECOGNITION IN NOISY NATURAL IMAGES

by

Prasanna Kannappan

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 Mechanical Engineering

Winter 2017

© 2017 Prasanna Kannappan
All Rights Reserved

OBJECT RECOGNITION IN NOISY NATURAL IMAGES

by

Prasanna Kannappan

Approved: _____
Suresh G. Advani, Ph.D.
Chair of the Department of Mechanical Engineering

Approved: _____
Babatunde A. Ogunnaike, Ph.D.
Dean of the College of Engineering

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: _____

Herbert G. Tanner, Ph.D.
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: _____

Ioannis Poulakakis, Ph.D.
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: _____

Christopher Rasmussen, Ph.D.
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: _____

Guoquan Huang, Ph.D.
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: _____

Arthur Trembanis, Ph.D.
Member of dissertation committee

ACKNOWLEDGEMENTS

First and foremost, I would like to thank my advisor Dr. Herbert Tanner, for his patience and valuable guidance throughout my PhD. The knowledge and systematic outlook he instilled in me, was instrumental in enabling me to finish this dissertation. I would like to thank Dr. Tanner for exposing me to a multitude of inter-disciplinary projects, that enabled me to gain valuable expertise for building commercial robotic solutions.

I would like thank members of my PhD committee: Dr. Ioannis Poulakakis, Dr. Christopher Rasmussen, Dr. Guoquan Huang, Dr. Arthur Trembanis and Dr. Sunil Agarwal for their comments and advise that helped me channel my dissertation in the right direction. I would also like to thank Dr. Christopher Rasmussen for mentoring me during my computer science masters and introducing me to building robotic systems through Robot Operating System (ROS). I would like to thank Dr. Jeffrey Heinz and Dr. Adam Jardine for their valuable support in collaborative projects that exposed me to concepts in computational linguistics.

I would like to thank members of Coastal Sediment, Hydrodynamics, and Engineering Laboratory (CSHEL) at University of Delaware for the helping me with data that was utilized in this disseration. It was a great pleasure learning the way of life in Sea from them (even though I got sea sick the first time I was on a ship). I would like to thank Dr. Art Trembanis for allowing me to use the scallop data collected during his research projects and for opening to me the facilities in College of Earth and Ocean sciences to support my experiments. Special thanks to Hunter Brown, lab manager at Robotic Discovery Laboratories (RDL) for providing me access to test facilities in Lewis, Delaware. I would like to mention my thanks to Justin H. Walker for providing me the annotated scallop data which was critical to my experiments. Thanks to Carter

DuVal for building with me my first unsuccessful scrappy Remotely Operated Vehicle (ROV). The ROV building painfully continued a few years before getting to the point of a working prototype. Thanks also to other graduate students of Dr. Matthew Oliver and Dr. Mark Moline who provided support during experiments at Lewis. Special thanks to Scott Gallager and Amber York of Woods Hole Oceanographic Institution for sharing the datasets on which Matthew Dawkings and Charles Stewart of RPIs Computer Science Department worked on. The building of CoopROV took a lot of lives, in this case, apart from mine, I would like to thank undergraduate interns Yehan Zhang, Yang Liu and Austin Crouse for their willing sacrifice.

Thanks to all the funding agencies National Science Foundation (NSF), Army Research Lab (ARL) and National Institute of Health (NIH) that graciously supported part of my work over the years through grants NSF RI #0913015, NSF CPS #1035577, NSF EAGER #1548149, NIH #R01HD087133 and ARL MAST CTA #W911NF-08-2-0004.

Thanks to all my lab mates for stimulating discussions and support through out this PhD, with special mention to Dr. Jie Fu, Dr. Shridhar Shah, Luis Valbuena Reyes, Aditya Boddu, Dr. Konstantinos Karydis, Dr. Jianxin Sun and Adam Stager. Thanks also to all other university faculty, staff and fellow students who had supported me in various ways.

Finally thanks to my father A.S. Kannappan, mother PL Meenakshi and brother K. Priyadharshan for their relentless moral support throughout this PhD. Thanks to my girlfriend Karthika Ramanathan for motivating me get through this dissertation. Thanks to all other friends, hiking buddies and fellow chess players who made this PhD bearable through intermittent distractions.

TABLE OF CONTENTS

LIST OF TABLES . . . . . **xi**
LIST OF FIGURES . . . . . **xii**
ABSTRACT . . . . . **xiv**

Chapter

1 INTRODUCTION **1**

1.1 Benefits of Robotic Systems 1
1.2 Tele-operation Solutions 2
1.3 Components of Automated Informed Decision Making 3
1.4 Automated Decision Making in Natural Environment 4

1.4.1 Natural Environment 4
1.4.2 Robotic Systems in Natural Environments 4

1.5 Noise in Sensor Measurements 6

1.5.1 Noise Sources 6
1.5.2 Noise Filtering 8

1.6 Object Recognition in Natural Environments 9
1.7 Object Recognition as a path to Robot Autonomy 11
1.8 Approach Overview 12
1.9 Dissertation Organization 13

2 EIGEN-VALUE BASED SHAPE DESCRIPTORS **14**

2.1 Introduction 14
2.2 Background 14
2.3 Preliminaries 15
2.4 Subway-car Detection from Seabed Images 16
2.5 Discussion 19
2.6 Conclusion 21

3 MULTI-LAYERED SCALLOP RECOGNITION FRAMEWORK 22

3.1 Introduction . . . 22
3.2 Background . . . 23

3.2.1 Underwater Animal Recognition . . . 23

3.2.1.1 Methods for Recognition of Moving Underwater Organisms . . . 24
3.2.1.2 Methods for Recognition of Sedentary Underwater Organisms . . . 25
3.2.1.3 Scallop Recognition Methods . . . 26

3.2.2 Motivation for a Generalized Automated Object Recognition Tool . . . 28

3.3 Preliminaries . . . 30

3.3.1 Visual Attention . . . 30

3.4 Problem Statement . . . 32
3.5 Scallop Survey Procedure . . . 34

3.5.1 Field Survey Process . . . 35
3.5.2 Sensors and Hardware . . . 36
3.5.3 Data Collection . . . 37

3.6 Methodology . . . 38

3.6.1 Layer I: Top-Down Visual Attention . . . 38

3.6.1.1 Learning . . . 38
3.6.1.2 Implementation and Testing . . . 42

3.6.2 Layer II: Segmentation and shape extraction . . . 43
3.6.3 Layer III: Classification . . . 44

3.6.3.1 Scallop Profile Hypothesis . . . 45
3.6.3.2 Scallop Profile Learning . . . 46

3.6.3.3 Scallop Template Matching 47

3.6.4 Layer IV: False Positives Filter 49

3.6.4.1 High-dimensional weighted correlation template matching (WCTM) 49
3.6.4.2 Histogram of Gradients (HOG) 51

3.7 Results 54
3.8 Discussion 56
3.9 Conclusions and Futurework 58
3.10 Future Work 58

4 DISTANCE BASED GLOBAL DESCRIPTORS FOR MULTI-VIEW OBJECT RECOGNITION 60

4.1 Introduction 60
4.2 Preliminaries 63

4.2.1 Grabcut-in-one-cut Algorithm 63
4.2.2 Data Collection Apparatus 64

4.2.2.1 Imaging Rig 64
4.2.2.2 Imaging Environment 64
4.2.2.3 Object Specimens 65

4.3 Definitions 66

4.3.1 Histogram Signature 66

4.4 Methodology 67

4.4.1 Data Collection 68
4.4.2 Annotation Process 69
4.4.3 Learning 75

4.4.3.1 Feature Distribution 77
4.4.3.2 Feature Weights 80

4.4.4 Validation 83

4.5 Results 85
4.6 Discussion 89

4.7 Conclusion . . . 92
4.8 Future work . . . 92

5 COOPROV: LOW COST UNDERWATER REMOTELY OPERATED VEHICLE . . . 93

5.1 Introduction . . . 93
5.2 Commercial Submersible Robots . . . 94
5.3 CoopROV . . . 95

5.3.1 System Overview . . . 96
5.3.2 Hardware . . . 97
5.3.3 Electronics . . . 98
5.3.4 Power Supply . . . 100
5.3.5 Sensors . . . 101
5.3.6 Software . . . 102

5.4 Test Infrastructure . . . 102
5.5 Localization Experiments . . . 102
5.6 Conclusion . . . 103

6 CONCLUSIONS AND OUTLOOK . . . 104

BIBLIOGRAPHY . . . 108

LIST OF TABLES

3.1 List of missions and number of images collected 39

3.2 Camera specifications 40

3.3 Top-down weights for feature maps 42

3.4 Comparison of tested false positive filter layer methods 54

3.5 Results of multi-layer scallop classification 55

5.1 CoopROV Specifications 97

5.2 CoopROV Battery Specifications 101

LIST OF FIGURES

2.1 Analysis of sonar backscatter image of Redbird reef 17

2.2 Plot of similarity measures of different underwater objects to the subway car reference profile 18

2.3 Subway car recognition results 20

3.1 Seabed image with scallops shown in red circles 33

3.2 Scallop features 33

3.3 Map of the scallop survey region 35

3.4 Schematics and image of the Gavia AUV 37

3.5 Illustration of computation flow for the construction of saliency maps 41

3.6 Illustration of fixations 41

3.7 Scallop fixation window dimension analysis 43

3.8 Segmentation layer process flow 43

3.9 Mean and variance map of scallops in each quadrant 46

3.10 Illustration of scallop profile hypothesis 47

3.11 Template matching masks 48

3.12 Precision-Recall curves for Classification Layer 49

3.13 Precision-Recall curves for False-positive Filtering Layer 51

3.14 Comparision of scallop recognition results with existing methods . . 57

4.1 Imaging rig used for data collection in multi-image object recognition approach . . . 65

4.2 Illustration of the inter-class and intra-class variance exhibited by naturally occurring objects . . . 66

4.3 Set of images collected for each specimen from different heights . . 69

4.4 Images of a strawberry specimen after cropping a portion of background . . . 70

4.5 Images of an orange specimen after cropping a portion of background 70

4.6 Illustration of annotation process flow . . . 73

4.7 Illustration of a case where automated annotation process fails . . . 74

4.8 Annotation output for a single specimen . . . 76

4.9 Feature distribution . . . 81

4.10 Feature weights . . . 83

4.11 Strawberry specimen validation results . . . 87

4.12 Orange specimen validation results . . . 88

4.13 Classification results . . . 90

5.1 CoopROV . . . 96

5.2 CoopROV data-flow diagram . . . 97

5.3 CoopROV CAD model . . . 99

5.4 CoopROV electronics schematics . . . 100

5.5 CoopROV software architecture . . . 103

ABSTRACT

Building autonomy into robotic systems enables them to operate unsupervised over remote or potentially dangerous domains. Object recognition is an important trait required for a robotic system to achieve autonomy. The task of object recognition is essentially understanding and labeling the different components in a robot's environment. This task can get complicated for robots that operate in unstructured natural environments, like forests or deep sea, due to noise in sensor measurements. Noisy sensor measurements can potentially affect a robot's perception of the world. To avoid being misled by noise in sensor measurements, robots need to possess robust object recognition capabilites that can handle noise in sensor measurements. Such robust object recognition capabilities are valuable for processing large natural image datasets. Underwater image datasets gathered by marine scientists and oceanographers, is one such case where object recognition capabilities could be invaluable. The objective here is to analyse these datasets to understand natural phenomenon, for instance recognize organisms. Sifting through such *big* datasets, ranging millions of images, to make inferences is growing to be big challenge to the research community. This motivates the need for automated object recongition and image analysis tools. This dissertation focusses on object recognition techniques capable of operating in noisy natural environments. A series underwater object recognition problems have been solved as means to validate the developed object recognition algorithms. Each technique was developed to complement the shortcomings of the existing tools available to the research community.

To start with, eigen-value based shape descriptor were tasked to solve the subway car recognition problem. Despite the success in solving this problem, eigen -value shape descriptor cannot leverage textural cues for object identification. This primary

drawback, among other shortcomings, lead to the development of a multi-layered object recognition architecture. This multi-layered architecture was tested on an scallop enumeration problem. Over 60% of scallop instances were successfully identified. To improve the machine learning classifier of this multi-layered framework, and also to minimize false positives, a multi-view object classication approach is proposed. This multi-view approach combines histogram-based global cues from a series of images of a target, captured from different heights, to construct a machine learning classifier. This multi-view method was successful in classifying all specimens in the available dataset. In addition to the developed object recognition methods, a low cost ROV, named CoopROV, was designed for underwater data collection to support research experiments.

Chapter 1

INTRODUCTION

1.1 Benefits of Robotic Systems

Robotic systems are beneficial for tasks that come under the purview of the *3D*'s–dirty, dull and dangerous. Currently manufacturing and assembly lines, where tasks of repetitive nature, otherwise considered dull, are places where robotic systems are ubiquitous. Tasks like traversing tunnels or sewers, dirty in nature, are being slowly delegated to tele-operated robots. The other avenue where robotic systems are being promoted relate to activities deemed dangerous for humans like deep sea exploration, operation in radiation prone zones, or war zones. With enabling new advances in the field of robotics, robots are beginning to take over new roles to complement and assist humans in various ways.

Though advances in robotic systems have enabled robots to outperform humans in certain specialized tasks, such as competing in games like Go [1] and Chess [2], robotic systems still lack the ability to reason and operate autonomously in most unpredictable real world environments. In such cases, a human expert is deemed necessary to make decisions for the robot and guide the machine to accomplish its objectives. Another domain where tele-operation is prevalent is in operations over inhospitable environments without exposing humans to risk. Some prime examples of such remote operation in dangerous environments include the rescue and repair effort at radiation affected zones in Fukushima Daichii nuclear power plant [3], or combat operations via remotely operated weapons like Packbot [4] and Predator [5] that have been deployed at war-zones in Iraq and Afghanistan. Though tele-operation appears to offer a solution, a more scalable alternative is aiming for full autonomy of robotic systems to minimize the strain on human workforce.

1.2 Tele-operation Solutions

Tele-operation is widely prevalent in robots as it offers an avenue for humans to leverage the benefits posed by a robotic system to perform activities that are *dirty* or *dangerous* without being physically present on the site of operation. During tele-operation, the human operator typically operates the robot from a safe distance away from the point of activity of the robot. This safe distance could range from a few meters to over a few hundred kilometers based on the nature of the activity. For instance, in case of a tunnel or sewer inspection, the operator can stay a few meters outside the structure and drive a robot that is traversing inside the structure. On the other hand, in case of weaponized drones like the predator [5] that operate in war zones, the operator typically controls the drone from a safe remote command center, a few hundred kilometers from the point of action of the drone. Tele-operation allows the operator to consume the sensory information collected by the robot and make *informed decisions* for the robot. The robot then the translates *informed decision* into a sequence of actions. The tele-operation solution allows a human operator to safely accomplish his or her objective without being in state of physical discomfort or harm.

Tele-operation allows the durability of a robotic agent and intellect of a human to provide a powerful solution to solve problems ranging from deep sea exploration to safe operation in combat zones. Despite the attractiveness of the tele-operation solution, it still requires constant attention from a human operator. Additionally, since the human operator is not present on site during tele-operation, a reliable communication channel is essential to remotely control the robot. If the communication links are unreliable, or subject to willful sabotage by an enemy in a war-zone, it could potentially lead to disconnection of the robotic system from the human operator. Such communication interruptions could possibly result is catastrophic failure of the robotic system. The constant need for a human operator and the need for reliable communication channels are some of the shortcomings that plague tele-operated systems.

If the objective is to move towards decreasing the strain on human workforce, it is imperative to decrease the reliance of robots on humans for operation. This

emphasizes the need for automated decision making in robotic systems. Furthermore, building automated decision making capability into robotic systems also overcomes the need for reliable communication channels. If a robot is capable of making decisions without any guidance from an external source, the need for constant communication is no longer a requirement. Thus improving the cognitive capabilities to enable automated decision making is essential to build robust self-sufficient robotic systems.

1.3 Components of Automated Informed Decision Making

The first component of informed decision making is gathering the information needed to make a decision. For a robotic system information about the environment and the objective it needs to accomplish dictate the decisions it makes. Collecting sensory data is the prime and often the only mechanism available for a robot to learn about its environment. Based on the sensors available to a robotic system, different measurements describing the state of the environment are available. This raw sensor data needs to be processed to get usable information that enable a robot to reason about the state of its environment. An informed decision is nothing but extrapolation of the reasoning process to choose an action from the set of actions available to an agent. If the entire sequence of sensory data aggregation to choosing an action based on processed information is performed by a robotic system without any external intervention, the due process can be described as automated informed decision making.

Transforming sensory data into usable information involves parsing the data to gain a semantic understanding about the state of an environment. The basic stage in understanding a scene[1] is separation and labeling of different objects present. A more refined version of this understanding is deciphering the relationship between the objects in a scene. Such a detailed semantic understanding allows a robot to infer the ability of the scene elements to actively participate in the robot's planned actions.

The ability of a robot to detect and label objects in its environment is an important step towards gaining semantic understanding of its environment. Since semantic

[1] Scene here is defined as the state of an environment at a given point in time.

understanding of a scene dictates the ability of a robotic system to make informed decisions, object recognition capability is central to automated informed decision making.

1.4 Automated Decision Making in Natural Environment

Building robotic systems that are fully autonomous with an ability to make informed decisions based on their observations under all possible environmental conditions is an open problem. The existing solutions typically apply to certain specialized domains only. Before further discussion on this regard, it is important to understand that natural environments pose significant challenges that compound the problems faced by robotic systems. In the remainder of this section, (i) Section 1.4.1 states the definition of natural environment in the context of this dissertation and, (ii) Section 1.4.2 goes into the details on the challenges faced by robotic systems in natural environments.

1.4.1 Natural Environment

The word *Natural Environment* in this dissertation refers to uncontrolled real world environments depicting scenes containing predominantly naturally occurring objects, where the robot has no direct control over any environment variable. However we assume that the robot can use accessories, like artificial light sources, attached to it to influence the state of the environment and thereby enhance sensor measurements. Due to the wide variability in natural environments, it is challenging to model such environments. The task of modeling gets further complicated in the presence of unpredictable levels of noise. The natural environments are typically challenging to model due to the unpredictability in the environment parameters and unknown noise parameters. As per this definition of natural environment, deep sea or forest landscapes with vegetation are some prime examples that come under the defined category.

1.4.2 Robotic Systems in Natural Environments

Robots in the past, have been specialized to do tasks in well defined environments like assembly lines where environmental variables (e.g. visibility and lighting)

are strictly controlled. If such environment variables are known in advance, a mathematical model that maps robot actions to finite known environment states can be built. In such cases, based on the state of the environment observed, a robot action that complies with a predefined database of logical rules is chosen. Systems that operated on this principle, *Expert Systems* [6], came into existence in 1980's. Such systems fail in scenarios where the environment cannot be perfectly modeled. Despite the advances in field of robotics since the time of Expert Systems, robotics systems still face challenges when dealing with unpredictable environments in the likes of unstructured natural settings seen in forests or deep sea.

The task of understanding natural scenes and recognizing objects like animals from them is shown to be a cognitively challenging task even for humans [7]. The unstructured nature and unpredictability associated with natural environment makes the task of building models to capture natural scenes challenging. Using global features to characterize the nature of a scene is shown to be possible [8]. However these models can only offer a high level understanding of the scene. For instance, the high level understanding translates to differentiating widely different scenes like a beach, foliage or a busy city street. It does not talk about separating the different components in the scenes. Most object recognition techniques depend on features like edges or texture to identify objects. Such feature-based approaches are difficult to implement in natural scenes where the edges of objects can be difficult to distinguish. The noise in sensor measurement in such natural scenes could further complicated the object recognition task. Thus, for a robotic system to be successful in making automated informed decisions in natural environments, it has to possess robust object recognition capabilities that can handle noise and variations in environmental conditions.

The challenges associated with object recognition in natural environments often hamper the ability of robotic systems to understand their surroundings. Difficulty in understanding or reasoning about their surroundings in robotic systems can lead insufficient information to make automated informed decisions. In case of such failure

to understand its surrounds, a robotic system can resort to human support via teleoperation or act dubiously based on the limited knowledge it has inferred. Thus, for a robotic system to successfully operate in natural environments, its critical for the system to be able to recognize objects despite unpredictable environmental conditions or noisy sensor measurements.

1.5 Noise in Sensor Measurements

Sensors are designed to measure a physical quantity. This physical quantity is captured in form of a signal by the sensor. The measurement reported by the sensor almost always does not match the real value of the physical quantity. This error in the measured value is defined as the noise in the measurement reported by the sensor. The ubiquitous presence of noise in almost all sensor measurements has spurred a field of study in the name of *Filter Theory* [9]. Filter theory deals with estimation of signals from noisy data. If the source and nature of noise that affects measurements of a sensor can be modeled, then specialized filters can be designed to recover the underlying signal from noisy sensor measurements. The challenge here is in identifying the noise sources and modeling them. The sources of noise in sensor data can be manifold. An exhaustive discussion on sources of noise and methods to model them is beyond the scope of this dissertation. In the remainder of this section, we will focus on specific sources of noise that affect imagery data collected by robots operating in natural environments (Section 1.5.1) and some filters that can help recover the underlying signal in these cases (Section 1.5.2).

1.5.1 Noise Sources

The sources of noise associated with an application need to be carefully analyzed before designing appropriate filters to mitigate this noise. In the specific case of imaging applications that employ a camera sensor in natural environments, the various noise sources can be broadly divided into (i) noise inherent to the sensor, (ii) noise associated with data collection setup and (iii) noise introduced by the environment. For instance,

in case of an underwater imaging application using an Autonomous Underwater Vehicle (AUV), the noise associated with the imaging setup constitutes noise introduced due to vibration or motion of the AUV. Additionally, the environmental parameters associated with deep sea environments could introduce noise which could further degrade the sensor measurements. Further details about the different noise sources is discussed in the rest of this section.

In an imaging application, some noise is inherent to the camera sensor. A previous study [10] has analyzed the various constituents of noise in a Charge-Coupled Device (CCD) video-camera. According the this study, the sources of noise in a camera can further be divided into 3 categories, namely, illumination independent noise, illumination dependent noise and digital processing noise. After further analysis, this work suggests that there are 4 major noise sources that contribute to the camera sensor noise. Out of these 4 sources, *readout noise* and *fixed pattern noise* are constant illumination independent noise sources. One of the factors that determine the value of readout noise and fixed pattern noise is the temperature of the sensor. The remaining two primary contributing noise sources, *photo-shot noise* and *photo-response non-uniformity noise* are illumination dependent noise sources. Its important to note that, illumination dependent noise sources implies that the value of noise is a function of the intensity value of the pixel. Modeling all the noise sources for different camera sensors poses a challenge in itself.

Another source of noise is the physical setup which holds the camera. This setup could be an AUV or just a stationary imaging rig. Such physical structures that hold the camera can introduce noise in form of vibration or motion of the structure during imaging. Such motion could lead to out-of-focus or blurred pictures. There could also be cases where the physical setup can produce artifacts like shadows that could affect the sensor measurements. The few examples mentioned here are not all encompassing. Each imaging setup needs to be studied to identify the possible sources of noise it can introduce into sensor measurements.

Environmental conditions are one of the chief sources of noise. There is a key

difference between noise due to environmental conditions and other sources like sensor noise or the imaging setup noise: the former is relatively difficult to model due to wide range of variables associated with environment compared to the latter. These environmental variables are fairly limited in controlled environments, but in case of natural environments the number of variables are far higher. Furthermore, the ability to predict or model the environmental variables in natural environments is far more challenging compared to controlled environments. To understand some of the variables associated with natural environments, lets consider the case of imaging experiments in sub-sea environments. Some of the variables at play here are non-uniform low illumination, absorption of light by water and scattering of light by different particles in water. Each of these parameters can have considerable impact on the quality of the image acquired by a camera sensor. Modeling these environment variable and accounting for the noise induced by them in the sensor measurements is critical to extract any useful information from the sensor data.

Any object recognition method has to account for the different noise sources that can affect sensor measurements. Especially in case of natural environments, environmental conditions typically act as major noise sources. In any case, its important to design appropriate ways to mitigate the effect of noise in the sensor measurements before useful information can be extracted. Even in the absence of accurate models of different noise sources, it is important to attempt to model the noise seen in the sensor data and devise filters to extract the underlying signal from recorded sensor measurements.

1.5.2 Noise Filtering

One of the prerequisites for extracting information from noisy sensor data. Modeling the noise sources is followed by filter design. The sensor noise model is often a part of the sensor specification provided by the manufacturer. Thus the person integrating the sensor into a custom application, can avoid modeling the sensor noise. However, the noise induced by the setup is unique to each system and hence needs

to be analyzed to identify the characteristics of noise. Once a model is identified, the filter design involves designing a filter that can recover the signal from recorded sensor measurements.

The task of filtering noise induced by the environment is more complex relative to the sensor noise or the setup noise. The reason for this is the difficulty in modeling environmental variables. In case of underwater imaging applications, understanding the physics of light propagation, that includes phenomenon like absorption and scattering, is required. Some proposed models [11, 12] try to explain light propagation in water and also suggest filter models that can correct illumination and color in noisy underwater images. A more detailed survey by Jaffe [13] describes different existing filter models available to deal with noise in underwater images. In most cases, modeling all the environment parameters is intractable. Hence identifying the dominant noise source or simply identifying the nature of noise affecting the images are possible alternatives. For instance, in the scallop recognition work [14], the AUV images used were corrupted by speckle noise, hence a median filter [15] was used to minimize noise without elaborate modeling of noise sources.

Since extracting useful information from sensor data is essential for object recognition, noise filters constitute a core component of object recognition systems. The first stage in noise filtering involves identifying noise sources and modeling them. Accurate modeling of noise sources in often challenging, hence most systems use simplified models that only capture dominant noise sources. Since there are several filtering mechanisms available, filter design often involves human intuition to pick appropriate filters that can handle the noise pattern seen. Ultimately, noise filtering allows distilling useful information for informed decision making.

1.6 Object Recognition in Natural Environments

For an object recognition framework to be effective in natural environments, it should be robust to noise and variations in environmental conditions. To avoid dealing with uncertainties in environmental conditions, researchers tend to specialize their

object recognition technique to work in certain controlled conditions. In some instances even the sensing apparatus used can be explicitly designed to detect a certain organism as in the case of plankton recognitions systems [16, 17]. Such techniques that are highly specialized to solve a specific problem often do not transfer to other application domains. An example of object recognition in controlled conditions is the scallop recognition system designed to work in artificial scallop beds [18, 19]. In this method, a series of stationary cameras under known conditions are used to perform object recognition. The object recognition systems built on the assumption of controlled environmental conditions, generally do not translate well to natural environments that exhibit wide variation in environmental conditions.

Building specialized object recognition frameworks that translate between different application domains is challenging. At the same time, it is equally challenging to build a generalized object recognition method that can accommodate wide variations in environmental conditions. One solution to this problem is the use of Multi-layered object recognition frameworks. Multi-layered object recognition frameworks with dedicated modules for dealing with different sub-problems like filtering and hypothesis testing allow easy customization of layers based on application requirements. Multi-layered frameworks allow researchers to configure specialized layers that are custom designed for their object recognition problem. The scallop recognition framework [14, 20] offers one such multi-layered framework. Another interesting example is the set of underwater object recognition tools [21] used for recognizing organisms like sea-anemones.

In conclusion, multi-layered object recognition frameworks offer a flexible solution to researchers to solve automated object recognition problem. Customizing a multi-layered object recognition framework to work with a specific problem amounts to picking the right layers that would be effective in solving the problem. The layers used can be adaptations of existing computer vision or machine learning tools. For cases where there is no such off-the-shelf solution, specialized layers can be engineered to fit the problem. A researcher can thus build a multi-layered framework with existing tools in tandem with custom designed solutions to solve specialized object recognition

applications.

1.7 Object Recognition as a path to Robot Autonomy

Object recognition capability allows a robotic system to identify and label the different components in its environment. Such understanding of individual objects in its surrounding is required to build a semantic model that captures the relationship between the different components in an environment. Cognitive reasoning relies on such semantic models to determine the best possible action for an agent to undertake given its knowledge about the state of the environment. Thus we can conclude that object recognition capabilities are essential to build robotic systems capable of automated informed decision making.

The task of object recognition and understanding the state of the environment becomes challenging in the presence of noise. Sensor measurements that are supposed to measure the state of the environment could be corrupted due to the presence of noise. In order to avoid being misled by bad measurements, a robotic agent needs to employ noise filters to extract the underlaying signal from sensor measurements. Since sensor measurements in natural environments are typically corrupted by noise, noise filters become a core component of the perception system that aggregates sensor measurements to gain knowledge about the state of the environment. Building accurate models of the surroundings despite the presence of noise is key to the success of a robotic agent that is intended to operate autonomously in natural environments.

In conclusion, object recognition capabilities, with robustness to noise, is required to build systems that operate in natural environments. Object recognition coupled with an ability to model and reason about the state of an environment allows a robotic system to make automated informed decisions. Automated informed decision making in conjunction with autonomous actuation paves the way for a robot autonomy. Autonomous systems can act independently without any human support in any environment. Such systems can ease the strain on human work force by taking up tasks that are physically challenging or risky for a human to perform.

1.8 Approach Overview

Since we have established that object recognition in noisy environments is critical for robot autonomy, this dissertation will propose a series of techniques designed to solve the object recognition problem in noisy environments. Furthermore, since operation in natural environment is typically characterized by noisy sensor measurements, we choose underwater object recognition as the primary domain to conduct the experimental validation for all these approaches. The rest of this section will offer a brief overview of the different object recognition techniques developed as a part of this dissertation along with insights on the problems they address.

Recognizing submerged subway cars from seabed images was the first application that initiated the need for an object recognition technique capable of operating in noisy seabed images. Since recognizing subway cars can be reduced to detecting rectangles, eigen-ratio based shape descriptors with in-built Rotation, Scaling and Translation (RST) invariance was the candidate technique tested to solve this specific object recognition problem. Eigen-value based shape descriptors were successful in detecting simple shapes like rectangles from images. During the course of this experiment, several shortcomings of eigen-value shape descriptors were exposed: Poor performance in noisy domains, or inability to identifying objects characterized by complex shape profiles. Chapter 2 of this dissertation offers a detailed treatment of the eigen value shape descriptors and the results of the subway car recognition problem they were tested on.

When the scope of the object recognition problem was expanded from simple shapes like rectangles to underwater organisms characterized by complex profiles, the shortcomings of eigen-value shape descriptors became apparent. The primary application domain that drove the design of the multi-layered object recognition system in Chapter 3 was the need for scallop recognition system capable of automated scallop enumeration to support image-based benthic survey efforts. The multi-layered object recognition system thus designed was capable of recognizing objects from noisy natural images was proposed. The method was detected 60-75% of scallops. One important point here is that the method was expressly designed to deal with noise introduced

by non-uniform lighting, low resolution images and large levels of speckle noise, that previous scallop recognition efforts were not equipped to handle. Chapter 3 talks in detail about this multi-layered object recognition framework.

Despite the ability of this multi-layered object recognition framework proposed in 3 to detect objects from noisy images, there were several instances of false positives in the ensuing detections. This prompted the need for a classification technique that can recognize object will lesser false positives. One way to accomplish this was to merge information from multiple views of an object before determining its identity. The details involved in this multi-view object recognition approach is discussed in Chapter 4.

During the development of these object recognition algorithms, the need for a low-cost research platform that can be used as a test bed to evaluate different object recognition algorithms was realized. In response to this, CoopROV, a low cost underwater ROV was developed. CoopROV's specifications and design procedure is discussed in Chapter 5.

1.9 Dissertation Organization

The rest of this dissertation is organized as follows (i) Chapter 2 discusses the eigen-value shape descriptors and the subway car detection problem they were evaluated on. (ii) The multi-layered object recognition technique developed to support an automated scallop survey effort is discussed Chapter 3. (iii) This is followed by Chapter 4 that discusses a multi-view object recognition technique that allows to combine information from multiple views of a target object, designed to decrease the number of false positives seen in the multi-layered object recognition technique. (iv) Chapter 5 provides details on the underwater ROV designed as a research prototype to test object recognition algorithms. (v) Finally, Chapter 6 expresses the insights gleaned during the development of the different object recognition techniques along with possible future directions that can be explored.

Chapter 2

EIGEN-VALUE BASED SHAPE DESCRIPTORS

2.1 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 [22]. Eigen-value based shape descriptors [23, 24], 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 [25, 26]. When research studies were conducted to study the impact of these subway cars on the geologic features [27, 28], 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.

2.2 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 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 [29] 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.[30] 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 [23, 24] 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.

2.3 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 [23, 24]. The λ_i's

are the eigen-values of the helmholtz differential equation (2.1) with Dirichlet boundary condition $u = 0$ on the boundary of the domain $\delta\Omega$.

$$\Delta u + \lambda u = 0 \tag{2.1}$$

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

$$0 < \lambda_1 \leq \lambda_2 \leq \lambda_3 \leq \cdots \leq \lambda_k \leq \cdots \to \infty. \tag{2.2}$$

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 ($n = 17$).

$$F(\Omega) = \left\{ \frac{\lambda_1}{\lambda_2}, \frac{\lambda_1}{\lambda_3}, \frac{\lambda_1}{\lambda_4}, \ldots, \frac{\lambda_1}{\lambda_n} \right\}. \tag{2.3}$$

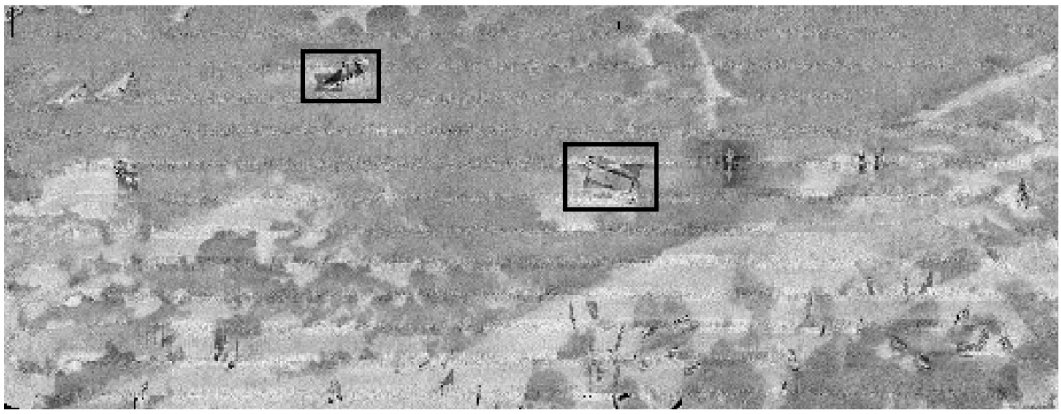
The angle Θ between two shape descriptor vectors $F(\Omega_1)$ and $F(\Omega_2)$ can be used as relative a measure of dissimilarity between the two shapes Ω_1 and Ω_2,

$$\Theta\left(\Omega_1, \Omega_2\right) = \cos^{-1}\left(\frac{\langle F(\Omega_1), F(\Omega_2) \rangle}{\|F(\Omega_1)\| \|F(\Omega_2)\|} \right). \tag{2.4}$$

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.

2.4 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 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.



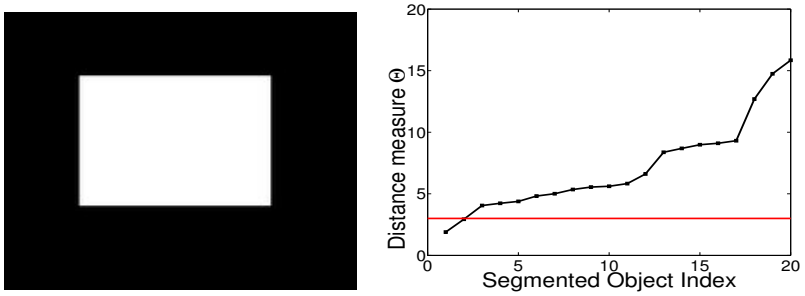
(a) Backscatter image of sea-floor from Gavia AUV

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

Figure 2.1

This problem of recognizing artificial objects from natural scenes was encountered while marine geologists were studying the Redbird artificial reef site [27, 28]. 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 2.1a shows the shape profiles of some sunken objects resting on the seabed. Two subway cars are marked using black rectangles. If we look for rectangular objects like the rectangle in Figure 2.2a, it is likely that eigen-value descriptors will distinguish the profiles of subway-cars, as there are no other objects with such rectangular shape profile in the seabed image (Figure 2.1a).



(a) (b)

Figure 2.2: (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

In order to verify the ability of eigen-value descriptors to pick out the subway-cars in the Figure 2.1a, 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 2.1b shows the different "blobs" obtained after thresholding. The blobs were then compared to the rectangular shape in Figure 2.2a using the angle Θ metric that was defined in (2.4). The Θ value of a blob is a direct indicator on how close in appearance it is to the reference rectangle in Figure 2.2a. 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 2.2b.

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 2.3a. Similarly the blob with the second lowest Θ value is shown in Figure 2.3b($\Theta = 2.93°$). In Figures 2.3a and 2.3b, 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 (2.4). 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 2.3c. 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.

2.5 Discussion

In Section 2.4, 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 2.2a. 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 2.1a). 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. 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

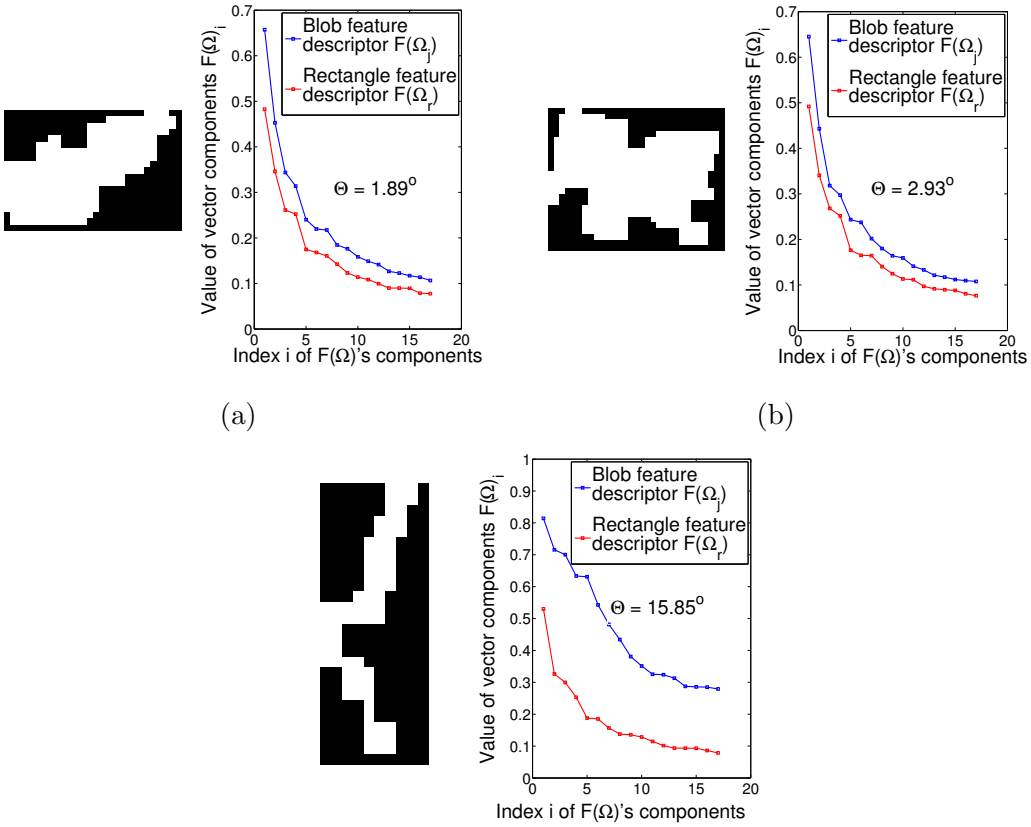


Figure 2.3: Parts (a) and (b) correspond to the blobs that are closest to the reference rectangle in Figure 2.2a. The shape of the blob is shown on left and the plot of shape descriptor $F(\Omega_j)$ of blob j (red line) along with the plot of the reference rectangular profile (blue line) is shown on 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).

descriptors. High levels of noise and errors in segmentation are other significant factors that can lower the performance of eigen-value shape descriptors.

2.6 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. Natural objects, like marine organisms, are typically associated with complex shape profiles. Furthermore texture is often a key discriminant required to identify marine organisms. The inability to use textural information, among other factors, limits eigen-value shape descriptors to few specialized object recognition applications only. Chapter 3 offers a multi-layered object recognition solution that attempts to overcome some drawbacks seen in eigen-value shape descriptor approach.

Chapter 3

MULTI-LAYERED SCALLOP RECOGNITION FRAMEWORK

3.1 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 [31, 32, 33, 34, 35]. 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 [36].

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 [37, 14, 20]. 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 dissertation 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 [37, 14, 20]. 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 [38] is discussed.

3.2 Background

3.2.1 Underwater Animal Recognition

In natural settings, living organisms often tend to blend into their environments to evade detection via camouflage. Webster's thesis work [39] provides a detailed exposition on the visual camouflage mechanisms adopted by animals to blend into their background. Under such circumstances of camouflage, there are very limited visual cues that can be used to identify animals. Even in the presence of visual cues, the task of

identifying animals from natural scenes is shown to be a cognitively challenging and complex task [7].

Previous efforts to detect animals like plankton [16, 17], clam [40] and a range of other benthic megafauna [21] exist. Most of these methods here are specialized to a specific species, or only tested in controlled environments. In some cases, the methods require specialized apparatus (like in the plankton recognition studies [16, 17]). A series of automated tools like specialized color correction, segmentation and classification modules along with some level of manual expert support, can be combined identification of several marine organisms like sea anemones and sponges from natural image datasets [21].

The existing techniques for marine animal recognition can be broadly divided into methods devised for identifying mobile organisms and methods for sedentary organisms. The former category is useful in dealing with a wide range of sea organisms like the varied species of fish that swim through water. The latter category is less studied. It includes identifying sedentary marine animals like scallops, corrals and sponges. Both categories present their own set of challenges. In the rest of this section we visit the techniques relevant to moving animals and show how they are different from the methods employed for sedentary animals. An overview of the existing literature on recognizing sedentary animals follows, with special emphasis on methods developed for identifying scallops.

3.2.1.1 Methods for Recognition of Moving Underwater Organisms

Recognizing and counting mobile marine life like fish [41, 42, 43] and studies in aquaculture [44] have been attempted. The recurring theme in these efforts involves the use of stationary cameras to detect the presence of moving species, provided that the background can be described by a prior model. This technique of assuming a known background, and using changes in the background as an evidence for the presence of a moving object entering the field of view of a sensor, is called background subtraction. In the marine species identification case, any changes to the background are assumed

to be caused by a moving marine organism. The pixels in the image that deviate from the background model can be labeled as the pixels belonging to the organism.

Once a marine organism is detected through background subtraction, then other computer vision or machine learning techniques can be used to classify the organism into a specific species based on its visible characteristics. This classification task can be achieved through conventional machine learning approaches. For instance the salmon species classification algorithm developed by Williams et al. [43] uses active contours to model the shape of the fish before comparing these contours to known salmon species. However, if the pixels corresponding to the organism are contaminated by high levels of noise, a specialized technique that is robust to noise might be required.

Background subtraction requires a mathematical model that describes the distribution of background pixels. In an underwater setting, such a background model can only be obtained if the camera is stationary and is observing a static background, or in the case that the background model represents , the evolution of which over time can be captured through a mathematical model. Such well defined background models are not always available. An opportunity to employ the background subtraction-based techniques arises in underwater environments with stationary fixtures designed to study a specific underwater location. In instances where such stationary arrangement of cameras is not available, background subtraction is inapplicable due to the lack of a background model.

3.2.1.2 Methods for Recognition of Sedentary Underwater Organisms

Since sedentary marine animals like scallops do not typically move (unless chased by a predator), a mobile robotic platform is required to traverse subsea relief to image and recognize those marine animals. Extending background subtraction to work with mobile robotic platforms is challenging, since the motion of the platform causes changes in its background. Generating a model for the background to perform background subtraction in these cases is problematic. This makes the task of detecting sedentary organisms with moving sensors even more challenging than detecting moving organism

with stationary sensors. The lack of background model in these cases motivates the development of a foreground model. If a foreground model is available, the task of detecting an organism can be realized as a search for pixels satisfying the foreground model in the image.

Detecting an organism typically involves segmenting all pixels of the organism, in order for one to classify the organism into a known category. The motion-based segmentation of marine animals that involves subtracting a known model of background from a snapshot of the environment, followed by attributing the pixels with non-zero values to the foreground is inapplicable in cases where the background model does not exist. Furthermore, the task of segmentation can be challenging in noisy images with weak edges, since the boundary pixels of the foreground object cannot be easily distinguished from background pixels.

Thus, the lack of background model makes background subtraction problematic. This leads to the need for techniques that depend on foreground models, and use of other features to detect and segment organisms from the background. This task becomes even more complicated if the organism does not present significant visual cues that make it distinctive from the background, as in the case of creatures exhibiting camouflage. High levels of noise or unpredictable environmental variables could also significantly affect the effectiveness of any animal recognition mechanism.

3.2.1.3 Scallop Recognition Methods

There are several aspects that make scallop recognition challenging. Scallops, especially when viewed in low resolution, do not provide features that would clearly distinguish them from their natural environment. This presents a major challenge in designing an automated identification process based on visual data. To compound this problem, visual data collected from the species' natural habitat contain a significant amount of speckle noise. Some scallops are also partially or almost completely covered by sediment, obscuring the scallop shell features. A highly robust detection mechanism is required to overcome these impediments.

There is a range of previously developed methods specialized for scallop recognition [38, 45, 18, 19, 46, 37, 14, 20] that operate on different assumptions, either with regards to the environmental conditions or the quality of data. Existing approaches to automated scallop counting in artificial environments [18, 19] employ a detection mechanism based on intricate distinguishing features like fluted patterns in scallop shells and exposed shell rim of scallops, respectively. Imaging these intricate scallop shell features might be possible in artificial scallop beds with stationary cameras and minimal sensor noise, but this level of detail is difficult to obtain from low resolution images of scallops in their natural environment. A major factor that contributes to the poor image resolution is the fact that sometimes the image of a target is captured several meters away from it. Overcoming this problem by operating an underwater vehicle much closer to the ocean floor will adversely impact the image footprint (i.e. area covered by an image) and increase the risk of damaging the vehicle.

Furthermore, existing work on scallop detection [38, 45] in their natural environment is limited to small datasets (often less than 100 images). A sliding window approach has been used [45] to focus the search for the presence of scallops. The large number of overlapping windows that need to be processed per image raises scalability concerns if this method were to operate on a large dataset containing millions of images. Additionally, the small number of natural images used as a test set raises questions about the generalizibility of this method and its ability to function under varied environmental conditions. The work by Dawkins [38] is more detailed in its treatment of the natural environmental conditions spanning the scallop habitat. The images used here are collected using a towed camera system that minimizes noise, a fact which greatly enhances the performance of the machine learning and computer vision algorithms. Despite the elaborate imaging setup designed to minimize noise, the results reported are derived only from a few tens to hundreds of images. It is not clear if those machine learning methods [38] can extend to noisy image data captured by AUVs. From these studies alone, it is not clear if such methods can be used effectively in cases of large datasets comprising several thousand seabed images. An interesting example

of machine-learning methods applied to the problem of scallop detection [46] utilizes the concept of Bottom-Up Visual Attention (BUVA). The approach is promising but it does not use any ground truth for validation.

There is more work [37, 14, 20] that offers a multi-layered object recognition framework validated on a natural image dataset for scallop recognition application. The main emphasis there (and in this dissertation) is to develop a technique that can work on low quality noisy sensor data collected using AUVs. The other objective is to build a scalable architecture that can operate on large image datasets in the order of thousands to millions of images and can be generalized for recognizing other marine organisms. A detailed comparison between the scallop recognition approaches in Dawkins et al. [38] and Kannappan et al.[20] is provided in Section **??**.

3.2.2 Motivation for a Generalized Automated Object Recognition Tool

Understanding the parameters that affect the habitat of underwater organisms is of interest to marine biologists and government officials charged with regulating a multi-million dollar fishing industry. Dedicated marine surveys are needed to obtain population assessments. One traditional scallop survey method, still in use today, is a dredge-based survey. Dredge-based surveys have been extensively used for scallop population density assessment [47]. The process involves dredging part of the ocean floor, and manually counting the animals of interest found in the collected material. In addition to being invasive and detrimental to the creatures habitat [48], these methods have accuracy limitations and can only generalize population numbers up to a certain extent. There is a need for non-invasive and accurate survey alternatives.

The availability of a range of robotic systems in form of towed camera and Autonomous Underwater Vehicle (auv) systems offer possibilities for such non-invasive alternatives. Optical imaging surveys using underwater robotic platforms provide higher data densities. The large volume of image data (in the order of thousands to millions of images) can be both a blessing and a curse. On one hand, it provides a detailed picture of the species habitat; on the other requires extensive manpower and time to process

the data. While improvements in robotic platform and image acquisition systems have enhanced our capabilities to observe and monitor the habitat of a species, we still lack the required arsenal of data processing tools. This need motivates the development of automated tools to analyze benthic imagery data containing scallops.

One of the earliest video based surveys of scallops [49] reports that it took from 4 to 10 hours of tedious manual analysis in order to review and process one hour of collected seabed imagery. The report suggests that an automated computer technique for processing of the benthic images would be a great leap forward; to this time, however, no such system is available. There is anecdotal evidence of in-house development efforts by the HabCam group [50] towards an automated system but as yet no such system has emerged to the community of researchers and managers. A recent manual count of our AUV-based imagery dataset indicated that it took an hour to process 2080 images, whereas expanding the analysis to include all benthic macro-organisms reduced the rate down to 600 images/hr [51]. Another manual counting effort [52] reports a processing time of 1 to 10 hours per person to process each image tow transect (the exact image number per tow was not reported). The same report indicates that the processing time was reduced to 12 hours per tow by subsampling 1% of the images.

Future benthic studies can be geared towards increasing data densities with the help of robotic optical surveys. It is clear that the large datasets, in the order of millions of images, generated by these surveys will impose a strain on researchers if the images are to be process manually. This strongly suggests the need for automated tools that can process underwater image datasets. Motivated by the need to reduce human effort, Schoening [21] has proposed a range of tools that can be generalized to organisms like sea-anemones. With an additional requirement of being able to work with low-resolution noisy underwater images, a generalized multi-layered framework that can be used to detect and count underwater organisms has been proposed [37, 14, 20]. This method has been evaluated on a scallop population assessment effort on a dataset containing over 8000 images, the details of which is the subject of this chapter.

3.3 Preliminaries

3.3.1 Visual Attention

Visual attention is a neuro-physiologically inspired machine learning method [53] 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 [53], 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. [54] 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

$$I(c, s) = |I(c) \ominus I(s)| \quad . \tag{3.1}$$

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

$$\bar{I} = \bigoplus_{c=2}^{4} \bigoplus_{s=c+3}^{c+4} w_{cs}\, \mathcal{N}(I(c, s)) \quad . \tag{3.2}$$

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

$$S = w_{\bar{I}}\, \mathcal{N}(\bar{I}) + w_{\bar{C}}\, \mathcal{N}(\bar{C}) + w_{\bar{O}}\, \mathcal{N}(\bar{O}) \quad . \tag{3.3}$$

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 [54, 55] 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 [56], the weights in (3.2) and (3.3) 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 [56]. 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

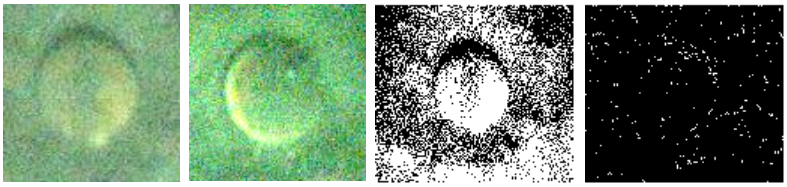
$$\begin{aligned} w'_j &= \frac{\sum_{i=1}^{N} N_{iT}^{-1} \sum_{k=1}^{N_{iT}} P_{ijT_k}}{\sum_{i=1}^{N} N_{iD}^{-1} \sum_{r=1}^{N_{iD}} P_{ijD_r}} \\ w_j &= \frac{w'_j}{\frac{1}{N_m} \sum_{j=1}^{N_m} w'_j}, \end{aligned} \tag{3.4}$$

where $j \in \{1, \ldots, N_m\}$ is the index set of the different maps to be combined. Equations (3.4) are used for the selection of weights w_{cs} in (3.2), and $w_{\bar{I}}$, $w_{\bar{O}}$, $w_{\bar{C}}$ in (3.3).

3.4 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

Figure 3.1: Seabed image with scallops shown in red circles



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

Figure 3.2: (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.

work with is shown in Figure 3.1 (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 thresholding and edge detection on the images (see Figure 3.2c and 3.2d) 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 AUV strobe light (see Figure 3.2a). 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 3.2b). A third pattern may be a yellowish tinge associated with the composition of the scallop image (see Figure 3.2b).

We have leveraged visual patterns [14] 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 [14] by adding a fourth, false-positives filtering layer [20].

3.5 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.

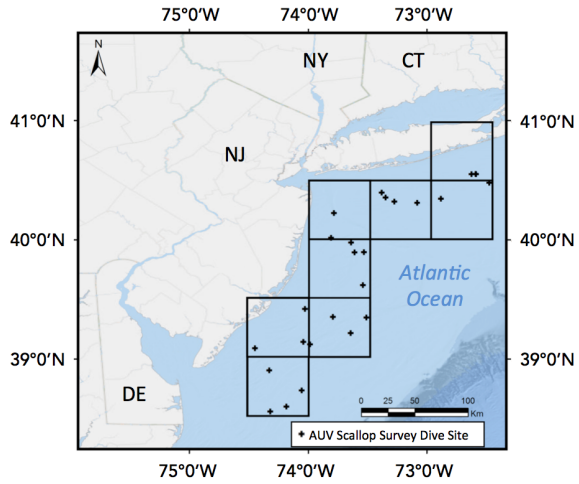


Figure 3.3: Map of the survey region from Shinnecock, New York to Cape May, New Jersey, divided into eight blocks or strata

3.5.1 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 3.3). The *f/v Christian and Alexa* served as the surface support platform from which a Gavia AUV (see Figure 3.4) was deployed and recovered. The 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 [38]).

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.

3.5.2 Sensors and Hardware

The University of Delaware AUV (Figure 3.4) 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

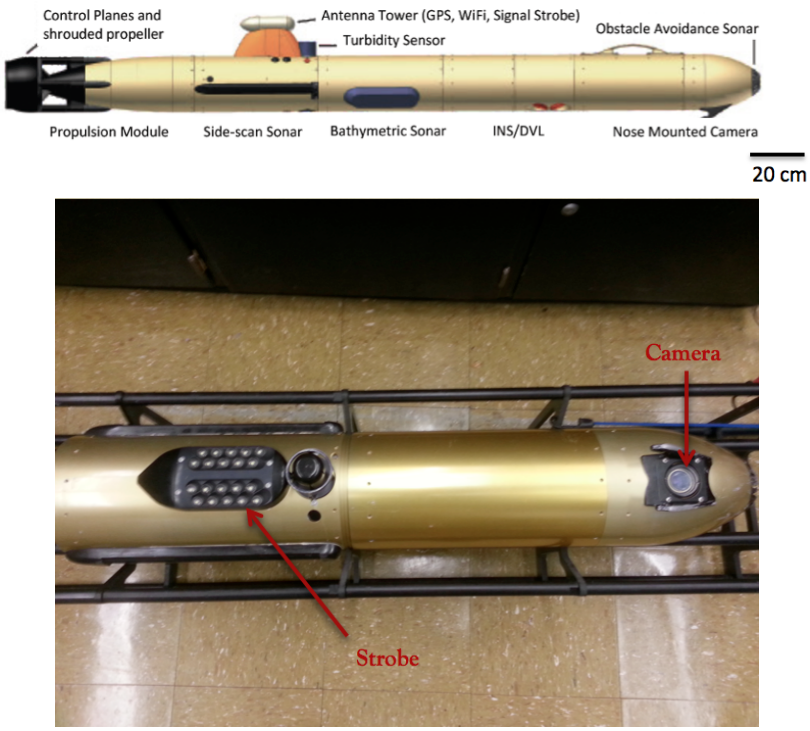


Figure 3.4: Schematics and image of the Gavia AUV

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.

3.5.3 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 3.1). 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 3.2), 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) [51].

3.6 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.

3.6.1 Layer I: Top-Down Visual Attention

3.6.1.1 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, BUVA based saliency computation. The saliency computation is performed on a collection of randomly selected 243 annotated images,

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 3.1: 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 3.2: Camera specifications

collectively containing 300 scallops. This collection constitutes the *learning set*. Figure 3.5 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*.

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 (3.4), 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 (3.4) and consequently used in (3.2), are shown in Table 3.3. For the saliency map computation, the weights resulting from the application of (3.4) on the conspicuity maps

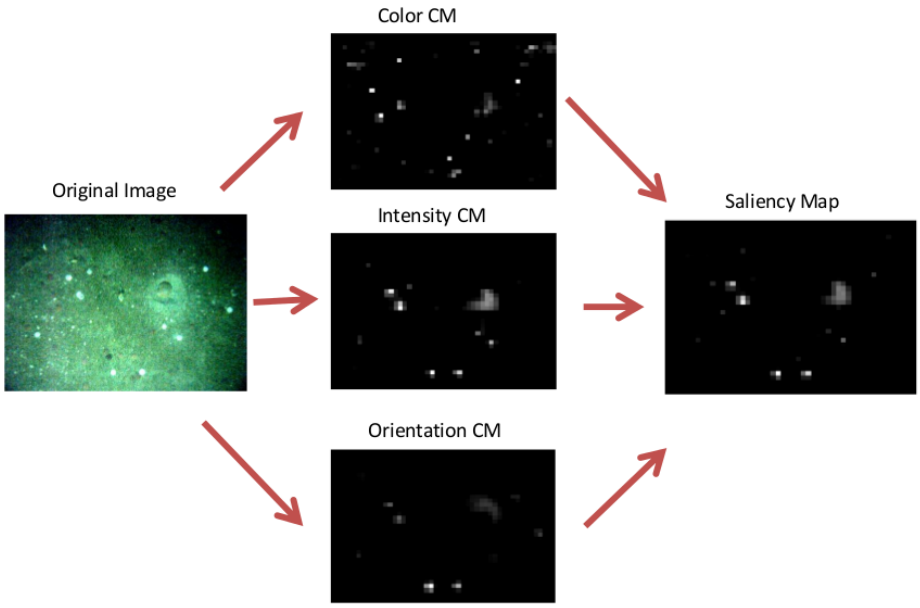


Figure 3.5: Illustration of computation flow for the construction of saliency maps

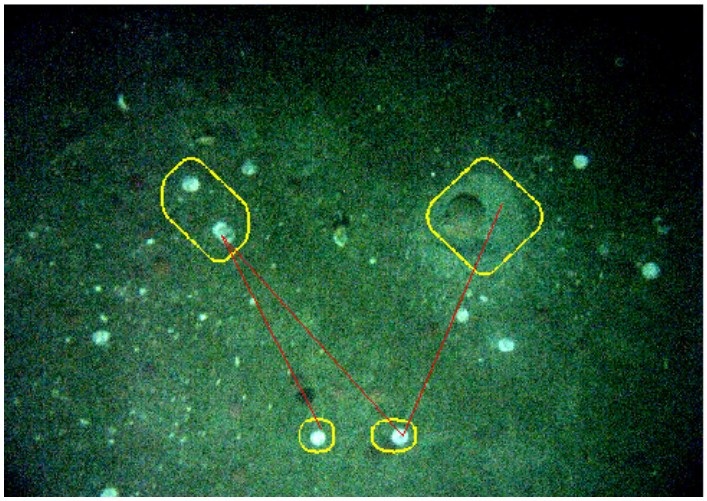


Figure 3.6: Illustration of fixations (marked by yellow boundaries): red lines indicate the order in which the fixations were detected with the lower-left fixation being the first.

Table 3.3: 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

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.

3.6.1.2 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 3.5. In this application, the saliency maps are computed via the formulae (3.3) and (3.2), using the weights listed in Table 3.3. Convergence time of the winner-takes-all neural network that finds fixations in the saliency map of each image in the datasets of Table 3.5, 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 3.5 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

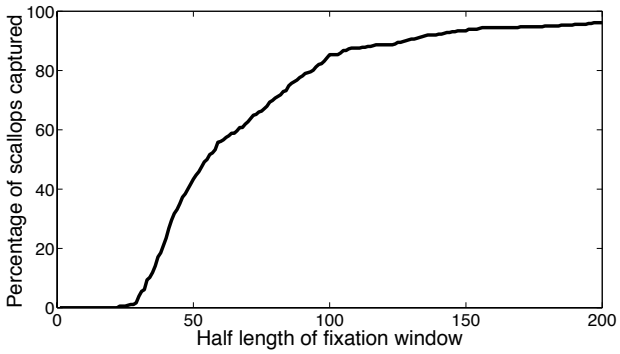
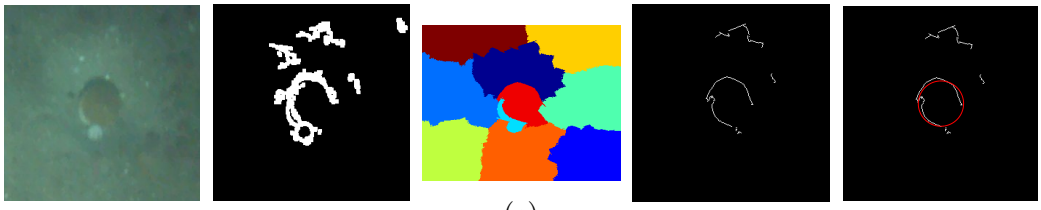


Figure 3.7: Percentage of scallops enclosed in the fixation window as a function of window half length (in pixels)



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

Figure 3.8: 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.

accommodated inside the window (Figure 3.7).

3.6.2 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 3.8. Figure 3.8a shows a fixation window. Edge-based segmentation on this window yields the edge segmented image of Figure 3.8b. At the same time, graph-cut segmentation process [57] is applied on the fixation window to decompose it into 10 separate regions as seen in Figure 3.8c. 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 3.8d. This is followed by fitting a circle to each of the contours in the filtered region boundaries (Figure 3.8d). Only circles with dimensions close to that of a scallop (diameter $20-70$ pixels) are retained (Figure 3.8e), 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 [58, 59].

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:

$$F_2(A, B, C, D) = \frac{\sum_{i=1}^{n} [A(x_i^2 + y_i^2) + Bx_i + Cy_i + D]^2}{n^{-1} \sum_{i=1}^{n} [4A^2(x_i^2 + y_i^2) + 4ABx_i + 4ACy_i + B^2 + C^2]} \ . \tag{3.5}$$

It is shown [58] that minimizing (3.5) 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:

$$a = -\frac{B}{2A} \ , \qquad b = -\frac{C}{2A} \ , \qquad R = \sqrt{\frac{B^2 + C^2 - 4AD}{4A^2}} \ . \tag{3.6}$$

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 (3.6), 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 [51] have a measurement error of 5–10%.

3.6.3 Layer III: Classification

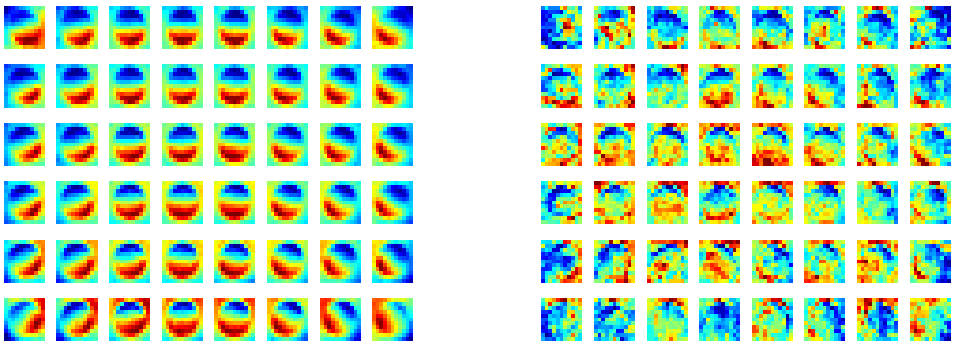
The binary classification problem solved in this layer consists of identifying specific features in the images which mark the presence of scallops. These images are

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.

3.6.3.1 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 3.9a) and standard deviation (Figure 3.9b) of the 11×11 scallop profiles of all scallops per grid square over the whole dataset of 3 706 images was recorded. The 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] 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.

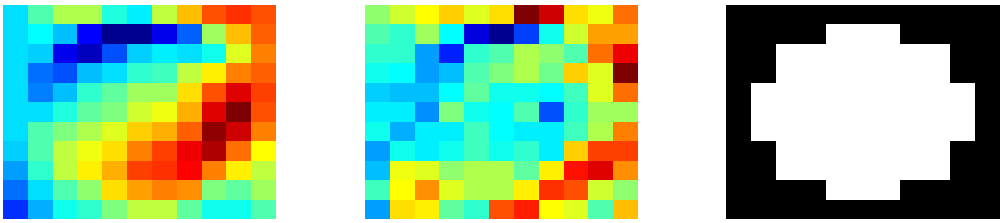


(a) (b)

Figure 3.9: 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.

3.6.3.2 Scallop Profile Learning

The statistics of the dataset of 3 706 images used to produce Figure 3.9 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 3.10), 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 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 (3.7)).



(a) (b) (c)

Figure 3.10: 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.

3.6.3.3 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 $X^o = (X_1^o, X_2^o, \ldots, X_{46}^o)$ denote the feature vector computed for the segmented object, and $X^s = (X_1^s, \ldots, X_{46}^s)$ 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

$$X_p^{\bar{o}} = \min_k X_k^s + \left(\frac{\max_k X_k^s - \min_k X_k^s}{\max_k X_k^o - \min_k X_k^o} \right) \left[X_p^o - \min_k X_k^o \right] \text{ for } p = 1, \ldots, 46 \ ,$$

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

$$D_t = \sqrt{\sum_{k=1}^{n} \frac{\|X_k^{\bar{o}} - X_k^s\|^2}{\sigma_k}} \ . \tag{3.7}$$

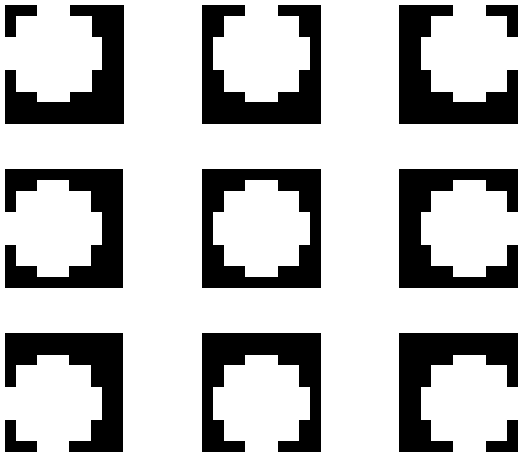
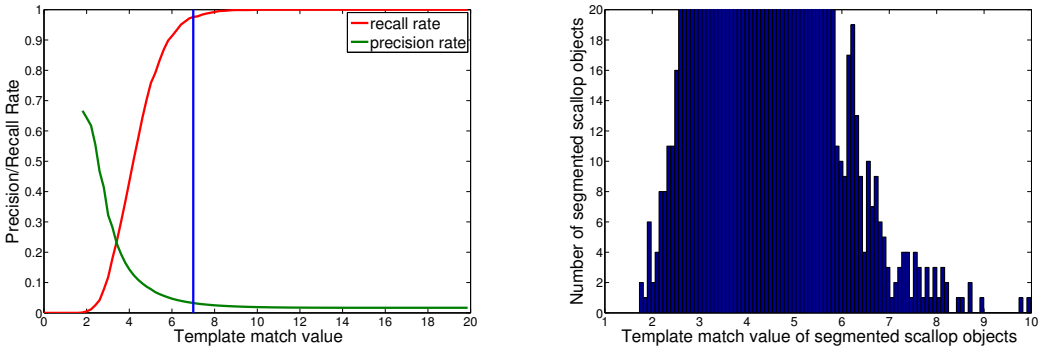


Figure 3.11: Nine different masks slightly offset from the center used to make the classification layer robust to errors in segmentation

Small variations in segmentation can produce notable deviations in the computed distance metric (3.7). To alleviate this effect, the mask of Figure 3.10c 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 3.11). 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 3.12a–3.12b, the threshold value was chosen at $D_{\mathsf{thresh}} = 7$ to give a recall[3] rate of 97%. Evident in Figure 3.12a is the natural trade-off between increasing recall rates and keeping the number of false positives low.

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

3.6.4 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 Weighted Correlation Template Matching (WCTM) technique and a Histogram of Gradients (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.

3.6.4.1 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 3.9), 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:

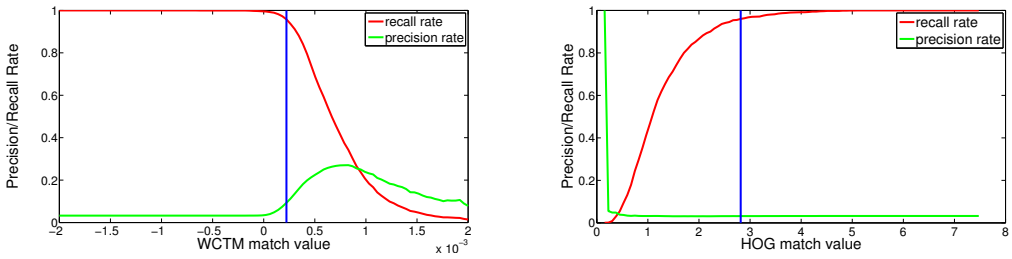
$$X_p^{t'} = \min_k X_k^u + \left(\frac{\max_k X_k^u - \min_k X_k^u}{\max_k X_k^t - \min_k X_k^t} \right) \left[X_p^t - \min_k X_k^t \right] \ ,$$

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:

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



(a) (b)

Figure 3.13: 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.

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

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

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 3.13a) 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%.

3.6.4.2 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 [60] 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 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 [60], 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 [61] 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 [61]). Then the EMD metric D_{emd} that quantifies dissimilarity between two histograms A and B would be expressed as

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

A precision-recall curve (shown in Figure 3.13b) 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

	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 3.4: Comparison of tested false positive filter layer methods

positives actually get eliminated (less than 3%).

3.7 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 3.4 for both datasets. Rows 1 and 2 in Table 3.4 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 3.4).

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 [60] might need further customization

Table 3.5: 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%

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 3.5. 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 3.1) 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 3.9a). 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 [51].

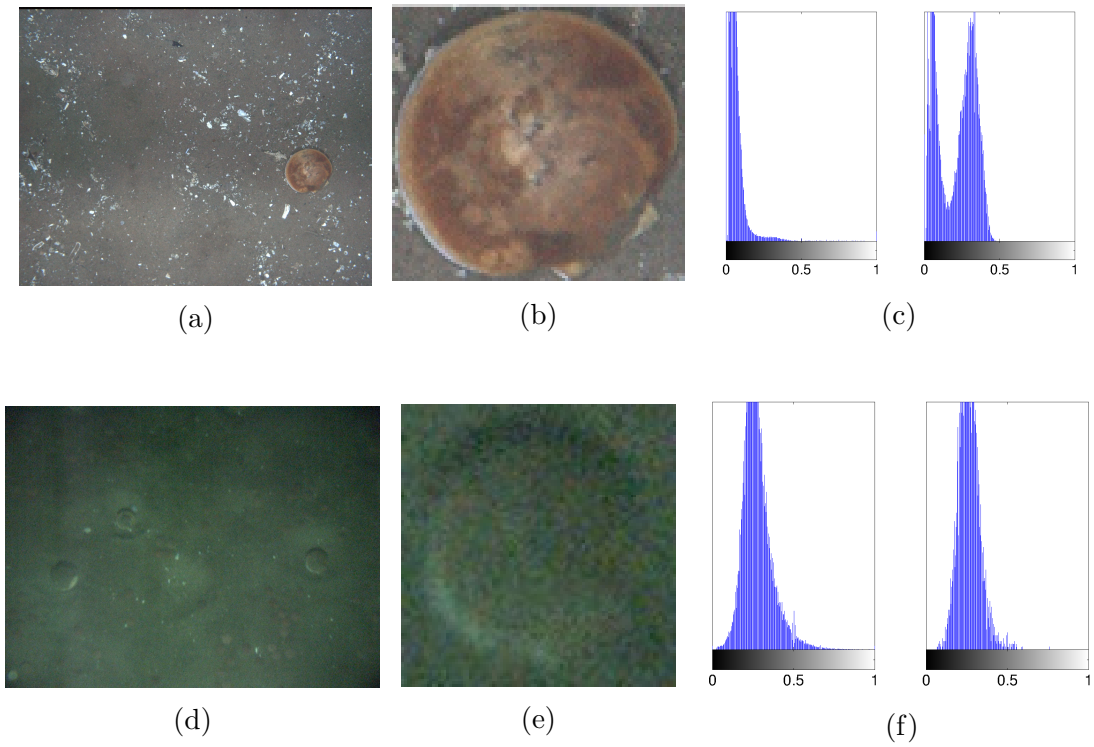
Table 3.5 shows the results of the 3-layer pipeline along with the improvements in

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

3.8 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 [62, 38], reported higher detection rates, but the quality of the images used was visibly better. Specifically, the datasets on which the alternative algorithms [38] operated on, exhibit much more uniform lighting conditions, higher resolution, brightness, contrast, and color variance between scallops and background (see Figure 3.14). Evidence of this can be seen in Figure 3.14: the color variation between scallops and background data is reflected in the saturation histogram of Figure 3.14. While the histograms of scallop regions in the datasets of Table 3.5 is often identical to the global histogram of the image, the histograms of the Woods Hole data used by the alternative algorithms [38] present a bimodal saturation histogram (Figure 3.14c), from which foreground and background are easily separable.

Compared to another alternative approach that uses a series of bounding boxes to cover the entire image [45], the one reported here employs only ten windows per image, scanning the images at a much faster rate. Additionally, the detection rates there [45] were based on a dataset of just 20 images; statistically significant differences



(a) (b) (c)

(d) (e) (f)

Figure 3.14: Representative samples of different imagery data on which scallop detection algorithms may be called to operate on. Figures 3.14a and 3.14d, show an image containing a single scallop from the dataset used by Dawkins et al.[38] (used with permission from the authors) and the datasets used in this paper respectively. A magnified view of a scallop cropped from Figure 3.14a and 3.14d can be seen in Figures 3.14b and 3.14e respectively. Figure 3.14c gives the saturation histogram of background or the complete image in Figure 3.14a to left and saturation histogram of Figure 3.14b to the right. Similarly, Figure 3.14f gives the saturation histogram of Figure 3.14d to the left and saturation histogram of Figure 3.14e to the right. The bimodal nature of the scallop histogram in Figure 3.14c derived from the dataset used in Dawkins et al.[38], 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 3.14f) to aid the identification of scallops.

in performance rates between that approach and the one reported here would need much larger image samples.

3.9 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.

3.10 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 [38] as a part of image preprocessing will improve results.

Building robust object classification techniques capable of handling noisy data, is one of the primary directions where improvement is necessary. It is possible that a single noisy image of a target object might lack the information needed to accurately recognize it. With this in mind, a multi-view object recognition approach that combines information from multiple images is proposed in Chapter 4.

Chapter 4

DISTANCE BASED GLOBAL DESCRIPTORS FOR MULTI-VIEW OBJECT RECOGNITION

4.1 Introduction

The ability to recognize objects on distorted images is critical for robotic systems that operate in natural environments where measurement is contaminated by noise. In the presence of noise, the information contained in an image might not be sufficient to unambiguously classify the *object-of-interest*. One possible way to resolve this ambiguity regarding the object's identity is by combining information from multiple views. The approach described here therefore, combines information from multiple views of a target, and employs global image descriptors to solve a classification problem.

Object recognition can be cast as a generalized machine learning problem. Traditional machine learning approaches [63] use a learning set with annotated instances to learn the characteristics of an object. A classifier can be used to label a given instance as a target object, if sufficient evidence is available from the learning set to support this hypothesis. Machine learning techniques, like convolution neural networks [64], use several thousand labeled images to solve the object recognition problem in a purely *data-driven* fashion. In the absence of large labeled datasets for specialized applications like underwater animal recognition, such data-driven approaches require a labor intensive annotation process.

An alternative to a data-driven approach of this type is a *feature-based* object recognition approach [65]. In feature-based object recognition, prior knowledge about the object to be recognized can be leveraged to build specialized feature descriptors that capture the appearance traits of an object. The models used in such cases can

range from simple geometric shapes to complex free-form models [66, 67]. The feature descriptors thus trained encode the object appearance through a low-dimensional mathematical model. This model can be used to check for the presence of an object in an image.

Feature-based approaches can be further subdivided into two classes; local, and global. Local feature-based approaches like, Scale-invariant Feature Transform (SIFT) [68] and Speeded-up Robust Features (SURF) [69], encode a configuration of localized features specific to an object to build a descriptor. In the case of SIFT and SURF, the feature descriptors obtained are invariant to in-plane rotation and scaling. These properties make SIFT and SURF very robust and suitable for a multitude of object recognition applications. Since local descriptors primarily depend on artifacts like corners in images, they are sensitive to noise which can degrade and distort such artifacts. Therefore, local descriptors can be severely affected by noise in images. On the other hand, global feature descriptors like *gist* [70] are more resilient to noise as they encode general characteristics of an image which are less impacted by the presence of noise. Global feature representation also has the additional benefit of not requiring segmentation. Since segmentation is a hard problem in itself, especially in cases of noisy images where it is difficult to delineate foreground and background regions, its beneficial to use a method that does not require segmentation. However, the global feature descriptors often only provide a weak description of an object which might not be sufficient to unambiguously detect the occurrence of a specific object.

One way to strengthen a weak feature descriptor is to combine information about a single object instance, derived from multiple information sources. For instance, this can be done by merging information from multiple views of an object. Such an approach is more effective compared to a case in which a single view of an object does not contain sufficient information to unambiguously expose an object. Based on these multiple views, a sequence of hypothesis tests can be combined to eventually classify the object. This process of combining multiple weak classifiers is a common theme in machine learning methods like boosting and bagging [63].

The idea of combining multiple sensor measurements is also encountered in the area of active sensing [71]. In most active sensing problems, for instance *next best view* [72, 73], the task is to determine successive sensor positions that increase the perceived information associated with a target object. In the approach described in this chapter, there are differences in the way the final objective of object classification is accomplished: a target is observed from several pre-determined sensor positions in a way is conceptually similar to active sensing. However in active sensing, the next sensor position is determined at each iteration based on the current information available.

Combining information from multiple views can also be related to computer vision techniques that operate on the principle of multi-resolution processing or image-pyramid-based methods. Visual attention based object recognition frameworks (like VOCUS [74]), and multi-resolution pedestrian detection [75], utilize features from multiple scales to build a robust feature descriptor for objects. These multi-resolution techniques essentially resize an image to compute features at different scales. Thus such techniques can only represent the information available from a single parent image in different forms. In cases where objects exhibit different appearance traits when viewed from different heights, multiple images of an object are needed to capture new information disclosed at different scales.

This chapter, describes an object recognition technique that combines information from multiple images of an object gathered from different heights to perform binary classification. The contribution here is in the design of a novel histogram-based global feature descriptor, along with a hypothesis testing mechanism that combines information from multiple-views of a target object. The approach developed here lies at the intersection of global feature descriptors, active sensing, and multi-resolution image processing to offer a novel object recognition technique that is intended to operate on noisy natural images. The use of global feature descriptors obviates segmentation of noisy images. At the same time, a strong feature descriptor is constructed by combining information from several weak global feature descriptors, each learnt from a different height scale. This approach also offers a semi-automated annotation framework to

minimize the human effort involved in the annotation task.

4.2 Preliminaries

4.2.1 Grabcut-in-one-cut Algorithm

A *graphcut* algorithm treats the segmentation problem as a graph partition problem [76]. Consider a case where each pixel p of an image I is treated as a vertex i in the graph $G(\mathcal{V}, \mathcal{N})$. Two neighboring pixels p and q, represented by vertices i and j respectively, are connected by an edge e_{ij} in graph G. The task of partitioning this graph G into foreground and background is cast as an optimization problem where some form of "energy" E is to be minimized. Here, the energy E is formulated as

$$E(L) = \sum_{p \in I} D_p(L_p) + \sum_{(p,q) \in \mathcal{N}} V_{p,q}(L_p, L_q) \tag{4.1}$$

where L_p corresponds to a labeling for pixel p, D is a data penalty function that assigns a likelihood for pixel p to belong to either foreground or background label class, and $V_{p,q}$ is an interaction potential function that encourages spatial coherence by penalizing discontinuity in label values of pixels. A binary partition of this graph is obtained by choosing a label for each pixel p such that the energy E of the image is minimized. This graph-based segmentation approach, also referred to as the max-cut min-flow algorithm was adapted to solve binary image segmentation problems. To this end, a variant called Grabcut-in-one-cut [77] uses pre-specified foreground and background seeds to perform a binary partition of an image. According to the Grabcut-in-one-cut algorithm, the energy function in (4.1) is reformed as

$$E_{seeds}(S) = -\beta \|\theta^S - \theta^{\bar{S}}\|_{L_1} + \lambda |\partial S| \ . \tag{4.2}$$

Here, θ^S and $\theta^{\bar{S}}$ are the histograms of the foreground and background pixels in the image. The first term of (4.2), $\|\theta^S - \theta^{\bar{S}}\|_{L_1}$ penalizes the pixels with intensities that overlap with both foreground and background distributions. The second term of (4.2), $|\partial S|$ is a term analogous to V in (4.1), that tries to enforce similar labeling of neighboring pixels and therefore penalizes a change in labeling across neighbors. The weights

λ and β allow to vary the contribution of the first and second term to the value of the energy function in (4.2). Finally L_1 in (4.2) refers to the L_1-norm.

The Grabcut-in-one-cut algorithm generates a binary partition of an image into background and foreground by extrapolating the input foreground and background seed pixels. If there is an automated means to find the location of foreground objects, the seeds required by this algorithm can be automatically generated, hence making this guided segmentation approach fully autonomous.

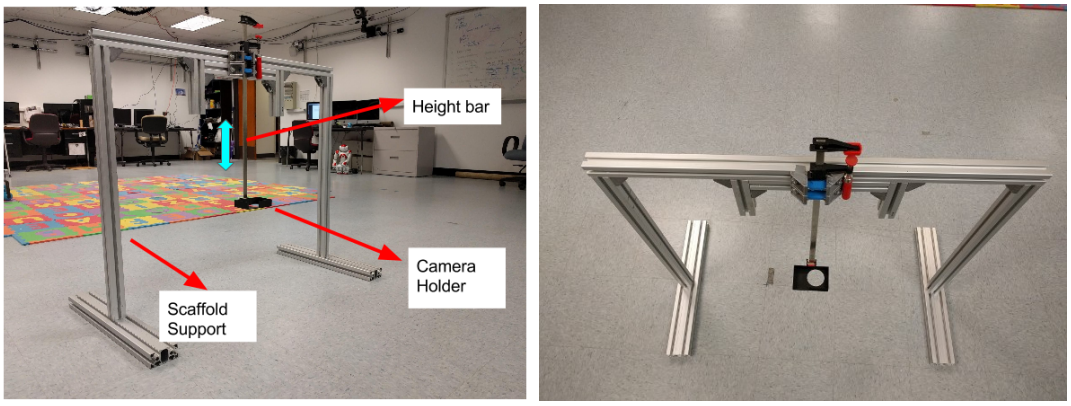
4.2.2 Data Collection Apparatus

4.2.2.1 Imaging Rig

The imaging rig shown in Figure 4.1 allows a camera to be held at different heights from the ground for imaging experiments. The height bar slides up or down and can be locked at a specific configuration through specially designed clamps on the scaffold support of the imaging rig. By varying the position of the height bar, the height of the camera holder that is attached to the lower end of the height bar can be varied. This allows the camera held on to the camera holder to capture images of targets placed on the ground from different controlled heights. The camera holder was designed to hold a GoPro Hero 4 camera [78].

4.2.2.2 Imaging Environment

The environment where object recognition experiments are performed could affect the performance of the algorithm used. In case of an environment with limited to no control over the imaging parameters, the level of noise introduced into the sensor measurements difficult to gauge. This complicates the task of designing noise filters to take any corrective measures. It might be tempting to perform object recognition experiments in controlled laboratory conditions, thereby allowing the possibility to tune the environmental parameters to enhance the performance of an algorithm. However, if an algorithm is intended to operate on natural images, there are often going to be



(a) (b)

Figure 4.1: The imaging rig allows the height bar to slide up or down, thereby letting the height of the camera from the ground to be varied. The different components of the imaging rig are labeled in Figure 4.1a. The camera holder attached to the lower end of the height bar is designed to carry a GoPro Hero 4 camera.

unpredictable variations in environmental factors that could adversely affect the performance of the algorithm. To accurately validate an algorithm that is intended to operate on natural environments, it is imperative to choose a test environment where the conditions are close to the expected natural conditions. Following this line of reasoning, since this object recognition algorithm is intended to operate on noisy natural environments, an underwater environment with uncontrolled lighting was chosen as the test environment. A circular tank in the Robot Discovery Lab (RDL) at University of Delaware filled with 4 feet of fresh water was thus utilized to collect experimental data. The imaging rig described in Section 4.2.2.1 was submerged inside this test tank during the data collection process.

4.2.2.3 Object Specimens

The focus of this work is to recognize a class of objects that can exhibit some level of intra-class variance that is characteristic to naturally occurring objects. One such example is underwater marine organisms, where each species can have several identifiers that distinguish them from a different species. However in most cases, the members of a single species also exhibit some level of variation in their appearance. To accommodate

(a) Potatoes (b) Tomatillos

Figure 4.2: An illustration of the inter-class and intra-class variance exhibited by naturally occurring objects like potatoes (a) and tomatillos (b)

.

such cases, the specimens used to validate this multi-view object recognition algorithm were required to have (i) some characteristics that are unique to the class they belong to (ii) small variations with regards to appearance within the same class of objects, and (iii) easy accessibility for experimentation purposes. All these points were satisfied by natural produce like potatoes, oranges, tomatillos and strawberries. Natural produce exhibit significant inter-class variance, sufficient intra-class variance and are readily available in stores making them ideal candidates for object specimens. Figure 4.2 shows an assortment of potatoes and tomatillos to visually reinforce the inter-class and intra-class variance exhibited by naturally occurring objects. The underwater data on which this multi-view algorithm is validated is composed of a set of 11 oranges and 11 strawberries.

4.3 Definitions

4.3.1 Histogram Signature

As a part of this object recognition technique we define a histogram signature that improves on the generic histogram, specifically designed to deal with possible noise in the pixel values. The rest of this section defines this histogram signature.

A generic histogram $H_f : b_i \mapsto n_i$ of some colorspace f, captures the distribution of values of pixels in colorspace f, where n_i represents the count of bin $b_i \in \mathcal{B}$, and $\mathcal{B}$ is the set of bins used to partition colorspace f. Any pixels affected by noise in image I

could corrupt the histogram H_f. If we assume that $e\%$ of the pixel values of an image constitutes noise, then the histogram H_f can be refined by rejecting bins containing $e\%$ of pixel values that are least likely to occur according to histogram H_f. Before computing the refined version of this histogram, the histogram H_f is first normalized in to the form $\bar{H}_f : b_i \mapsto \bar{n}_i$ where

$$\bar{H}_f(b_i) = \frac{n_i}{\sum_j n_j} = \bar{n}_i \ . \tag{4.3}$$

Now the refined *histogram signature* $\mathcal{H}_f : b_i \mapsto n'_i$ is computed, such that

$$n'_i = \begin{cases} \bar{n}_i & b_i \in \bar{\mathcal{B}}_e \\ 0 & b_i \in \mathcal{B}_e \end{cases}, \tag{4.4}$$

where $\mathcal{B}_e$ and $\bar{\mathcal{B}}_e$ constitute a binary partition of $\mathcal{B}$, i.e. $\mathcal{B}_e \cap \bar{\mathcal{B}}_e = \phi$ and $\mathcal{B}_e \cup \bar{\mathcal{B}}_e = \mathcal{B}$, chosen in a way that $\bar{\mathcal{B}}_e$ is the smallest cardinality subset of $\mathcal{B}$ that satisfies the condition $\sum_{b_i \in \bar{\mathcal{B}}_e} \bar{H}_f(b_i) > 1 - e$. The definition of set $\bar{\mathcal{B}}_e \subseteq \mathcal{B}$ can be restated as

$$\bar{\mathcal{B}}_e = \underset{|\bar{\mathcal{B}}_e|}{\operatorname{argmin}} \sum_{b_i \in \bar{\mathcal{B}}_e} \bar{H}_f(b_i) > 1 - e \tag{4.5}$$

The $|\mathcal{B}|$-element histogram signature $\mathcal{H}_f$ now encodes the histogram information while attempting to filter some of the noise present in sensor measurements.

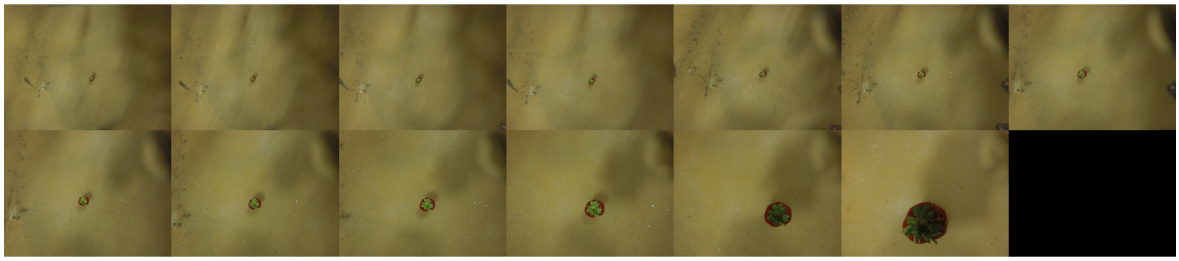
4.4 Methodology

The distance-based global descriptor algorithm presented in this chapter is essentially a machine learning technique. As it is common with such techniques, it consists of two parts: a learning part and a validation part. In the learning phase, features and other attributes of an object class are captured from labeled instances of the object available in the learning set. In the validation phase, the learned descriptor for an object class is validated against pre-labeled images to evaluate the performance of an algorithm. The data collection and annotation phases that provide data used for learning and validation parts of this algorithm are discussed next.

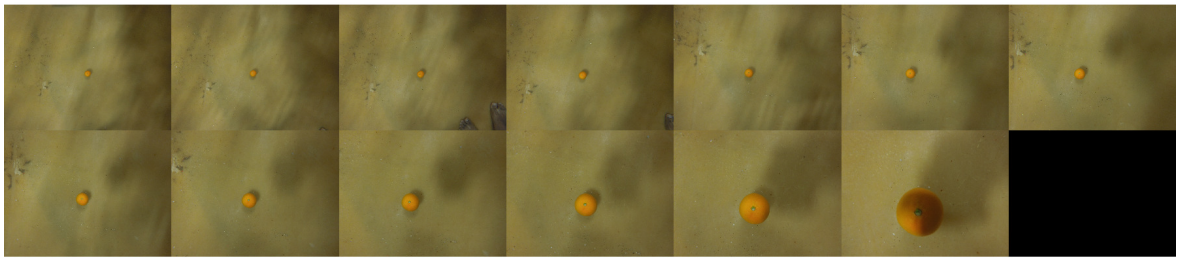
4.4.1 Data Collection

The data collection process here involves capturing 13 images for each object specimen. For the validation of this multi-view object recognition algorithm, data was gathered from 22 specimens (11 oranges and 11 strawberries) bringing the total images collected to 286 (22 specimens $\times$ 13 images per specimen). Each specimen was first placed on the floor below the camera-holder on the imaging rig (see Figure 4.1). The height bar was moved up till the camera holder was 32 inches away from the ground. A GoPro Hero 4 camera, attached to the holder, was then triggered to capture an image of the specimen. This first image captured the visual appearance of the specimen 32 inches away from the ground. After the first image was captured and without disturbing the object, the next image was captured after lowering the height bar such that the camera was now 30 inches away from the ground. This process of lowering the camera by 2 inches between subsequent images is continued till the camera gets to a height of 8 inches away from the ground. The series of images captured from a range of heights from 32 inches to 8 inches, every 2 inches results in 13 images per specimen. These 13 images capture the appearance of a specimen at different heights. An illustration of the 13 images captured for a strawberry and orange specimen is shown in Figure 4.3. An important point to note here is that the imaging rig is handled manually between each image capture which sometimes results in displacement of the imaging rig or the specimen. Additional disturbances in the underwater testing environment where the specimen was placed, added another source of inconsistency that sometimes lead to the specimen or the imaging rig getting displaced during the data collection process. These uncontrolled factors in the data collection process added some level of variance to the gathered dataset. Since this algorithm is intended to operate on uncontrolled natural environments, such variance in the collected data was welcomed to some degree, given that it could test the algorithm's robustness.

Due to the wide-angle fish-eye lens of the GoPro camera used, the ratio of the pixels of the object specimen to the background could be very small when the camera is far away from the target. In technical terms, when the images of a target are captured



(a) Strawberry specimen seen from different heights



(b) Orange specimen seen from different heights

Figure 4.3: A set of 13 images gathered for a specimen (strawberry in (a) and orange in (b)) starting from a height of 32 inches (top left) up to a height of 8 inches (bottom right) away from the target. Each subsequent image was captured 2 inches closer to the specimen.

by a camera that is far away from the specimen, the number of pixels that correspond to the specimen in the image could be statistically insignificant. This reported object recognition procedure depends on histograms of images (more details of how the histogram is relevant is explained in Section 4.4). For the histogram to accurately capture the properties of a specimen, the images of the specimen need to have a relatively high ratio of object pixels over background pixels. To improve this ratio of object pixels in the image, the images were cropped to discard a portion of background. The strawberry and orange specimens shown in Figure 4.3a and Figure 4.3b are shown again in Figure 4.4 and Figure 4.5 respectively, after cropping of portions of background.

4.4.2 Annotation Process

The task of manual image annotation is a costly and labor intensive process. Developing an object recognition algorithm is often motivated by a desire to avoid

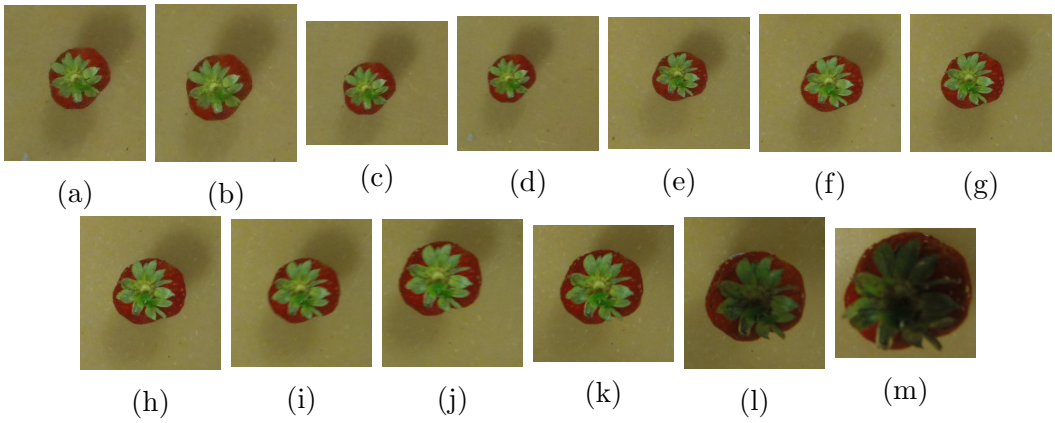


Figure 4.4: The strawberry specimen in Figure 4.3a shown after cropping a portion of the background

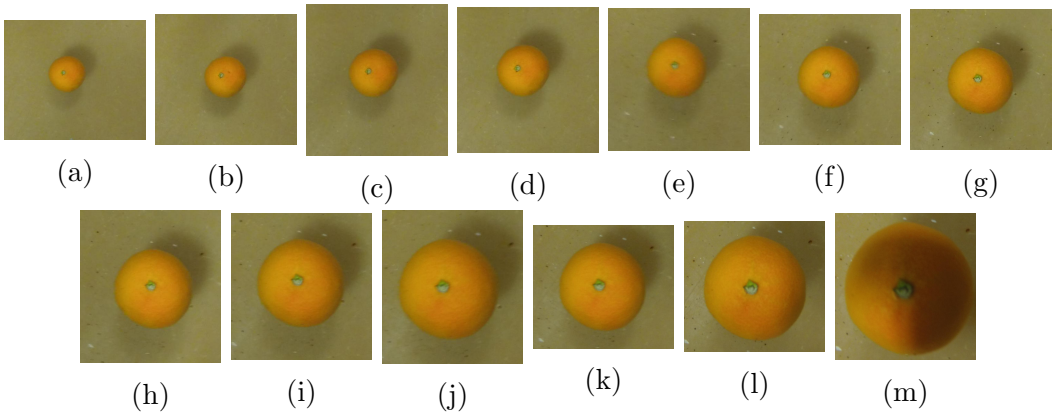


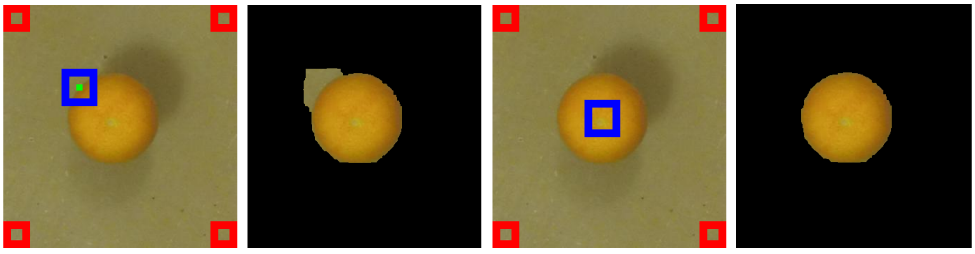
Figure 4.5: The orange specimen in Figure 4.3b shown after cropping a portion of the background

manual inspection in order to identify objects from images. However, to build a machine learning framework for object recognition, an initial set of images needs to be annotated manually to contribute to the learning set. Just like any other supervised machine learning algorithm, this procedure requires annotation of images into labeled foreground and background. Here, each learning sample consists of 13 images of a specimen captured from different heights. Increasing the number of images of each specimen quickly increases the annotation load. To mitigate this burden, a semi-automated annotation framework is developed.

In the first stage of the annotation process, a customized BUVA technique (described in Section 3.3.1) is used to pick the most *interesting* point in an image. This most *interesting* point is referred to as the first fixation. According to the visual attention hypothesis, if a distribution of features can be used to characterize an image, the first fixation corresponds to the point in the image which is most unlikely to belong to this distribution of features. In other words, the first fixation corresponds to a point that does not belong to the background and hence most likely is a foreground object pixel. In the first stage of this semi-automated annotation process, the first fixation point along with all its neighboring pixels are assumed to be foreground pixels. Thus a rectangle of dimensions 15% image width $\times$15% image height, centered on the fixation is assumed to be the foreground region. We will refer to this rectangle as the *foreground hypothesis rectangle*. Since the data collection process ensures that an object in an image is close to the center of the image rather than the edge, the pixels in the boundary of the images can safely be assumed to be background pixels. Hence, rectangular blocks of pixels in the 4 corners of an image are assumed to be background pixels. Each of these background blocks is 5% image width $\times$5% image height in dimension. This foreground and background pixel hypothesis is used in the next stage of the annotation process to segment the image into foreground and background. A visual representation of the background and foreground seed pixels is shown in Figure 4.6a as blue and red rectangles, respectively. The visual attention fixation is shown here as a green dot.

The foreground and background seeds thus obtained can be used to initialize a grabcut-in-one-cut algorithm [77]. This grabcut-in-one-cut algorithm, a variant of graphcut algorithm (described in Section 4.2.1) splits an input image into background and foreground. Grabcut-in-one-cut algorithm operates by labeling pixels *similar* in appearance to foreground seeds as foreground and the other pixels that appear closer to background seeds as background. However, the foreground seeds supplied by visual attention, might not produce accurate results in all cases. There are often instances when the fixation tagged as foreground constitutes edge pixels of an object. Since object edge pixels of an object are essentially discontinuities indicating a transition between foreground and background pixel distributions, the likelihood of visual attention picking edge pixels as fixations is high. If an object edge pixel is picked as the first fixation, this could result in foreground hypothesis rectangle, erroneously containing background pixels. An illustration of this effect can be seen in Figure 4.6a where the green dot at the edge of the object is the visual attention fixation. The blue foreground hypothesis rectangle around the fixation point contains some wrongly labeled background pixels. Wrongly labeled background pixels in foreground seeds supplied to graphcut algorithm can result in inaccurate segmentation, as in the case seen in Figure 4.6b. Thus, to improve the performance of the graphcut algorithm, we use a process that iteratively refines the foreground seeds. Iterative refinement works by taking the result of the graphcut segmentation, computing the centroid of the foreground pixels, and utilizing a foreground hypothesis rectangle around this centroid as input foreground seed to the next iteration of this iterative graphcut algorithm. This process is repeated until the segmentation results stabilize. In other words, this iterative graphcut process is repeated until the number of foreground pixels from the previous iteration, rejected as background by the current iteration, is within a small threshold of 5%. In mathematical terms, if F_i is the set of foreground pixels labeled by the graphcut algorithm in iteration i, then the iterative process is continued till the condition below is satisfied.

$$\frac{|F_{i-1} - F_i|}{|F_i|} \leq 0.05 \ . \tag{4.6}$$

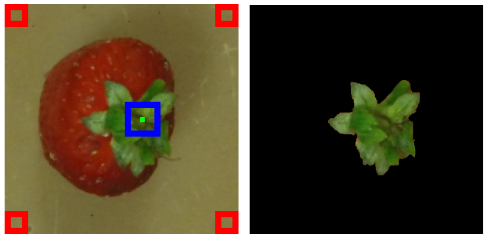


(a) (b) (c) (d)

Figure 4.6: An illustration of operation of visual attention and recursive graphcut as a part of the annotation process flow is shown here. (a) shows the visual attention fixation as a green dot. Since the visual attention fixation is on the edge of the object, the foreground hypothesis rectangle shown in blue contains background pixels. When the graphcut algorithm operates on (a) utilizing the pixels inside the blue rectangle as the foreground seeds and the pixels inside the red rectangles as the background seeds, the resulting segmented foreground region is displayed in (b). Due to the presence of background pixels in the input foreground seeds, the segmentation result obtained in (b) is inaccurate. When recursive graphcut algorithm is applied, the blue foreground hypothesis rectangle eventually moves towards the center of the object (as seen in (c)) and no longer contains background pixels. With the updated foreground hypothesis rectangle after recursive graphcut segmentation, the improved segmentation result can be seen in (d).

The change in the blue foreground hypothesis rectangle after recursive graphcut segmentation and the resulting improved foreground labeling can be seen in Figure 4.6c and Figure 4.6d respectively. When the foreground region labels from the graphcut algorithm stabilize, it is unlikely that the foreground region will change after subsequent iterations. This stabilization condition captured in (4.6), indicates that the foreground region label hypothesis generated by graphcut has converged. In case the convergence condition in (4.6) is not met after $N = 5$ iterations, further recursion is abandoned and the labeling results available after N iterations are adopted by default.

Though this process appears to be fully automated, there are cases where this automated segmentation fails. A case of segmentation failure could be due to visual attention picking up *interesting* artifacts in the image that are not necessarily part of the specimen. There are also cases where the texture of the specimen is not uniform, in which case visual attention could be biased towards a specific part of the object. This could result in graphcut segmentation only identifying the sub-region of the specimen



(a) (b)

Figure 4.7: An illustration of a case where automated annotation process fails, and human verification and correction is required. The visual attention fixation shown as a green dot in (a) is biased towards the green stalk region of the strawberry. Since this strawberry specimen exhibits binary texture–green stalk and red pulp, the combined visual attention and graphcut automated annotation approach ends up segmenting only the stalk sub-region of this strawberry specimen (see (b)). Such cases of failure calls for human verification and correction of foreground seeds to enable graphcut algorithm to segment the whole strawberry specimen.

that biased the visual attention fixation. An instance of only segmenting a sub-region of the specimen is often seen in cases of strawberries, which are characterized by the occurrence of a green stalk along with a reddish pulp. Visual attention and graphcut could end up mistakenly segmenting either the green stalk or the red pulp only. This case is illustrated in Figure 4.7. The visual attention fixation shown by a green dot in Figure 4.7a is biased towards the green stalk region of the strawberry specimen. An application of recursive graphcut on this example only ends up segmenting the green stalk region as seen in Figure 4.7b. This error in segmentation can be rectified through inclusion of appropriate foreground seeds that are representative of the complete specimen, i.e. both the green stalk and the red pulp sub-regions in case of strawberry. Therefore, to prevent inaccurate segmentation from degrading the learning set, a manual verification process is added as the last step of this annotation process. During this process, for all failed instances of automated segmentation, the foreground and background seed pixels are manually set.

This final manual verification and correction process requires human effort. However, during experiments it was noted that a very small percentage of cases needed corrective action. In summary, this semi-automated annotation process is relatively

less cumbersome than a fully manual annotation effort. An illustration of the output of this annotation process for a single specimen of orange is shown in Figure 4.8. Figure 4.8 represents a montage of 13 images of an orange specimen with the top-left component showing the segmented foreground from the image captured 32 inches away from the orange specimen. Likewise, the bottom-most row shows the image that was captured 8 inches away from the specimen. The montage in Figure 4.8 is arranged such that the 13 images, from left to right, are in the decreasing order of the heights from which the images were capture. This sequence of images in the montage wraps around after every 4 images resulting in four rows to represent 13 images.

4.4.3 Learning

The learning phase of a machine learning based object recognition method involves identifying patterns in the data that represent the presence of an object. The information fed to a machine learning algorithm is a pre-specified labeling of the data into foreground and background. The responsibility of the learning method here is to identify patterns in designated foreground regions, using information from labeled background to reject false positives. The patterns identified by the learning algorithm are then channeled into the construction of a classifier that is capable of identifying foreground pixels from an image.

In the machine learning technique proposed in this chapter, instead of using individual images with annotations of foreground and background, a collection of 13 images each featuring the same specimen from a known height is utilized. The goal is to encode the variation in appearance of a specimen from different heights to build a robust object recognition classifier. Height offers an additional dimension in the feature space for the learning algorithm to capitalize on.

This object recognition algorithm is designed to operate on images without prior segmentation. That is, the features used in this algorithm should be sufficiently generic to capture the appearance of an object directly from an image without foreground segmentation. To achieve this, we use the histogram signatures (described in

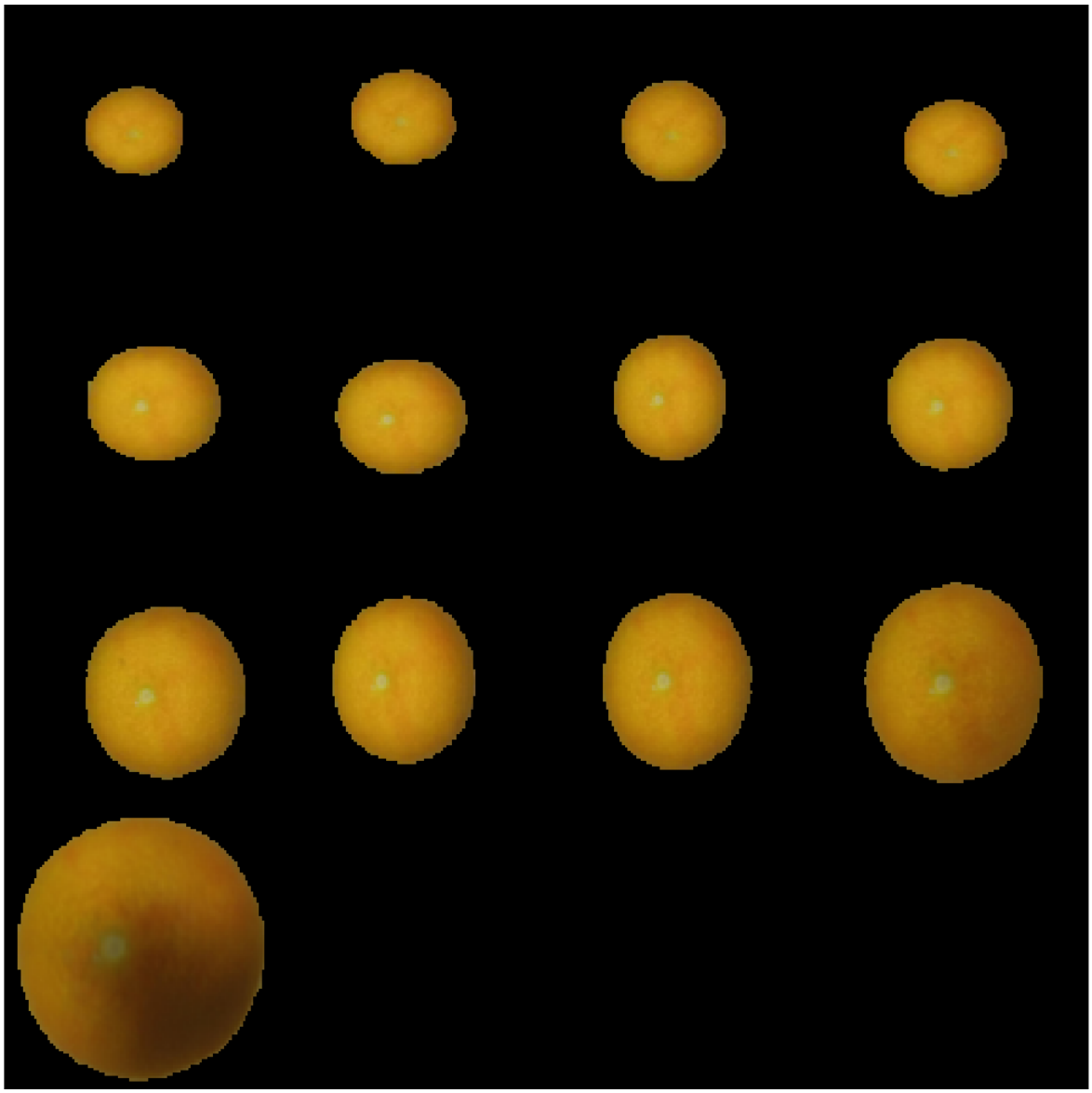


Figure 4.8: The output of the annotation process showing a montage of the segmented foreground from a set of 13 images that belong to a single orange specimen. The left-top montage component is the image of the orange specimen captured 32 inches from the ground. The montage components arranged from left to right (with wrap-around after every 4 images) to progressively show images that were captured closer to the orange specimen. The closest image that was captured is 8 inches away from the specimen and is shown in the bottom-most row.

Section 4.3.1) to extract information from an image in Hue Saturation Intensity (HSI) colorspace. The histogram information from multiple specimens of the same object class is then combined to generate a feature distribution. The details of this process are described below.

4.4.3.1 Feature Distribution

The first step in computing feature distribution is the computation of the hue, saturation and intensity histogram signatures on the labeled foreground pixels in the learning dataset. The parameters chosen for the histogram signature computation are the number of equally spaced histogram bins, $|\mathcal{B}| = 256$, and the upper bound on the number of pixels affected by noise, $e = 5\%$. Using these parameters the histogram signatures for all three components of the HSI colorspace are calculated for all 13 height-tagged images of a specimen. Note that this histogram signature computation only utilizes the labeled foreground pixels in the image. Let the histogram signature of the labeled foreground for specimen k, with height tag h, belonging to class c, on the color component $f \in \{H \text{ hue}, S \text{ saturation}, I \text{ intensity}\}$ be represented as $\mathcal{H}_f^{c_{hk}}$. Since there are 3 color components, we generate 3 histogram signatures per height-tagged specimen image. If we consider all the 13 height-tagged images available for a specimen, the total histogram signatures now available is 39 (13×3).

In the next stage of the learning process, the histogram signatures $\mathcal{H}_f^{c_{hk}}$ from all specimens belonging to a single object class are combined to generate a generalized histogram signature for the object class. In order to achieve this, the mean of all histogram signatures corresponding to a specific height-tag from all specimens of a class are combined. In other words, the mean $\bar{\mathcal{H}}_f^{c_h}$ of all histogram signatures with height-tag h are computed by taking the mean of individual components of the 256-dimensional vectors of sequence-of-histogram signatures $\mathcal{H}_f^{c_{h1}}, \mathcal{H}_f^{c_{h2}}, \mathcal{H}_f^{c_{h3}}, \ldots, \mathcal{H}_f^{c_{hm}}$, where m is the number of specimens of class c available in the learning dataset. Technically, the computation of the j-th component of the 256-dimensional mean histogram $\bar{\mathcal{H}}_f^{c_h}$ is

found as

$$\left[\bar{\mathcal{H}}_f^{c_h}\right]_j = \frac{1}{m}\sum_{k=1}^{m}\left[\mathcal{H}_f^{c_{hk}}\right]_j . \tag{4.7}$$

Since there are 13 different height-tags along with 3 different color components, each object class c is represented by a sequence of 39 (13×3) histogram signatures, which will be collectively referred to as $\bar{\mathcal{H}}^c$.

The sequence of 39 histogram signatures $\bar{\mathcal{H}}^c$ captures the generalized appearance of the objects of class c. Its important to note that the data used to compute this generalized histogram signature are composed of foreground pixels of object specimens only, without any background information. If we are to use such a generalized histogram signature generated only from foreground information for classifying objects, it is necessary to pre-process an image to detect and segment objects present in the images before this generalized histogram signature $\bar{\mathcal{H}}^c$ can be used for classification purposes. However, one of the goals of this work is to avoid segmentation while attempting classification of objects from images. To realize this goal, we revisit the learning dataset, and use the background information to build a representation that captures how inclusion of background pixels to different individual histogram signatures affects the generalized histogram signature $\bar{\mathcal{H}}^c$ of class c.

In order to use background information for building such a classifier, we first generate another set of histogram signatures. For specimen k of class c from the learning dataset, we compute a sequence of 39 histogram signatures similar to the computation involved in $\mathcal{H}_f^{c_{hk}}$, but now utilizing all pixels in the image instead of just the foreground pixels. This set of 39 histograms $\ddot{\mathcal{H}}^{c_k}$ for specimen k of class c is given by

$$\ddot{\mathcal{H}}^{c_k} = \ddot{\mathcal{H}}_f^{c_{hk}}\Big|_{h\times f} \ , \tag{4.8}$$

where $h \in \{8, 10, 12, \ldots, 32\}$ and $f \in \{H, S, I\}$.

Once we have the sequence of histogram signatures $\ddot{\mathcal{H}}^{c_k}$ of specimen k belonging to class c, along with the generalized histogram signature $\bar{\mathcal{H}}^c$ of class c, a numeric distance measure D between the histogram signatures can be computed. If we have

a series of such distance measures evaluated for a collection of specimens of class c, a distribution of these distance measures can be generated. This distribution of distance measures encodes the variability in generalized histogram signature $\bar{\mathcal{H}}^c$ induced by the presence of background pixels along with foreground pixels. In effect, this offers a way to check images for the presence of an object of class c without any prior segmentation of foreground pixels. The distance measure is the standard L_2-*norm* between two vectors.

$$d_{chfk} = D(\bar{\mathcal{H}}_f^{c_{hk}}, \ddot{\mathcal{H}}_f^{c_{hk}}) = \sqrt{\sum_{j=1}^{|B|} \left(\left[\bar{\mathcal{H}}_f^{c_{hk}}\right]_j - \left[\ddot{\mathcal{H}}_f^{c_{hk}}\right]_j \right)^2} \tag{4.9}$$

Since there are 39 histograms in total, we get a sequence of 39 d_{chfk} values for the k-th specimen of class c, and this sequence is represented as

$$D_{ck} = d_{chfk}\Big|_{h \times f} \ , \tag{4.10}$$

where $h = \{8, 10, 12, \ldots, 32\}$ and $f = \{H, S, I\}$.

A sequence of distance measures D_{ck} for each specimen k among the m specimens of class c provides a representation of a feature vector of size 39 that describes the appearance of an image that contains a specimen of class c. If we consider the feature vectors of all the m available specimens $D_{c1}, D_{c2}, \ldots, D_{cm}$, a feature distribution sequence F_c that represents the appearance of images containing objects of class c can be formulated. Lets assume that the m values $d_{chf1}, d_{chf2}, \ldots, d_{chfm}$ are drawn from a normal distribution

$$F_{chf} \sim \mathcal{N}(\mu_{chf}, \sigma_{chf}^2) \tag{4.11}$$

$$\mu_{chf} = \frac{\sum_{k=1}^{m} d_{chfk}}{m} \tag{4.12}$$

$$\sigma_{chf}^2 = \frac{\sum_{k=1}^{m} (d_{chfk} - \mu_{chf})^2}{m-1} \ . \tag{4.13}$$

Then the feature distribution sequence

$$F_c = F_{chf}\Big|_{h \times f} \tag{4.14}$$

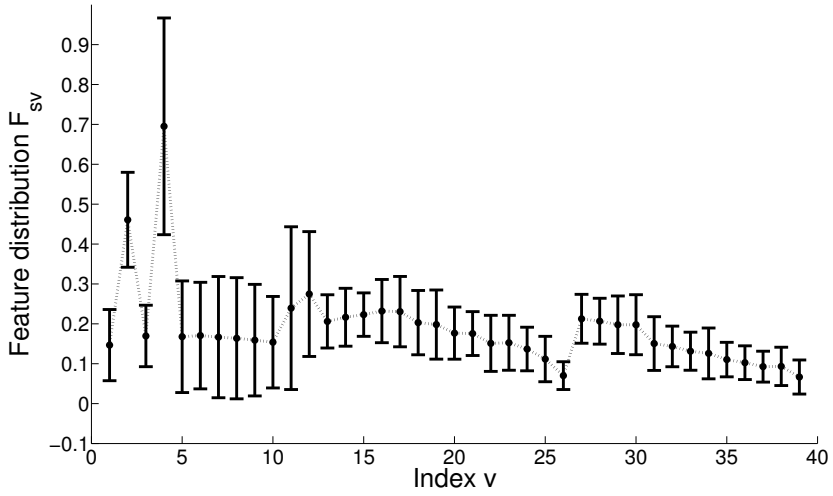
can be represented as a sequence of 39 distributions, where $h \in \{1, 2, \ldots, 13\}$ and $f \in \{H, S, I\}$.

For the two classes of objects, strawberries and oranges with 11 specimens each, the respective feature distributions F_s and F_o can thus be computed. The feature distributions F_s and F_o are shown in Figure 4.9. The 39 distributions (see (4.14)) in the distribution sequence are shown along the x-axis. Each distribution here is represented by a dot and a bar adjoining it. The dot corresponds to the mean and the bar corresponds to the 95% confidence interval, or the 2σ standard deviation interval of the distribution shown in (4.11).

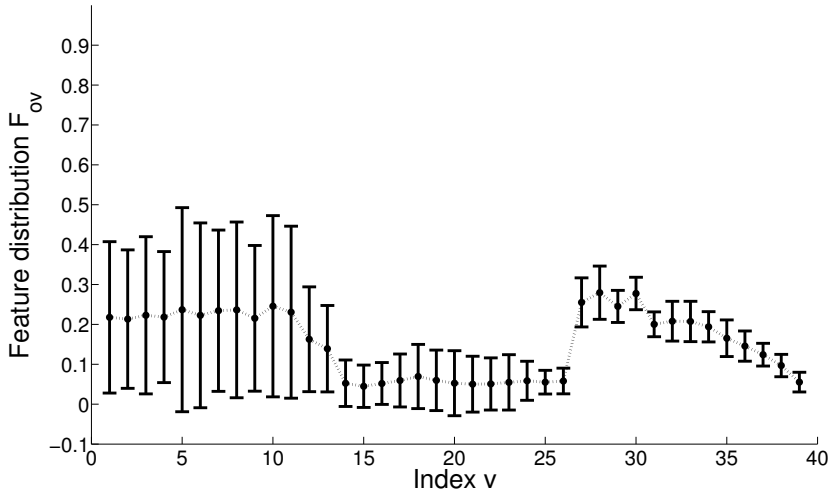
This feature distribution F_c, which comprises a sequence of 39 distributions, offers a way to test for the presence of an object of class c in an image without prior segmentation. In essence, the testing for the presence of an object can be broken down into 39 individual hypothesis tests. Each hypothesis test checks whether a certain colorspace-specific height-tagged image containing an object satisfies a specific distribution, among the 39 distributions in F_c. Not all of the 39 hypothesis tests might be equally informative– some of them could be more relevant than others. To this end, a weighting factor w_v can be attached to the v-th element in feature distribution sequence F_c. The details involved in the computation of these weight factors is described below.

4.4.3.2 Feature Weights

The feature weights offer a way to weight each distribution in the feature distribution sequence F_c. In case of a binary classification problem between objects of class p and q, we choose the feature weights such that the distributions in the sequence with lower intra-class variance are weighted higher compared to other distributions. Additionally, minimal overlap between the distributions in F_p and F_q with same index v is also another factor that increases the weight. In the rest of this section the formulation of these feature distribution sequence weights is discussed.



(a)



(b)

Figure 4.9: Strawberry feature distribution sequence F_s (a) and orange feature distribution sequence F_o (b) generated from 11 strawberry specimens and 11 orange specimens. All 39 distributions indexed by v are shown along the x-axis. The mean of each distribution is shown as a dot along with a bar that corresponds to the 95% confidence interval.

For object class p and object class q, the corresponding feature distribution sequences F_p and F_q are in both cases sequences of 39 normal distributions. Let the v-th component of sequence F_p be denoted $F_{pv} \sim \mathcal{N}(\mu_{pv}, \sigma_{pv}^2)$. The the 95% confidence interval of F_{pv} is given by

$$C_{F_{pv}}^{0.95} = [\mu_{pv} - 2 \times \sigma_{pv},\ \mu_{pv} + 2 \times \sigma_{pv}] \ . \tag{4.15}$$

The weight w_{pv} of the v component of F_p is given by

$$w'_{pv} = \frac{1}{(1 + |C_{F_{pv}} \cap C_{F_{qv}}|) \times \sigma_{pv}} \tag{4.16}$$

$$w_{pv} = \frac{w'_{pv}}{\sum_j w'_{pj}} \tag{4.17}$$

where $|C_{F_{pv}} \cap C_{F_{qv}}|$ corresponds to the length of the interval obtained after the intersection of intervals $C_{F_{pv}}$ and $C_{F_{qv}}$. The first term in the denominator of (4.16) penalizes distributions whose confidence intervals share common positions with an identical indexed distribution in the other class. In other words, this term penalizes distribution F_{pv} whose confidence interval overlaps with intervals of an identical indexed distribution F_{qv}. The second term in the denominator of (4.16) penalizes distributions with high intra-class variance. Ultimately, the weights are normalized as shown in (4.17). The 39 feature weight sequence thus computed for a class will collectively be referred to as W_c. For the binary classification problem, between strawberry and orange specimens, the weight factors computed using (4.17) are shown in the bar plot in Figure 4.10. The black bars show the strawberry class weights and the red bars show the corresponding orange class weights.

The distribution components in F_p with higher values of w_p are weighted higher than the other distributions while performing hypothesis tests that will be discussed. This is discussed in the next section on validation phase of this machine learning algorithm.

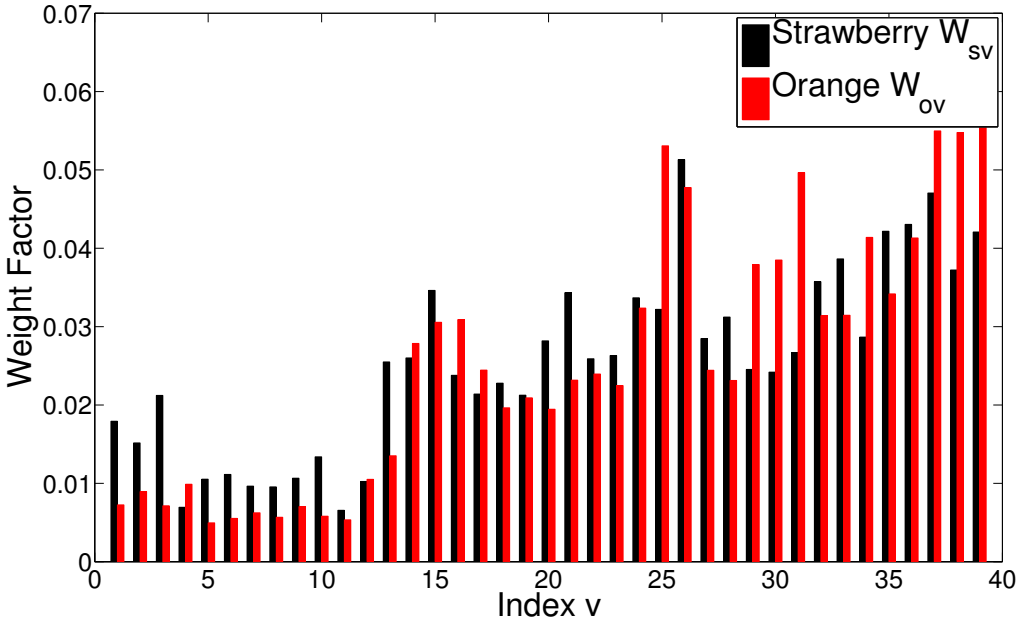


Figure 4.10: The feature weight sequences for strawberry class F_s (black bars) and orange class F_o (red bars)

4.4.4 Validation

In this object recognition method, a series of height-tagged images of an object specimen is used to perform a binary classification task. In other words, the 13 height-tagged images are used to decide if the object belongs to one of the two classes: p or q. In order to achieve this, the histogram signatures (described in Section 4.3.1) are computed on each of the 13 height tagged-images available for an object specimen. Since there is no pre-specified label information available, all pixels in the image are used here. The computation performed is identical to the one in (4.8), and results in a sequence of 39 histogram signatures $\ddot{\mathcal{H}}^k$.

The next step in the validation procedure is to use (4.9) and (4.10) to compute the distance measure sequences $D_k^p = D(\ddot{\mathcal{H}}^k, \bar{\mathcal{H}}^p)$ and $D_k^q = D(\ddot{\mathcal{H}}^k, \bar{\mathcal{H}}^p)$. The 39-value sequences D_k^p and D_k^q provide information about the *closeness* of the sample specimen k to each of the classes p and q. The information in these sequences need to be processed further in order to associate specimen k with either class p or class q.

Let us assume that specimen k belongs to class p. According to this hypothesis, the v-th component of D_k^p should follow the v-th distribution of the feature distribution sequence F_p. In other words, using (4.15), we can state with 95% *confidence* that condition

$$d_{kv}^p \in C_{F_{pv}}^{0.95} = [\mu_{pv} - 2 \times \sigma_{pv},\ \mu_{pv} + 2 \times \sigma_{pv}] \tag{4.18}$$

is satisfied. If we call this hypothesis test h_{kv}^p, then passing of this hypothesis test

$$h_{kv}^p = \begin{cases} 1 & d_{kv}^p \in C_{F_{pv}}^{0.95} \\ 0 & otherwise \end{cases} \tag{4.19}$$

meaning $(h_{kv}^p = 1)$, determines that specimen k belongs to class p.

There are 39 such hypothesis tests, $h_{k1}^p, h_{k2}^p, \ldots, h_{k39}^p$ that can be performed to determine if specimen k belongs to class p. The importance of each of these hypothesis tests for associating specimen k with class p is determined by the corresponding weight w_{pv}. The information available in the form of hypothesis tests, along with their associated weights, is combined in to a single numeric metric or *class confidence*

$$H_k^p = \sum_{v=1}^{39} h_{kv}^p w_{pv} \tag{4.20}$$

that measures the confidence in specimen k belonging to class p.

It is important to note that based on (4.17), it follows that $H_c^p \in [0, 1]$. If $H_k^p = 1$, then we can say with high confidence that specimen k belongs to class p. Alternatively, $H_k^p = 0$ suggests that its unlikely that specimen k belongs to class p. A similar metric H_k^q is computed to determine confidence in the assertion that specimen k belongs to class q. Finally, the binary classification task that classifies specimen k into class p or class q is decided based on the magnitude of H_k^p and H_k^q:

$$\begin{aligned} H_k^p > H_k^q &\qquad \text{specimen } k \in \text{class } p \\ H_k^p < H_k^q &\qquad \text{specimen } k \in \text{class } q \\ H_k^p = H_k^q &\qquad \text{Undetermined} \end{aligned} \tag{4.21}$$

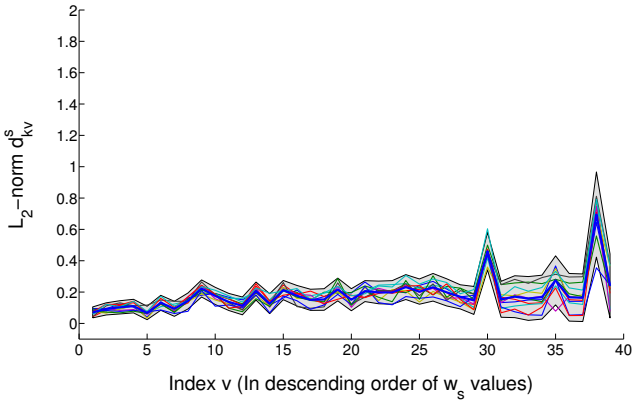
4.5 Results

The data collected during the experiments involve images from 11 specimens of both orange class and strawberry class. Data on each specimen included images taken from 13 different heights, starting at 32 inches from the ground, down to 8 inches, away from the ground. The imaging experiment was conducted in a test tank filled with water using the imaging rig shown in Figure 4.1. More details of the data collection process can be found in Section 4.4.1. Following the data collection process, the feature distribution sequence F_s for strawberry class, and F_o for orange class, are computed using the procedure discussed in Section 4.4.3.1; the results are shown in Figure 4.9.

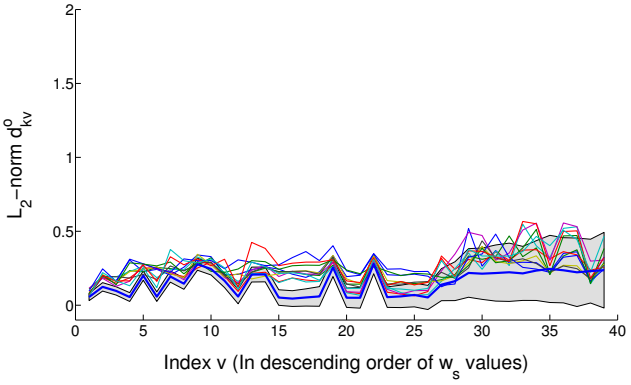
As part of an initial validation step, both learning and testing is done on the entire dataset of 22 specimens. The learning procedure results in feature distributions for strawberry and orange which are are shown in Figure 4.9. Then each specimen is evaluated against both strawberry and orange feature distributions as described in Section 4.4.4, to get distance metric sequences D_k^s and D_k^o. The 11 strawberry specimens are checked against the strawberry and orange feature distribution sequences F_s and F_o; see Figure 4.11. The solid blue line in Figure 4.11a along with the gray band is an alternate representation of strawberry feature distribution sequence F_s (shown in Figure 4.9a). The solid blue line represents the mean of the feature distributions along with the 95% confidence region shown as a gray band. The additional difference in this representation of F_s is that the indices v are now arranged in the decreasing order according to their importance, with the latter inferred by the strawberry feature weights W_s. In other words, the first distribution in Figure 4.11a corresponds to the feature distribution with the highest feature weight in W_s, and the last distribution is the feature distribution with the lowest feature weight in W_s. The colored lines in Figure 4.11a, represent the distance metric sequences D_k^s of the 11 strawberry specimens. All the 11 colored lines lie inside the confidence region indicating that the distance metric sequence of strawberry specimens are in agreement with the strawberry-class distribution sequence F_s. In Figure 4.11b, we compare the same 11 strawberry specimens against

the orange-class distribution sequence F_o. Here the orange-class distribution sequence F_o is again represented by the combination of solid blue line with a gray band around it. However, since only strawberry specimens are being considered here, the sequence of feature distributions are again arranged in the decreasing order of importance as captured in the strawberry feature weights W_s. The colored lines here represent the distance metric sequence D_k^o values for the 11 strawberry specimens. In contrast with Figure 4.11b, the colored lines are not in agreement with the confidence region of the orange-class feature distribution sequence F_o. In Figure 4.11a we can observe that the strawberry distance metric sequence D_k^s almost always lies inside the confidence region while compared with Figure 4.11b where the D_k^s often lie outside the confidence region. This is a clear indicator of strawberry specimens matching better with the distribution of strawberry-class feature distribution compared to orange-class feature distribution. On similar lines, the 11 orange specimens are compared against the orange-class feature distribution F_o in Figure 4.12a and also compared against strawberry-class feature distribution F_s in Figure 4.12b. The observations show that the orange specimens match closer with orange feature distribution sequence F_o rather than the strawberry feature distribution sequence F_s. This again reinforces that the distance metric values of both orange and strawberry specimens agree with their corresponding class distribution sequences.

A more concrete validation of this machine learning approach is by computing the *class confidence* H_k^s and H_k^o of each specimen, and deciding on the class of the specimen as described in Section 4.4.4. One thing to note here is that the class confidence measure requires the feature distribution sequences of the strawberry and orange classes to be computed. The feature distribution sequences is generated from the learning set; the learning set contains the specimen we are currently trying to recognize. It is a general rule to separate the learning and testing datasets before validating a

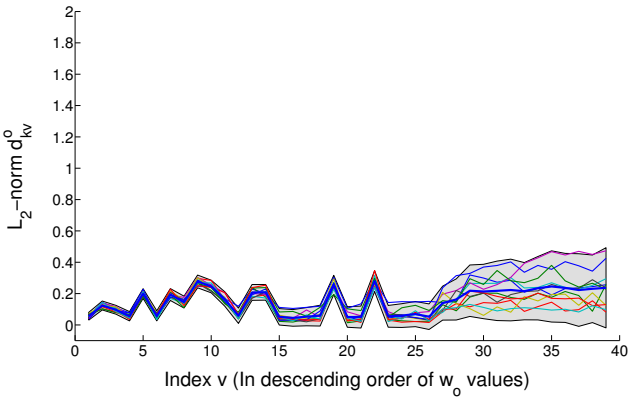


(a) Strawberry specimens evaluated against strawberry feature distribution sequence F_s

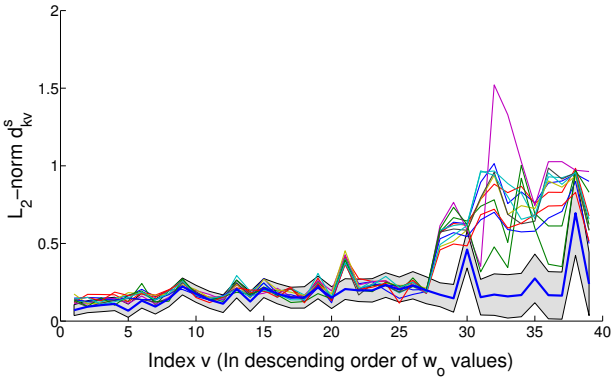


(b) Strawberry specimens evaluated against orange feature distribution sequence F_o

Figure 4.11: Validation results obtained while evaluating strawberry specimens against strawberry-class feature distribution sequence F_s (a) and against the orange-class feature distribution sequence F_o (b). The strawberry-class feature distribution sequence F_s is shown in (a): mean is represented as the solid blue line and the 95% confidence region is shown as a gray band around the mean. The sequence of feature distributions indexed by v is arranged in the decreasing order of importance as captured in the strawberry feature weights W_s. The colored lines in (a), represent the distance metric sequence D_k^s of the 11 strawberry specimens. All the 11 colored lines in (a) mostly lie inside the confidence region indicating that the distance metric values of strawberry specimens are in agreement with the strawberry-class distribution sequence F_s. In (b), the same 11 strawberry specimens are checked against the orange-class feature distribution sequence F_o. In this case, the solid blue line and gray confidence region are representative of the orange-class feature distribution sequence F_o. However, since only strawberry specimens are considered here, the sequence of feature distributions are again arranged in the decreasing order of importance as captured in the strawberry feature weights W_s. In contrast to (a), the colored lines in (b) do not agree with the orange-class feature distribution sequence F_o. This signifies that the strawberry specimens only agree with the strawberry feature distribution sequence F_s and not with orange distribution sequence F_o.



(a) Orange specimens evaluated against orange feature distribution sequence F_o



(b) Orange specimens evaluated against strawberry feature distribution sequence F_s

Figure 4.12: Similar to Figure 4.11 that show the validation results for strawberry specimens, this figure shows the validation results for orange specimens. Testing of orange specimens against orange-class feature distribution sequence F_o is shown in (a), and testing against the strawberry-class feature distribution sequence F_s is shown in (b). The orange-class feature distribution sequence F_o is shown in (a): mean is represented as the solid blue line and the 95% confidence region is shown as a gray band around the mean. The sequence of feature distributions indexed by v is arranged in the decreasing order of importance as captured in the orange feature weights W_o. The colored lines in (a), represent the distance metric sequence D^o_k of the 11 orange specimens. All the 11 colored lines in (a) mostly lie inside the confidence region indicating that the distance metric values of orange specimens are in agreement with the orange-class distribution sequence. In (b), the same 11 strawberry specimens are evaluated against the strawberry-class feature distribution sequence F_s. In this case, the solid blue line and gray confidence region are representative of the strawberry-class feature distribution sequence F_s. However, since only orange specimens are evaluated here, the sequence of feature distributions are again arranged in the decreasing order of importance as captured in the orange feature weights W_o. In contrast to (a), the colored lines in (b) do not agree with the strawberry-class feature distribution sequence F_s. This signifies that the orange specimens only agree with the orange feature distribution sequence F_o and not with strawberry distribution sequence F_s.

machine learning algorithm. In this case, since the dataset consists of only 22 specimens, a leave-one-out cross-validation [63] was implemented.[1] The class confidence values H_k^s and H_k^o of the 11 strawberry specimens are shown as black and red bars respectively in Figure 4.13a. It is clear that the black bars are taller than the red bars in all cases. According to the hypothesis test in force (see (4.19)), this implies that that all strawberry specimens are classified correctly. The same can be said for the orange specimens. As seen in Figure 4.13b: all the red bars are taller than the back bars. Therefore, all orange specimens are also classified correctly.

This validation procedure confirms the ability of this machine learning technique to combine data from multiple images of a specimen to perform a binary classification task. The 22 specimens drawn from strawberry and orange classes were all correctly classified by the proposed machine learning algorithm.

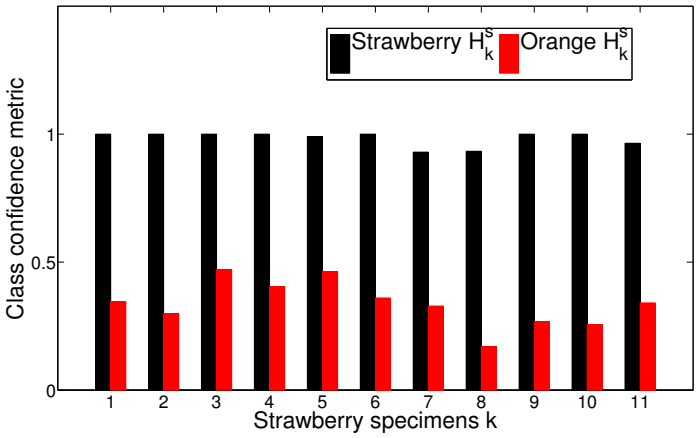
4.6 Discussion

The machine learning approach developed here was used to perform binary classification on a set of 22 specimens containing equal number of strawberry and orange specimens. The graphical results from Figures 4.11 and 4.12 show the conformance of the distance measures D_k^s and D_k^o of specimens only to the class they belong to. This is further reinforced by the class confidence measures H_k^s and H_k^o, which assigned each specimen to the right class. All the specimens were thus classified correctly, hence achieving 100% precision and recall rates.[2]

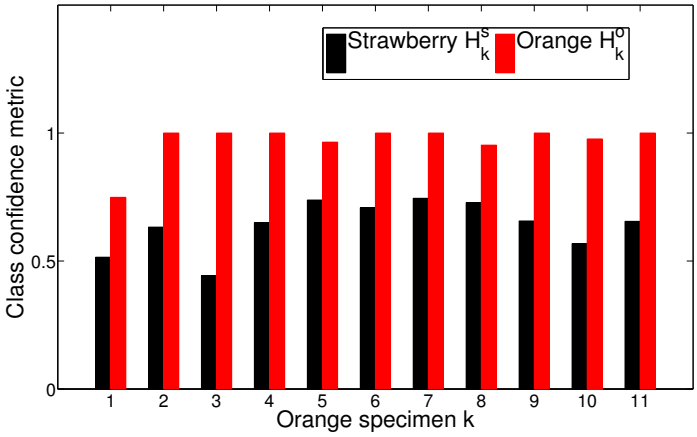
---

[1] Leave-one-out cross-validation is used in cases where the number of specimens available is very small. In such cases, one does not have the luxury of splitting the available labeled dataset into learning and testing sets. Therefore, in leave-one-out cross-validation, during each iteration one specimen is chosen as the testing set and the all other specimens are chosen as the learning set. Thus the learning procedure is repeated in each iteration, without adding the instance that is being tested to the learning set.

[2] *Precision* and *Recall* are performance measures for a machine learning algorithm. Precision is the ratio of true positives to the total number of detections and recall is the ratio of true positives to the total number of positives in the dataset.



(a)



(b)

Figure 4.13: The classification results for the 11 strawberry and 11 orange specimens are shown in (a) and (b) respectively. The class confidence values H_k^s and H_k^o are shown by black and red bars. In the case of strawberry specimens in (a), the black bars are taller than the red bars indicating that all strawberry specimens are classified correctly based on the hypothesis test of (4.21). Same can be said in case of orange specimens shown in (b) where the red bars are taller, indicating that all orange specimens are classified correctly.

The theoretical best possible performance obtained here portrays the potential of this algorithm to work in noisy natural images. One source of noise in the data collection process has been a collection of floating specks of dust in the water, along with dust on the floor of the tank. There were even instances when the foot of the person operating the imaging-rig was included in the images collected. In addition, the uncontrolled lighting conditions introduced noise in the form of light artifacts in the floor of the tank. The effect of uneven lighting conditions under which the data collection process took place is visible in Figure 4.3. The ability of this algorithm to produce consistent high performance even in the presence of noise is evidence of its ability to handle unpredictable environmental conditions.

Since only 22 objects were used for this particular evaluation, testing this algorithm on a natural image dataset containing a few thousand specimens would definitively be interesting. Obtaining multiple images of the same specimen from different heights and annotating large learning sets, might appear cumbersome at first. However the semi-automated annotation procedure provided in Section 4.4.2 alleviates considerably the manual labor involved.

Since this approach relies on multiple images for each specimen, imaging time and effort involved is typically higher than an object recognition algorithm that is designed to operate on a single image of a specimen. On the other hand, the use of multiple images and 39 independent hypothesis tests, contribute to identification of a target specimen with higher confidence than by an approach that just uses a single image of a target. Multiple images offer a way to extract features of an object from different scales, which when combined together offer a robust object recognition framework. In cases where it is critical to classify a target with confidence, for instance in case of detecting unexploded mines or submerged ordinance, an approach that is so accurate is valuable despite the additional imaging time required.

4.7 Conclusion

The machine learning method developed here offers a way to perform binary classification using global descriptors generated from multiple images of a specimen. This technique was successful in recognizing all the 22 specimens available in our dataset of 11 strawberries and 11 oranges. The performance of this technique on this (admittedly small) dataset is the absolute theoretical maximum that can be obtained for a machine learning technique. This exceptional performance, despite the presence of noise and several uncontrolled variables in the data collection process, is indicative of the robustness of this algorithm. Additionally, the ability of this algorithm to perform object recognition via histogram based global descriptors, avoids segmentation entirely during the testing phase. Since segmentation can be problematic in noisy images where the edges of objects is hard to determine, doing away with segmentation is a great incentive to adopt this technique for noisy images.

4.8 Future work

This algorithm has been tested on a set of 22 images collected using an imaging rig. It will be interesting to expand this technique to natural datasets where multiple instances of the same target object have been captured. The current procedure adopted here computes a set of discrete measurements from a fixed set of heights from which images are available. Expanding this to work with an arbitrary but known set of heights will be valuable for object recognition tasks where the data collected cannot adhere to such predetermined height standards. Expanding this binary classification problem into a multi-class classification problem for a larger number of object classes is another natural direction that can be explored.

Chapter 5

COOPROV: LOW COST UNDERWATER REMOTELY OPERATED VEHICLE

5.1 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 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 3 and the multi-view object recognition 4 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.

5.2 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 [79] 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 [80] 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 [81] and the Fathom One [82]. 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 [83] or Mission Oriented Operating Suite (MOOS) [84]. 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.

5.3 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 [81], 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 [85] in the backend to drive the robot.

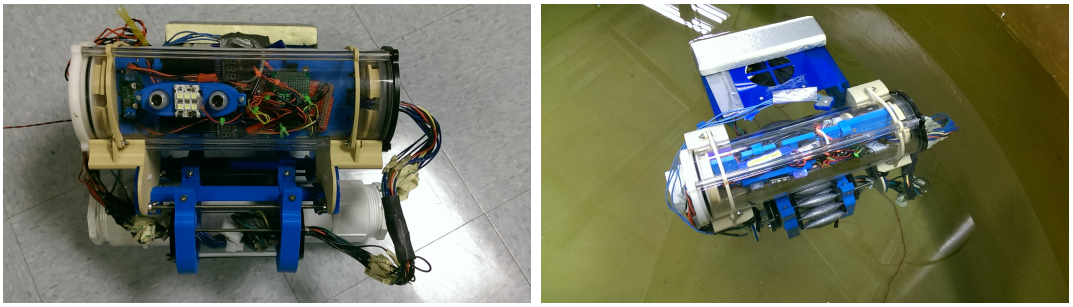


Figure 5.1: CoopROV shown in two different views

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 5.1.

5.3.1 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 5.1: CoopROV Specifications

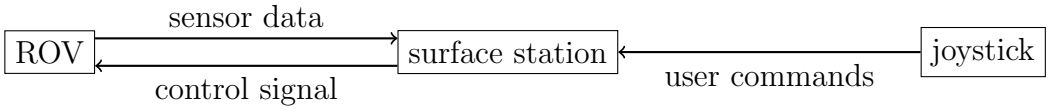


Figure 5.2: Block diagram of high-level data flow between different components in CoopROV

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 5.2. The computing on the robot is handled by a RaspberryPi [86] and Arduino Mega[87]. 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 5.1. A more detailed account of different subsystems comprising hardware, electronics, software, sensors and power supply is given in the following sections.

5.3.2 Hardware

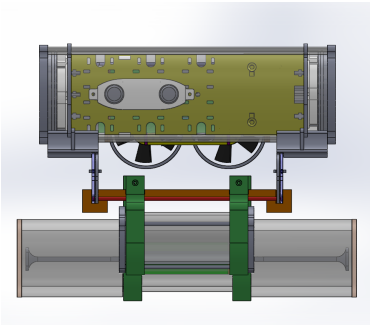
The exterior shell of CoopROV is composed of an acrylic frame that holds all housing tubes together. There are 3 housing tubes in total: an electronics tube, a connector tube and a battery tube. A Computer Aided Design (CAD) model of CoopROV in Figure 5.3 illustrates the position of the housing tubes 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.

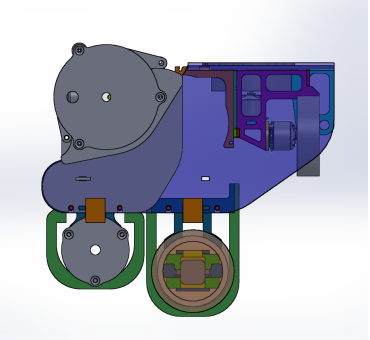
5.3.3 Electronics

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.

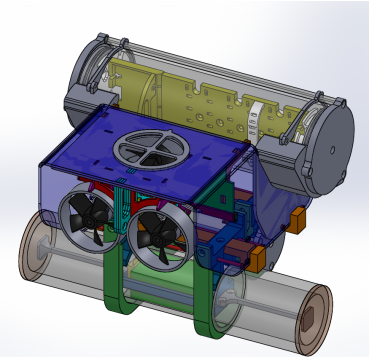
There are two parallel data streams. One transfers the control commands that flow through the system and other sends the sensor data being logged from the various sensors. The motion control commands that come from the surface station are fed by the RaspberryPi into the Arduino before being translated 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 5.4 provides a graphical overview of the connections between the major electronic components on CoopROV. It its important to note that only one



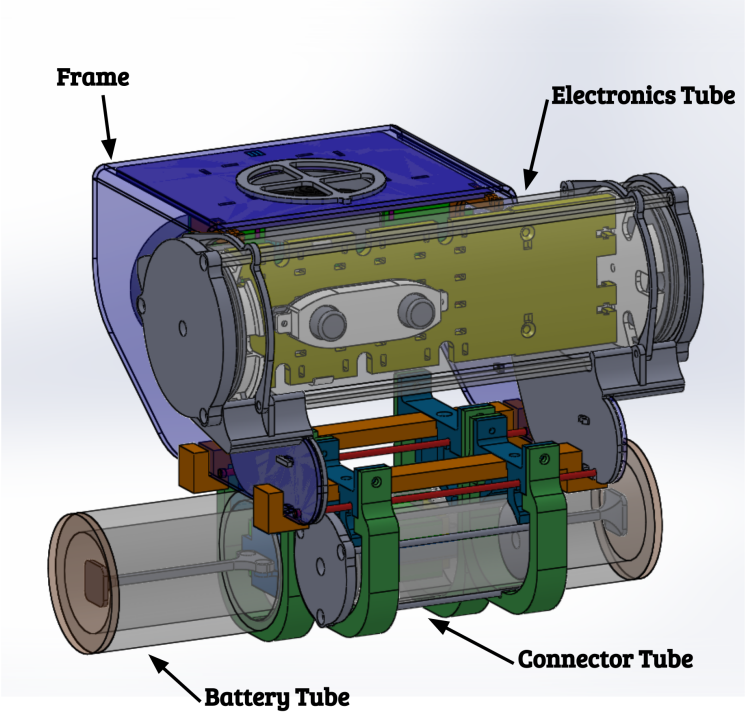
(a) Front view



(b) Side view



(c) Isometric view (back)



(d) Isometric view (front)

Figure 5.3: Different views of the CoopROV CAD model with some labeled parts

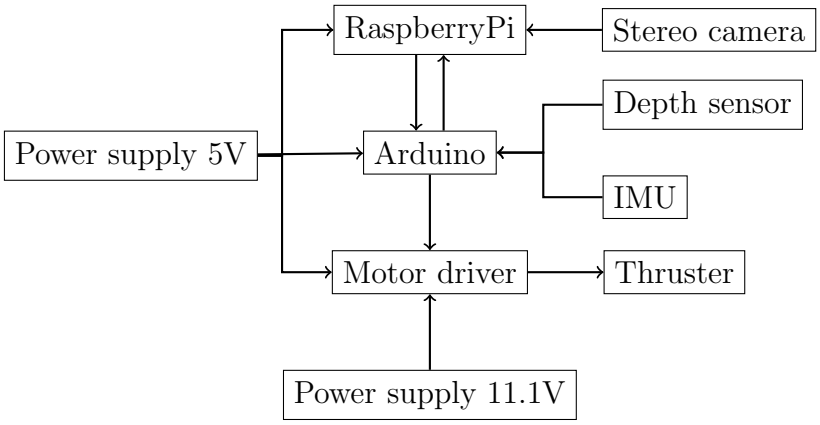


Figure 5.4: Block diagram depicting the connectivity between major components in the electronics schematics

motor and motor driver are shown in Figure 5.4, but in the actual system, there are 3 motors with dedicated motor drivers. Figure 5.4 also shows the two power supplies in the system. A detailed exposition on power supply components can be found in Section 5.3.4.

5.3.4 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 5.2: 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 5.4 represents the different power supplies and their connection to the different components that depend on them. Table 5.2 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.

5.3.5 Sensors

CoopROV comes with 4 sensorsp; namely, stereo cameras, IMU, depth sensor and voltage sensors. A low cost off-the-shelf Minoru 3D webcam [88] 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.

5.3.6 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 5.5.

5.4 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.

5.5 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

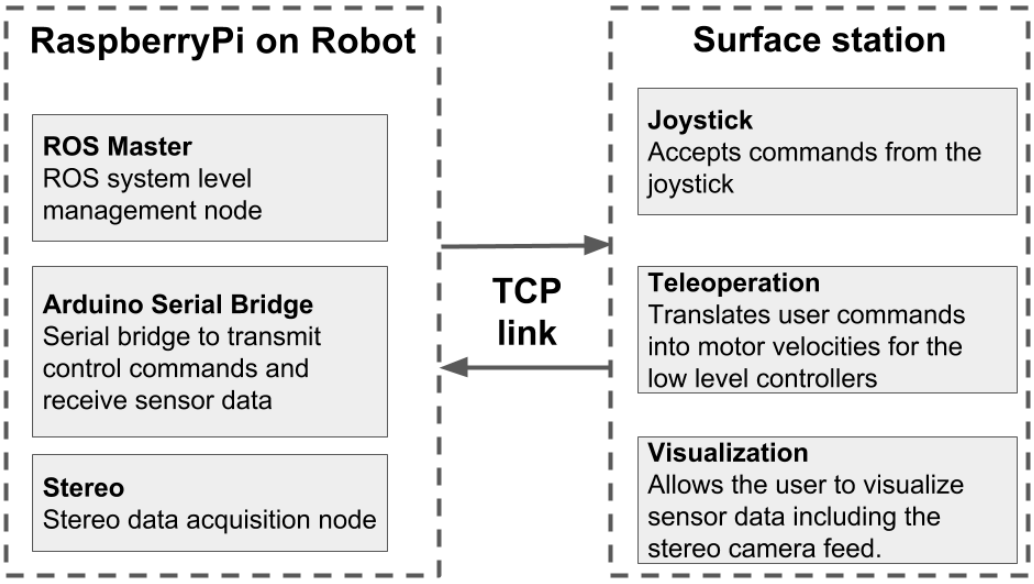


Figure 5.5: CoopROV software architecture showing all the ROS-nodes along with their functionality explained below.

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

5.6 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 3 and the multi-view object recognition 4 discussed in this dissertation.

Chapter 6

CONCLUSIONS AND OUTLOOK

The work in this dissertation offers robotic image analysis tools designed to operate in noisy natural environments. Underwater environment is the primary domain used to validate these tools. The two applications that drove the design of these object recognition tools are the automated subway car recognition and scallop recognition problems. Additionally, a multi-view object classifier, that can operate in noisy images without segmentation, is also proposed. Finally, an low cost ROV, CoopROV, was built to facilitate underwater experiments. The different insights gleaned from development of each of these tools will be discussed in the remainder of this section.

Chapter 2 showcases the application of eigen-value based shape descriptors to the subway car recognition problem. Eigen-value based shape identification is a tool that can be used to identify simple geometric shapes. By reducing subway car recognition to rectangle matching, eigen-value based shape descriptor was successful in recognizing subway cars from seabed images. Though shape matching can aid object recognition, other cues like texture can often play an important role in distinguishing an object. Since eigen-value shape descriptors only use shape information, they are not suited for applications where non-shape cues are more relevant in encoding object information. Additionally, while evaluating eigen-value shape descriptors, it was determined that discretization errors and segmentation errors can significantly impair the performance of eigen-value based shape descriptors. Therefore, eigen-value shape descriptors is a useful tool for identifying objects, provided we can guarantee that the shape of the object can be represented with minimal discretization 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 requirements are very restrictive, and render recognition tasks involving complex shapes problematic. One such challenging problem is scallop recognition. The visual cues presented by a scallop comprise its distinguishing crescent profile, along with its unique representative texture. Since eigen-value descriptors neither handle complex shape profiles like crescents, nor do they capture textural information, a more refined approach is warranted. This led to the development of multi-layered object recognition framework proposed in Chapter 3.

Marine biologists and oceanographers often depend on large natural image datasets to study benthic phenomenon. With the availability of robotic imaging vehicles, that generation datasets ranging millions of images are becoming increasingly common. Automated image processing tools are necessary to deal with such large image datasets. Most existing techniques are built on restrictive assumptions that do not generalize to noisy natural image datasets. The multi-layered object recognition approach described in Chapter 3 addresses this problem by providing researchers with a modular architecture that can scale to large natural image datasets. Modular architectures also contains the built-in flexibility to be reconfigured to solve varied object recognition problems. This multi-layered framework was tasked with automated scallop recognition from AUV images, as a means to study scallop population. This framework was successful in recognizing 60% of scallops in a dataset of over 8000 images. The uniqueness of this approach lies in its ability to handle noisy natural images under varied environmental conditions. To improve the performance of the classifier proposed here, and thereby reduce false positives, more information about a target object can be helpful. To inject this additional information into the classification process, a multi-view object recognition algorithm discussed in Chapter 4 was formulated.

The multi-view approach discussed in Chapter 4 is machine learning technique designed for binary classification tasks. The objective here is to formulate a robust object classifier even in case of noisy image data. Since noisy single-views of an object might often not contain sufficient information to unambiguously recognize it, information from multiple views are combined here to build a machine learning classifier.

In this approach, the information from 13 images of each object specimen, captured from different heights, is encoded into a single object model. Another salient character of this approach is the use of histogram-based global feature descriptors that do not require segmentation of object pixels. Since segmentation can be challenging in noisy natural images, the lack of need for segmentation greatly boosts the appeal of this method. This method is evaluated on a combined dataset of 22 specimens comprising oranges and strawberries specimens. All the specimens are correctly classified. Despite the small dataset of 22 images used, the exceptional performance of this classifier in noisy underwater data is encouraging. To decrease false positives, an approach like this could be used as a classification layer in a multi-layered approach, like the one described in Chapter 3.

To collect data in underwater environments, a submersible robotic vehicle is often required. Most commercially available robotic vehicles are either expensive or difficult to customize for research needs. To address this problem, CoopROV, a low cost ROV was designed as a research prototype to support underwater experiments. CoopROV carries an IMU, stereo cameras, depth and pressure sensors. The onboard electronics on the robot allows easy inclusion of new sensors. CoopROV also offers a ROS software interface that allows easy access to plethora of existing software tools. The low cost, and innate flexibility to modify the software and hardware, makes CoopROV an ideal platform to support underwater experiments.

There are some possible directions to extend the object recognition tools proposed in this dissertation. For instance the scallop recognition effort in Chapter 3 could benefit from using more descriptive templates. In Chapter 3, we see that the appearance of a scallop is characterized by two crescents near the scallop rim: one bright and one dark. Currently, only the position of a dark crescent is captured by the templates used. Using templates that capture additional information, like the position of the bright crescent could be interesting. As for the multi-view object recognition method discussed in Chapter 4, a more expansive dataset could be used to test this algorithm. Another possible improvement is to generalize this multi-view algorithm

to accept images from any series of heights instead of a fixed set of heights. Easing the restrictions on the different views needed by the multi-view algorithm could make the experiments and data collection effort easier. This multi-view approach could also be expanded to multi-class classification for dealing with more than 2 object classes. The CoopROV frame could be redesigned to improve the trim (weight distribution) of the vehicle. Designing controllers and building a localization system are other avenues where future improvements on CoopROV is possible.

This dissertation offers object recognition tools designed to work on noisy natural images. Each of these tools have been validated on different underwater applications. The developments here are intended to provide automated solutions to researchers dealing with large natural image datasets. The availability of modular reconfigurable architectures to allow researchers to build custom solutions to solve their object recognition needs is the prime objective. The techniques proposed here are first steps in providing tools capable of handling noisy natural images. Further exploration on this domain is necessary to build more robust architectures that can recognize multiple classes of objects. Some examples of object recognition applications like subway car and scallop recognition are not all encompassing but provide an initial thrust in this domain. Further work to build extensive tools required by the marine research community is warranted. The overarching goal of this dissertation is to contribute to the object recognition tools available to strengthen the perception capabilities of robots. Ultimately, better scene understanding and perception abilities allow robotic systems to move towards autonomy.

BIBLIOGRAPHY

[1] D. Silver, A. Huang, CJ Maddison, A. Guez, L. Sifre, GVD Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, and M. Lanctot. Mastering the game of go with deep neural networks and tree search. *Nature*, 529(7587):484–489, 2016.

[2] M. Campbell, AJ Hoane, and F. Hsu. Deep blue. *Artificial Intelligence*, 134(1):57–83, 2002.

[3] K. Nagatani, S. Kiribayashi, Y. Okada, K. Otake, K. Yoshida, S. Tadokoro, T. Nishimura, T. Yoshida, E. Koyanagi, and M. Fukushima. Emergency response to the nuclear accident at the fukushima daiichi nuclear power plants using mobile rescue robots. *Journal of Field Robotics*, 30(1):44–63, 2013.

[4] BM Yamauchi. Packbot: a versatile platform for military robotics. In *Defense and Security*, pages 228–237. International Society for Optics and Photonics, 2004.

[5] Predator MQ-1B. http://www.af.mil/AboutUs/FactSheets/Display/tabid/224/Article/104469/mq-1b-predator.aspx. Accessed: 2016-10-20.

[6] SJ Russell, P. Norvig, JF Canny, JM Malik, and DD Edwards. *Artificial intelligence: a modern approach*, volume 2. Prentice hall Upper Saddle River, 2003.

[7] FA Wichmann, J. Drewes, P. Rosas, and KR Gegenfurtner. Animal detection in natural scenes: critical features revisited. *Journal of Vision*, 10(4):6–6, 2010.

[8] Aude Oliva and Antonio Torralba. Modeling the shape of the scene: A holistic representation of the spatial envelope. *International journal of computer vision*, 42(3):145–175, 2001.

[9] SS Haykin. *Adaptive filter theory*. Pearson Education India, 2008.

[10] K. Irie, AE McKinnon, K. Unsworth, and IM Woodhead. A technique for evaluation of ccd video-camera noise. *IEEE Transactions on Circuits and Systems for Video Technology*, 18(2):280–284, 2008.

[11] R. Garcia, T. Nicosevici, and X. Cuf. On the way to solve lighting problems in underwater imaging. In *OCEANS'02 MTS/IEEE*, volume 2, pages 1018–1024. IEEE, 2002.

[12] J. Ahln, E. Bengtsson, and D. Sundgren. Evaluation of underwater spectral data for colour correction applications. In *Proceedings of the 5th WSEAS International Conference on Circuits, Systems, Electronics, Control & Signal Processing*, pages 321–326. World Scientific and Engineering Academy and Society (WSEAS), 2006.

[13] JS Jaffe. Underwater optical imaging: the past, the present, and the prospects. *IEEE Journal of Oceanic Engineering*, 40(3):683–700, 2015.

[14] 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.

[15] GIMP despeckle plugin. http://oldhome.schmorp.de/marc/pdb/plug_in_despeckle.html. Accessed: 2016-10-23.

[16] C. McGavigan. A quantitative method for sampling littoral zooplankton in lakes: The active tube. *Limnology and Oceanography: Methods*, 10:289–295, 2012.

[17] CP Stelzer. Automated system for sampling, counting, and biological analysis of rotifer populations. *Limnology and Oceanography: Methods*, 7:856, 2009.

[18] K. Enomoto, M. Toda, and Yasuhiro Kuwahara. Scallop detection from sand-seabed images for fishery investigation. In *2nd International Congress on Image and Signal Processing*, pages 1–5. IEEE, 2009.

[19] K. Enomoto, M. Toda, and Y. Kuwahara. Extraction method of scallop area in gravel seabed images for fishery investigation. *IEICE Transactions on Information and Systems*, 93(7):1754–1760, 2010.

[20] 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*, pages 22–40, 2015.

[21] T. Schoening. *Automated detection in benthic images for megafauna classification and marine resource exploration: supervised and unsupervised methods for classification and regression tasks in benthic images with efficient integration of expert knowledge*. PhD thesis, Universität Bielefeld, 2015.

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

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

[24] 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.

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

[26] 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.

[27] 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.

[28] 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.

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

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

[31] 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.

[32] 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.

[33] 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.

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

[35] 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.

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

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

[38] 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.

[39] RJ Webster. PhD thesis, 2013.

[40] AL Forrest, ME Wittmann, V. Schmidt, NA Raineault, A. Hamilton, W. Pike, SG Schladow, JE Reuter, BE Laval, and AC Trembanis. Quantitative assessment of invasive species in lacustrine environments through benthic imagery analysis. *Limnology and Oceanography: Methods*, 10:65–74, 2012.

[41] C. Spampinato, YH Chen-Burger, G. Nadarajan, and RB Fisher. Detecting, tracking and counting fish in low quality unconstrained underwater videos. In *3rd International Conference on Computer Vision Theory and Applications*, pages 514–519. Citeseer, 2008.

[42] DR Edgington, DE Cline, D. Davis, I. Kerkez, and J. Mariette. Detecting, tracking and classifying animals in underwater video. In *Oceans'06 MTS/IEEE-Boston Conference and Exhibition*, pages 1–5. IEEE, 2006.

[43] RN Williams, TJ Lambert, AF Kelsall, and T. Pauly. Detecting marine animals in underwater video: Let's start with salmon. In *Americas Conference on Information Systems*, volume 1, pages 1482–1490, 2006.

[44] B. Zion. The use of computer vision technologies in aquaculturea review. *Computers and Electronics in Agriculture*, 88:125–132, 2012.

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

[46] R. Fearn, R. Williams, M. Cameron-Jones, J. Harrington, and J. Semmens. Automated intelligent abundance analysis of scallop survey video footage. *AI 2007: Advances in Artificial Intelligence*, pages 549–558, 2007.

[47] National Marine Fisheries Service Northeast Fisheries Science Center (NEFSC). 50th northeast regional stock assessment workshop (50th SAW) assessment report. Technical Report 10-17, US Dept Commerce, Northeast Fisheries Science Center, 2010.

[48] SR Jenkins, BD Beukers-Stewart, and AR Brand. Impact of scallop dredging on benthic megafauna: a comparison of damage levels in captured and non-captured organisms. *Marine Ecology Progress Series*, 215:297–301, 2001.

[49] GE Rosenkranz, SM Gallager, RW Shepard, and M. Blakeslee. Development of a high-speed, megapixel benthic imaging system for coastal fisheries research in alaska. *Fisheries Research*, 92(2):340–344, 2008.

[50] SM Gallager, H. Singh, S. Tiwari, J. Howland, P. Rago, W. Overholtz, R. Taylor, and N. Vine. High resolution underwater imaging and image processing for identifying essential fish habitat. In D.A. Somerton and C.T. Glentdill, editors, *Report of the National Marine Fisheries Service Workshop on Underwater Video analysis*, NOAA Technical Memorandum NMFS-F/SPO-68, pages 44–54. 2005.

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

[52] L. Oremland, D. Hart, L. Jacobson, S. Gallager, A. York, R. Taylor, and N. Vine. Sea scallop surveys in the 21st century: Could advanced optical technologies ultimately replace the dredge-based survey? Presentation made to the NOAA Office of Science and Technology, October 2008.

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

[54] 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.

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

[56] 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.

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

[58] 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.

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

[60] 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.

[61] 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.

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

[63] E. Alpaydin. *Introduction to machine learning*. MIT press, 2014.

[64] A. Krizhevsky, I. Sutskever, and GE Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[65] PM Roth and M. Winter. Survey of appearance-based methods for object recognition. Technical Report ICG-TR-01, Institute for Computer Graphics and Vision, Graz University of Technology, Austria, 2008.

[66] RJ Campbell and PJ Flynn. A survey of free-form object representation and recognition techniques. *Computer Vision and Image Understanding*, 81(2):166–210, 2001.

[67] S. Belongie, J. Malik, and J. Puzicha. Shape matching and object recognition using shape contexts. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(4):509–522, 2002.

[68] DG Lowe. Object recognition from local scale-invariant features. In *Proceedings of the seventh IEEE international conference on Computer vision*, volume 2, pages 1150–1157, 1999.

[69] H. Bay, A. Ess, T. Tuytelaars, and LV Gool. Speeded-up robust features (SURF). *Computer Vision and Image Understanding*, 110(3):346–359, 2008.

[70] A. Oliva and A. Torralba. Building the gist of a scene: The role of global image features in recognition. *Progress in brain research*, 155:23–36, 2006.

[71] S. Chen, Y. Li, and NM Kwok. Active vision in robotic systems: A survey of recent developments. *The International Journal of Robotics Research*, 30(11):1343–1377, 2011.

[72] SD Roy, S. Chaudhury, and S. Banerjee. Active recognition through next view planning: a survey. *Pattern Recognition*, 37(3):429–446, 2004.

[73] E. Dunn, JVD Berg, and JM Frahm. Developing visual sensing strategies through next best view planning. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 4001–4008, 2009.

[74] S. Frintrop. *VOCUS: A visual attention system for object detection and goal-directed search*, volume 3899. Springer, 2006.

[75] D. Park, D. Ramanan, and C. Fowlkes. Multiresolution models for object detection. In *European conference on computer vision*, pages 241–254. Springer, 2010.

[76] Y. Boykov and V. Kolmogorov. An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26:359–374, 2001.

[77] M. Meng, L. Gorelick, O. Veksler, and Y. Boykov. Grabcut in one cut. In *International Conference on Computer Vision*, December 2013.

[78] GoPro Hero 4. http://cbcdn1.gp-static.com/uploads/product_manual/file/256/UM_H4Silver_ENG_REVA_WEB.pdf. Accessed: 2016-10-06.

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

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

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

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

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

[84] Mission Oriented Operating Suite (MOOS). http://www.robots.ox.ac.uk/~mobile/MOOS/wiki/pmwiki.php/Main/HomePage. Accessed: 2016-10-03.

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

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

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

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

Acronyms

AC Alternating Current. 100

API Application Program Interface. 94, 95

AR Augmented Reality. 102

ARL Army Research Lab. vi

AUV Autonomous Underwater Vehicle. 7, 9, 27, 28, 34, 93, 94, 105

BUVA Bottom-Up Visual Attention. 28, 32, 38, 71

CAD Computer Aided Design. 97, 99

CCD Charge-Coupled Device. 7

CSHEL Coastal Sediment, Hydrodynamics, and Engineering Laboratory. v

DC Direct Current. 100

DOF Degree of Freedom. 97, 101, 102

DVL Doppler Velocity Log. 34

DYI Do-It-Yourself. 94

EMD Earth Mover's Distance. 53

FPS Frames Per Second. 101

HOG Histogram of Gradients. 49, 51–54, 58

HOV Human Operated Vehicle. 93

HSI Hue Saturation Intensity. 77

IMU Inertial Measurement Unit. 97, 98, 101–103, 106

INS Inertial Navigation System. 34

LiPo Lithium Polymer. 97, 100–102

MOOS Mission Oriented Operating Suite. 95

NIH National Institute of Health. vi

NSF National Science Foundation. vi

RDL Robotic Discovery Laboratories. v

ROS Robot Operating System. v, 95, 96, 98, 102, 103, 106

ROV Remotely Operated Vehicle. vi, xv, 13, 93, 94, 96, 100, 101, 103, 104, 106

RSA Research Set-Aside. 22, 34, 38

RST Rotation, Scaling and Translation. 12, 15, 19

SIFT Scale-invariant Feature Transform. 61

SURF Speeded-up Robust Features. 61

SVM Support Vector Machine. 53

TCP Transmission Control Protocol. 98, 102

TDVA Top-Down Visual Attention. 23, 32, 38, 42

WCTM Weighted Correlation Template Matching. 49, 51, 53–55, 58