University Transportation Research Center - Region 2

Final Report

Portable Bridge Scour Monitoring using Autonomous Underwater Vehicles Technology Development and Risk Assessment-based Platform for Deployment Prioritization

Performing Organization: Manhattan College

September 2018

Sponsor:
University Transportation Research Center - Region 2

Project No(s):
UTRC/RF Grant No: 49198-45-28

Project Date: September 2018

Project Title: Mobile Bridge Scour Monitoring Using Autonomous Underwater Vehicle

Project's Website:
http://www.utrc2.org/research/projects/mobile-bridge-scour-monitoring

Principal Investigator(s):
Mehdi Omidvar, Ph.D.
Assistant Professor
Department of Civil & Environmental Engineering
Manhattan College
4513 Manhattan College Parkway
Riverdale, NY 10471
Tel: (718) 862-8144
Email: momidvar01@manhattan.edu

Brent Horine, Ph.D.
Manhattan College
4513 Manhattan College Parkway
Riverdale, NY 10471
Email: horine@manhattan.edu

Performing Organization(s):
Manhattan College

Sponsor(s):
University Transportation Research Center (UTRC)

1. Report No.	2.Government Accession No.	3. Recipient's Catalog No.
4. Title and Subtitle Portable Bridge Scour Monitoring using Autonomous Underwater Vehicles: Technology Development and Risk Assessment-based Platform for Deployment Prioritization		5. Report Date September 2018 6. Performing Organization Code
7. Author(s) Mehdi Omidvar, Ph.D., Brent Horine, Ph.D.		8. Performing Organization Report No.
9. Performing Organization Name and Address Manhattan College 4513 Manhattan College Parkway Riverdale, NY 10471		10. Work Unit No. 11. Contract or Grant No. 49198-45-28
12. Sponsoring Agency Name and Address University Transportation Research Center Marshak Hall - Science Building, Suite 910 The City College of New York 138th Street & Convent Avenue, New York, NY 10031		13. Type of Report and Period Covered final, Sept. 1, 2016 - September 17, 2018 14.
15. Supplementary Notes		

16. Abstract

In this study, advances were made toward implementation of a geographic information system (GIS)-based scour risk assessment program facilitated by autonomous underwater vehicle (AUV) missions. The contributions of the work included:

(1) Adaptation of an existing AUV for bridge scour monitoring and inspection, including hardware upgrades, such as improvements on fabrication methods, upgrades on motor components for navigation in riverine environments, navigation, onboard processors, and instrumentation to accommodate collection of bathymetric data from a bridge site.

(2) Development of codes and algorithms to guide navigation of the AUV, and to process typical digital images collected in AUV missions at bridge pier sites. Specifically, codes were developed to analyze typical acoustic images of bridge piers for extraction of features of interest to scour monitoring, including the bridge pier structure and the riverbed outline. Methods developed included preprocessing using the Savitsky-Golay filter, entropy and range filtering, edge detection using the Prewitt operator, the Gabor filter, and k-means clustering, and the Hough transform. The algorithms were implemented in Matlab and OpenCV. Details of the image processing algorithms and results applied to a set of sample acoustic images were presented.

(3) Simulation of AUV path finding and navigation using the state-of-the-art Mission Oriented Operating Suite (MOOS) simulation environment, which is a multi-objective optimization system based on interval programming. A navigation algorithm was programmed to allow for typical AUV scour monitoring missions.

(4) A GIS-based platform to prioritize bridge scour monitoring and inspection programs. The HYRISK model was implemented by computing probability of failure and cost of failure for over 10,200 bridges in New York State for which scour susceptibility was applicable. Data from the National Bridge Inventory (NBI) were used to compile a database of bridge parameters relevant to the HYRISK model. The data were compiled in a GIS map of New York State. A set of risk maps were generated to demonstrate the efficacy of the developed model in visualizing risk distribution throughout the state, which can be useful in decision-making and planning for post-storm prioritization of AUV deployment for scour assessment and other mitigative actions.

17. Key Words Autonomous Underwater Vehicle; Risk Assessment; HYRISK; Image Processing; Bridge Scour.	18. Distribution Statement		
19. Security Classif (of this report) Unclassified	20. Security Classif. (of this page) Unclassified	21. No of Pages 73	22. Price

Form DOT F 1700.7 (8-69)

Table of Contents

List of Tables

List of Figures

EXECUTIVE SUMMARY

Infrastructure monitoring, inspection, and management is essential in ensuring the aging infrastructure is in a state of good repair, and is one the nation's top priorities. Maintaining expected performance levels for various infrastructure elements has become increasingly challenging in recent years, due to a number of factors, two of which include limited availability of funding, and increased occurrence and severity of extreme weather events. In particular, over 21,000 bridges over water have been deemed scour critical in the United States, with the numbers expected to grow. Faced with limited funding, states need to prioritize bridge scour monitoring, inspection, and mitigation programs, particularly in preparation for, and in the wake of, extreme events.

In this study, a cost-effective methodology was presented to conduct bridge scour assessment using autonomous underwater vehicles (AUV). There are several advantages to using AUVs in bridge scour monitoring, including (1) the proposed AUV is designed to be ruggedized and cost-effective, allowing states to deploy multiple units under unfavorable conditions, such as those experienced during a flooding event; (2) AUVs are portable units, and can be re-deployed at different locations; (3) they are able to produce bathymetric data from the entire channel, in addition to pier, contraction, and abutment scour hole depth; (4) they are equipped with optical cameras, which can be used to replace or supplement diving activities for bridge pier health monitoring, in addition to scour assessment; (5) AUVs can be equipped with multiple other sensors, allowing them to sample the stream bed, and to collect data regarding stream bed hydraulic properties, which is time-consuming and expensive to acquire by other means. As part of this study, an existing AUV was adapted for use in bridge scour monitoring. Software and hardware developments, as well as the required instrumentation were described. Image processing algorithms were developed to automatically segment and identify key components relevant to scour quantification at bridge piers following AUV missions. Simulations were conducted on autonomous navigation using a state-of-the-art simulation environment for AUVs.

The advances documented in this project support ongoing efforts to employ AUVs in bridge scour monitoring.

In order to implement the use of AUVs by local authorities in prioritized scour monitoring and inspection programs, a site-specific RISK assessment model was implemented, which facilitates the AUV deployment prioritization. Details of the risk assessment model, which is based on risk assessment methods developed by Stein et al. (1999; 2006), and uses National Bridge Inventory (NBI) data, were presented. The model was compiled on a geographic information system (GIS)-based platform to streamline the decision-making process, particularly in wake of extreme weather events.

1. BACKGROUND AND PROBLEM STATEMENT

Improving and sustaining the condition of the transportation infrastructure is one of USDOT's main strategic goals. An important step toward ensuring the infrastructure is in a state of good repair is to implement robust condition assessment and monitoring systems, so that updated information is available on the current status of the various infrastructure and transportation elements. In recent years, there has been an increase in occurrence and severity of extreme weather events. Flooding associated with heavy storms and hurricanes has caused severe damage to the already strained, and aging, existing infrastructure. In particular, flooding and flow surge in rivers and streams pose a serious risk of scour failure to bridges. Climate change has exacerbated hazards from flooding and extreme events. Ensuring that bridges are safe against extreme events requires continuous monitoring and inspection efforts, often beyond depleted state budgets. Prioritization of scour monitoring and countermeasure installation, particularly in wake of extreme events, remains a top priority toward maintaining the functionality of the transportation infrastructure.

Scour is the primary cause of bridge failure in the United States (Hunt 2009). Over 21,000 bridges have been determined scour critical in the U.S. (Gee 2008). Scour countermeasures are installed for bridges where scour poses a serious risk to structural safety. However, installation of scour countermeasures is not a permanent solution to the scour problem. The best practice in scour mitigation is, as recommended by FHWA HEC-18, to use a combination of scour monitoring and countermeasure installation (Melville and Coleman 2000; Hunt 2009). Bridge scour monitoring, therefore, constitutes an important component of a scour mitigation program, to ensure safety and a state of good repair for bridges.

There is an increasing need for monitoring scour at other locations in the vicinity of a bridge, because (1) scour can occur around the pier (termed pier scour), in the river bed where the bridge piers are installed (contraction scour), or at the abutments (abutment scour); (2) scour countermeasures, including riprap armors, can fail due to various mechanisms, which necessitates monitoring of the scour countermeasure, in addition to the scour hole itself; (3)

Availability of scour data can greatly support scour research. According to a USGS memorandum (USGS 2003; after Hunt 2009), "scour monitoring projects can represent a significant opportunity to collect field data that can be used for scientific research, while meeting a fundamental need of many highway departments."

Autonomous underwater vehicles are portable devices capable of monitoring scour conditions in bridges. However, technological and practical restrictions have limited their use in bridge scour monitoring programs in the past. In this study, a framework to employ autonomous underwater vehicles (AUV) as a portable scour monitoring tool is presented. The premise of this research is that AUVs can be used for inspection, condition assessment, and health monitoring of bridges. In particular, cost-effective, ruggedized AUVs can be employed before, during, and after extreme weather events, such as floods, to prioritize scour monitoring and mitigation plans, thereby optimizing allocation of limited resources.

Deployment of AUVs for routine bridge scour inspection in general, and for emergency condition assessment following extreme events in particular, requires a decision-making algorithm, which factors in the risk associated with scour-induced bridge failure. In this study, an AUV was adapted for use in scour monitoring. Significant hardware and software modifications were made to allow for robust mapping of the streambed, as wells as for visual inspection of bridge pier conditions. A risk-based decision-making tool was also developed to facilitate prioritization of AUV deployment for monitoring and condition assessment before, during, and after extreme events.

2. LITERATURE REVIEW

Scour is the primary cause of bridge failure in the United States (Hunt 2009). Over 21,000 bridges have been determined as scour critical (Gee 2008), that is, the foundation of these bridges has been determined to be unstable for the calculated/observed scour condition. Experience with monitoring and mitigation of scour has revealed that the best practice in scour mitigation is to use a combination of scour monitoring and countermeasure installation.

Many scour monitoring systems have been developed, and successfully installed, in recent years (Prendergast and Gavin 2014). Both fixed and portable scour monitoring systems have been developed. In most cases, bridge scour monitoring involves placing fixed scour monitoring systems to monitor scour hole depth, predominantly in the vicinity of piers. Fixed instruments placed near bridge piers are often used to continuously monitor scour at bridges. Portable instruments, however, are a more cost-effective means of monitoring scour at bridges (Schall and Price 2004). They can be used to monitor scour along the streambed, and can be used in multiple bridges. The main drawbacks of a portable scour monitoring system include (Hunt 2009): (1) continuous monitoring is not available for portable monitors, and (2) access to the bridge is often restricted during a storm event.

There has been a recent surge of interest in design and construction of autonomous underwater vehicles (AUV) and remotely operated vehicles (ROV) (Antonelli 2014; Sørensen and Ludvigsen 2015). Rapid technological advances have made AUVs and ROVs more viable and competitive compared to existing methods of marine operations, including monitoring and sampling. Multiple commercial and industrial applications already benefit from AUVs, or have invested in their development for future use (*e.g.*, Gilmour et al. 2012). As a result of rapid advances in control, navigation, instrumentation, and processing power, AUVs have now become a viable means of underwater exploration. While AUVs have been used by the United States Geological Survey (USGS) to obtain bathymetric data, mainly in deep waters, AUVs with accurate thrusting and navigation systems can be used in riverine environments as well. Rapid reduction of manufacturing and operation costs due to technological advances have now made AUVs as viable tools for river and stream bed monitoring, for applications such as scour monitoring. It is expected that future developments in this field will accelerate these trends.

Bridge inspection and monitoring is crucial in ensuring the safety of the aging transportation infrastructure. There are currently over 484,500 bridges constructed over water in the United States. According to FHWA guidelines, bridges require inspections at least every two years. Limited resources at the state level have necessitated prioritization of scour monitoring and mitigation efforts in many states. Design of new bridges has been re-evaluated in recent years in order to incorporate risk and uncertainty in scour predictions (*e.g.*, LaGasse et al. 2013). Risk-based models have also been incorporated in the decision-making process of scour monitoring and mitigation efforts, and have been used as a means to optimize expenditure of increasingly limited resources to ensure bridge safety against scour failure. A risk assessment model was introduced by the FHWA to prioritize scour-vulnerable bridges in the U.S. (Stein et al. 1999; Stein and Sedmera 2006). The model, known as HYRISK, defines risk as a product of the probability of scour failure and the costs associated with failure. The model has been updated to incorporate a more refined definition of the costs of failure. More recently, Khelifa et al. (2013) presented modifications to the HYRISK model to update the cost of failure, and to incorporate loss of life in the cost analysis. In addition, the model by Khelifa et al. (2013) presents a framework for incorporating climate change scenarios in scour risk assessment. A simpler approach has been presented by Johnson and Whittington (2011) for risk assessment of stream instability at bridge sites, which defines risk as the product of vulnerability of the bridge to scour failure, and criticality of scour conditions at the bridge. Both the HYRISK model and the model presented by Johnson and Whittington were designed to draw data from the National Bridge Inventory (NBI), which is maintained by the FHWA, and includes data from bridge inspections at the state level. The aforementioned risk assessment tools can be employed at the state level to effectively prioritize limited resources to scour monitoring and mitigation.

Inspired by the progress achieved in self-driving cars, state-of-the-art AUV technologies allow for collection of data in the challenging underwater environment. Signal processing, scene generation, interpretation, and action (*i.e.,* navigation) algorithms can be developed to navigate in the underwater environment to precise and accurate locations and vessel attitudes, and to collect quantifiable three-dimensional image data, so that bridge scour holes can be assessed under conditions that are now not compatible with diving operations.

The sonar technology available to the commercial and consumer user has dramatically advanced in recent years. Transducer technology is capable of forward and side scanning. They also now achieve wider bandwidths, which facilitate a broader choice in frequency selection and even chirp spread-spectrum techniques. Meanwhile, the analog to digital boundary has moved in favor of the digital domain, enabling inexpensive, low power, but advanced digital signal processing. These advances result in a greater awareness of the underwater environment around a vessel. The acoustic channel is very challenging; it is noisy and cluttered. Furthermore, the water environment can have temperature and salinity gradients, which can refract the propagation of the acoustic wave. Because high frequency sound is greatly attenuated in water, most sonars operate in the tens to hundreds of kHz range. In contrast, radars typically operate in the tens of GHz range. The velocity of sound in water is slower than the speed of electromagnetic waves in free space, the wavelengths of sonar and radar are comparable. Unfortunately, it is not as practical to scale the aperture of the transducer as it is in radar applications. The size of the aperture relative to the wavelength determines to a significant degree the width of the beam of energy. This has a large impact on the resolving power of the imaging system. Minimum resolvable feature size is on the order of the wavelength of the beam, but also a strong function of the beam width. Since the bridge scour application can use short range imaging, image resolution can be improved by increasing the frequency. Some special purpose transducers can be developed at the MHz range with large relative apertures, but still convenient to mount on a small vessel. The development of such a system is an expensive custom process. Before embarking on this path, it is important to establish the feasibility of the autonomous navigation and obstacle avoidance system and develop the necessary signal processing, scene generation, and interpretation algorithms with existing hardware platforms. The present study aims to contribute to the state of the art in this field through introduction of new hardware and software improvements to accommodate scour hole monitoring using AUVs, as described in the next sections.

3. METHODOLOGY AND APPROACH

The study consisted of three parts. The first part, described in section four of this report, involved adapting an autonomous underwater vehicle (AUV) for bridge scour monitoring and inspection. The AUV adaptations consisted of hardware upgrades, including improvements on fabrication, upgrades on motor components for navigation in riverine environments, navigation, onboard processors, and instrumentation to accommodate collection of bathymetric data from a bridge site.

The second part of the study, summarized in section five of this report, included development of codes, algorithms, and simulations to guide navigation of the AUV, and to process typical digital images collected in AUV missions at bridge pier sites. This part of the study consisted of the following steps:

- Develop codes and algorithms to analyze typical acoustic images of bridge piers for extraction of features of interest to scour monitoring, including the bridge pier structure and the riverbed outline. Methods developed included preprocessing using the Savitsky-Golay filter and entropy and range filtering, edge detection algorithms using the Prewitt operator, the Gabor filter, and K-means clustering, and the use of Hough transform. The algorithms were implemented in Matlab and OpenCV. Details of the image processing algorithms, the underlying theory, and results applied to a set of sample acoustic images are presented.

- Simulation of AUV path finding and navigation using the state-of-the-art Mission Oriented Operating Suite (MOOS) simulation environment, which is a multi-objective optimization system based on interval programming. A navigation algorithm was programmed to allow for typical AUV scour monitoring missions.

Finally, the third part of the study, detailed in section six of this report, consisted of developing a geographic information system (GIS)-based platform to prioritize bridge scour monitoring and inspection programs using AUVs and other methods. A risk assessment model based on the HYRISK model by Stein et al. (2006) was implemented by computing probability of failure and cost of failure for over 10,200 bridges in New York state for which scour susceptibility was applicable. Risk of scour-related failure was defined as the product of the probability of failure and the cost of failure, along with several risk adjustment factors. Data from the National Bridge Inventory (NBI) was used to compile a database of bridge properties relevant to the HYRISK model. The model uses several NBI items to quantify the probability of failure for a given bridge using overtopping frequency and scour vulnerability interpretations from the NBI data. The NBI items used to compute the probability of failure included the functional class (NBI item 26), waterway adequacy (NBI item 71), scour vulnerability (NBI item 61) and substructure conditions (NBI item 60). Cost of failure was quantified for each bridge as the product of the rebuilding cost, the running cost, time loss cost, and the cost of life. All relevant data was compiled in a GIS map of New York State, using the software ArcGIS.

In the following sections, details of each of the aforementioned components of the study are described in further detail. Examples are presented in the case of the image processing algorithms, and risk maps and risk tables are presented for the GIS-based risk assessment model.

4. AUTONOMOUS UNDERWATER VEHICLE (AUV) DESIGN AND DEVELOPMENT

The Harbor AUV, currently in final development stages by DURO AUS, is a hybrid autonomous system that is being designed for operation in littoral or shallow coastal waterways and harbors such as those found in the NY Harbor and surrounding areas, with a focus on infrastructure inspection and environmental monitoring.

The main purpose of employing an AUV for scour monitoring in this study was to enable bridge inspection and condition assessment before, during, and after high flow and storm events. The AUV was designed to operate under adverse circumstances, such as those encountered during flooding. Another advantage of the AUV developed in this study for scour assessment is that it is rugged, yet cost-effective. The designers of the AUV developed the vehicle as a robust, yet cost-effective tool for underwater data acquisition. It can therefore serve as a valuable tool in scour assessment and monitoring programs in states with a large number of scour-critical bridges, such as New York.

The feasibility of using the technology described above was demonstrated. Key enabling technology was developed in the form of custom algorithms running on embedded computing hardware. Simulations were carried out in virtual environments to demonstrate the effectiveness of the developed algorithms.

The Harbor AUV combines the movement capabilities of a tethered remotely operated vehicle (ROV) with the freedom and flexibility of an AUV. Below are some technical details of the vehicle, shown in Figure 1.

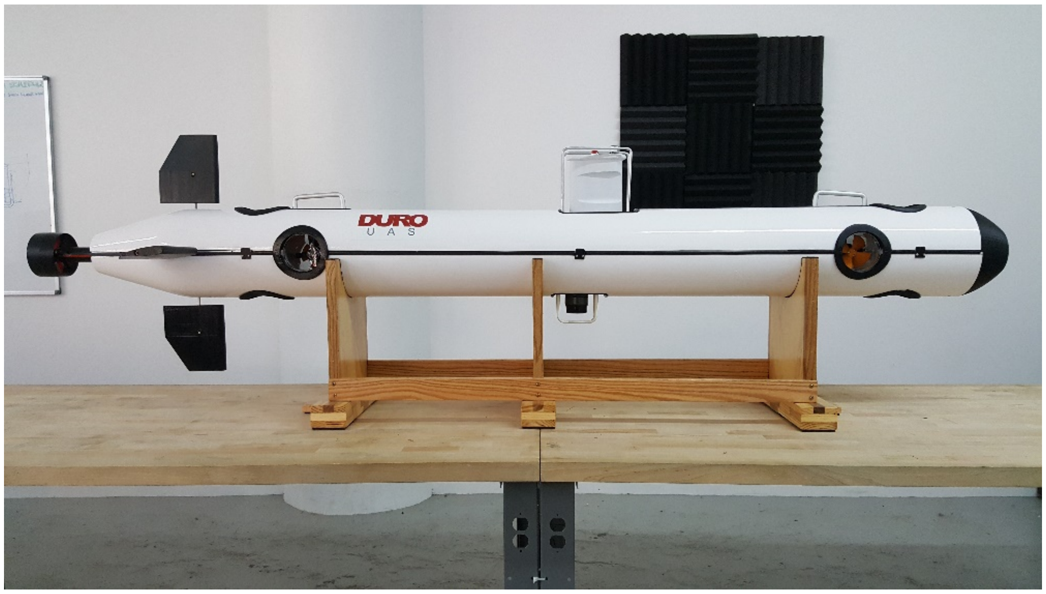


Figure 1. Side view of the Harbor AUV under development for scour monitoring of bridge piers.

Physical Design: The AUV was mostly fabricated using aluminum and composite materials, and consisted of five thrusters as well as four rudders in a torpedo-shaped design to maximize hydrodynamic properties. Over 85% of the vehicle was made in-house using additive manufacturing and advanced manufacturing techniques.

Motor Components: The Harbor AUV uses five thrusters in total to achieve over five degrees of freedom while in operation. The thrusters also act as correctional thrusters in heavy currents. Four of the thrusters (orange propellers shown in Figure 2) are custom-made while the main rear thruster, shown in Figure 3, was manufactured by Maxon Motors (MT 40). The horizontal and vertical thrusters were custom made and provide over 8 lb of thrust in both forward and reverse directions. The main thruster provided over 22 lb of thrust.

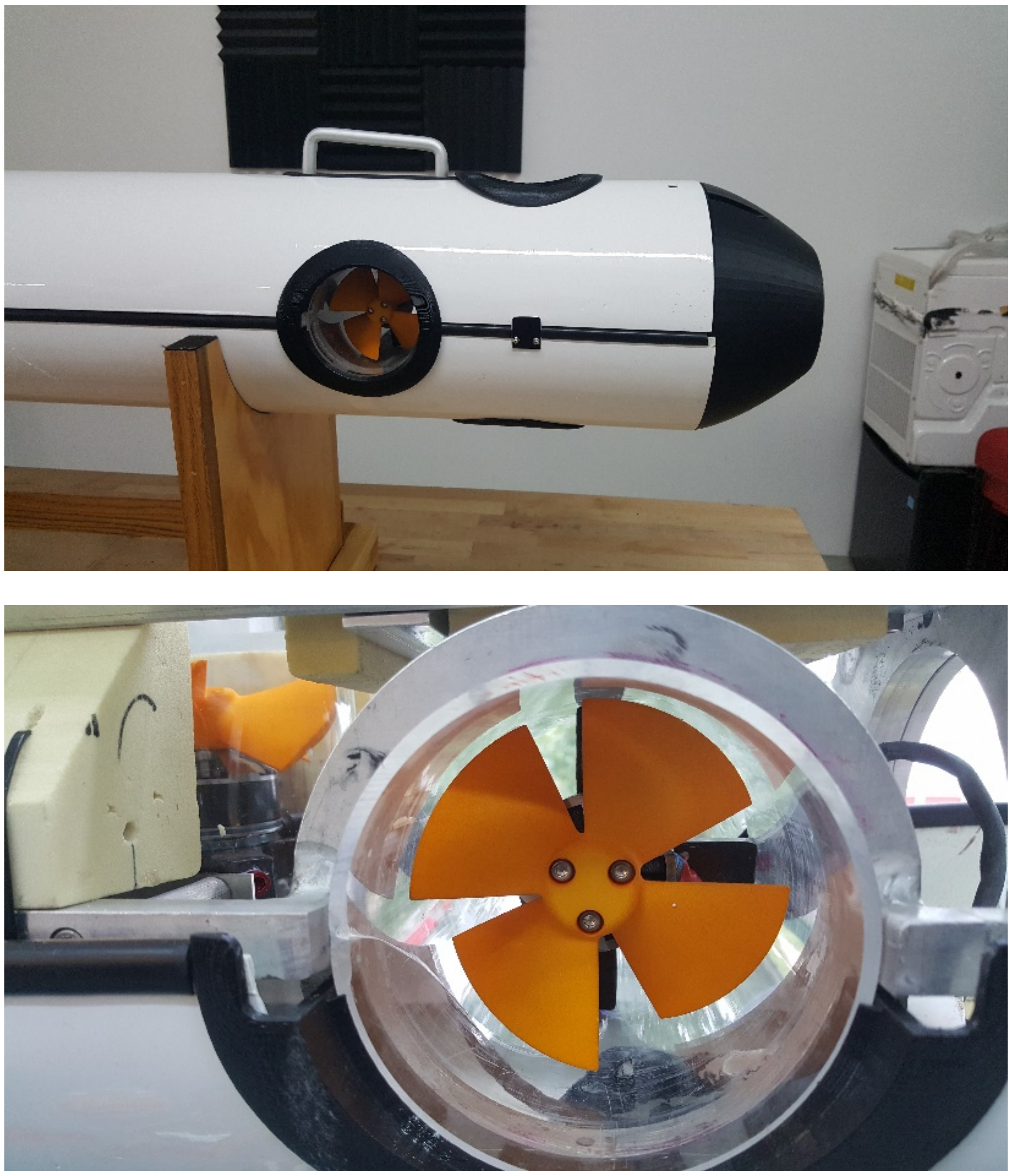


Figure 2. One of the five thrusters used in the Harbor AUV (top), and close-up of a thruster used for navigation and path correction.

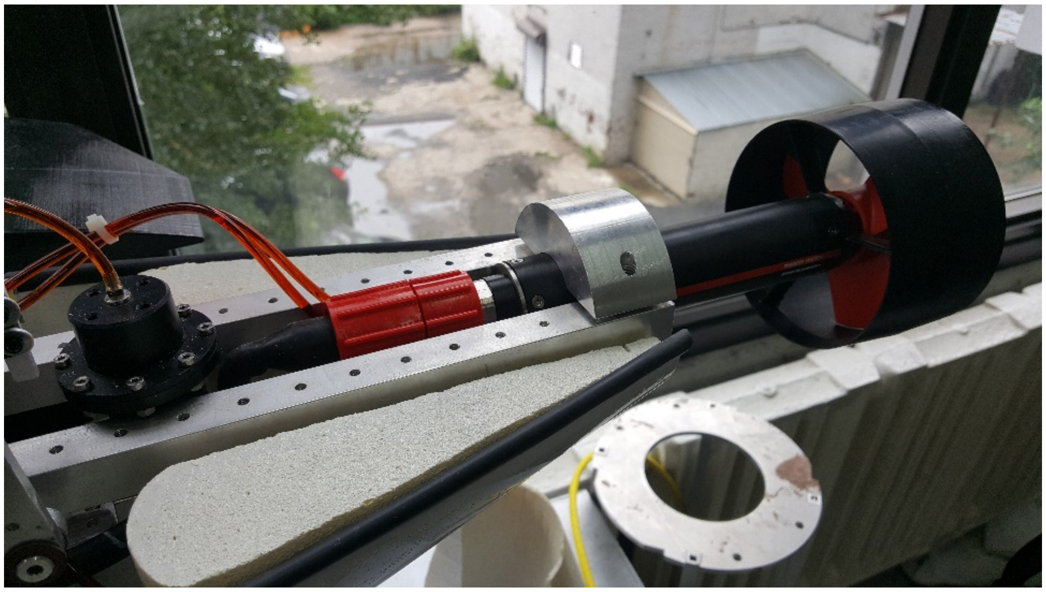


Figure 3. Main rear thruster used in the Harbor AUV.

Navigation Components: Navigation for the vehicle is still in the testing phase with developments underway for full autonomy. Navigation will be achieved using a variety of sensors to help localize the vehicle during missions. Some of the initial components developed during this study are as follows:

- **IMU** – DURO AUS is currently collaborating with X-IO Technologies to incorporate the NGIMU (Figure 4) onto the Harbor AUV. The IMU on-board sensors include a triple-axis gyroscope, accelerometer and magnetometer, as well as a barometric pressure sensor and humidity sensor. An on-board AHRS sensor fusion algorithm combines inertial and magnetic measurements to provide a drift-free measurement of orientation relative to the Earth. Each device is individually calibrated using robotic equipment to achieve the specified accuracy. Serial data sources, such as GPS modules, can be connected to the auxiliary serial interface.

- **Real-time communication** – This is achieved via USB, Wi-Fi, or serial/RS-232. Data may also be logged to an on-board micro SD card. The NGIMU uses the popular OSC communication protocol, which makes it immediately compatible with many software applications, and is straightforward to integrate with custom applications with libraries available for most programming languages.

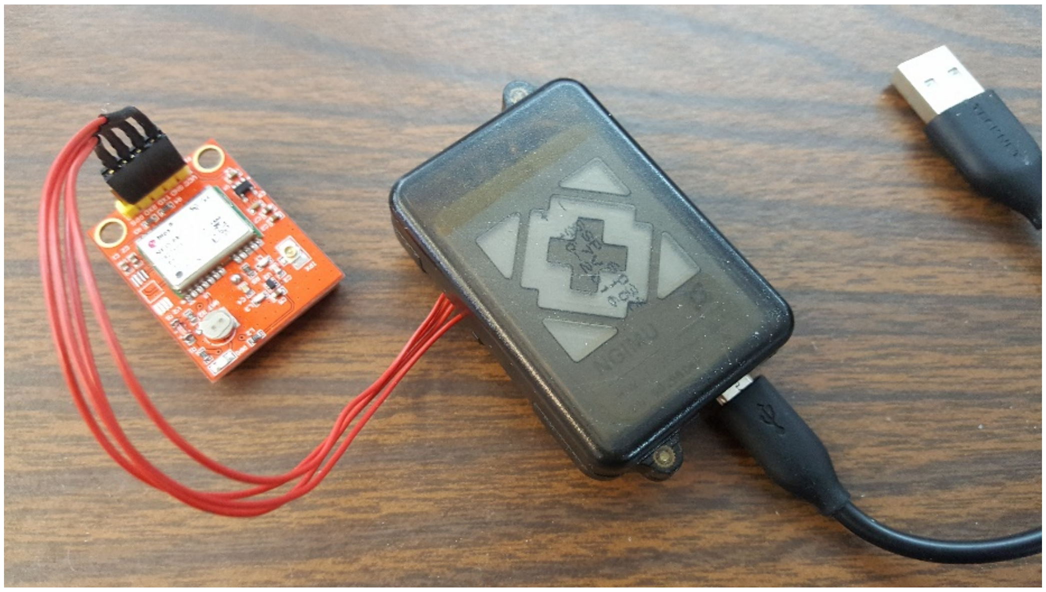


Figure 4. X-IO NGIMU with UBlox GPS module attached.

Onboard Processors: The Harbor AUV has 3 computers onboard, each dedicated to different tasks in the vehicle. Work is currently underway to reduce the processors by one unit to reduce power consumption and increase processing speeds. The processors include (1) **Jetson TX1:** This unit is the main computer used for image processing and control systems. This unit is responsible for the simultaneous localization and mapping (SLAM) system developed as part of this study, as well as for machine vision for pier and object detection, also developed in this study. The NVIDIA Jetson TX1 module embedded in the AUV is comprised of four ARM 64-bit Cortex-A57 CPU cores (max frequency of 1.91GHz) and four Cortex-A53 low-power cores, 4GB of LPDDR4 memory with 25.6GB/s of memory bandwidth, and has a 256-core Maxwell graphics processor capable of 1 TFLOP/s. (2, 3) **Raspberry Pi and Arduino:** The other two microcontrollers or processors are used for basic functions such as running the custom-built leak sensors, strobe, rudder controls and correctional thrusters, among other functions.

Instrumentation: While multiple sensors can be mounted on the AUV, the main instruments currently mounted include a sonar unit and an optical imaging unit. An Echologger MRS900

Scanning Sonar unit was mounted on the AUV as part of this study (Figure 5). Testing is currently under way in a controlled pool-testing environment. Though initial tests were promising, DURO AUS is currently investigating other scanning sonar units as well, including the Tritech Micron Sonar Scanner and the TriTech Starfish 453 side scan sonar unit as well.

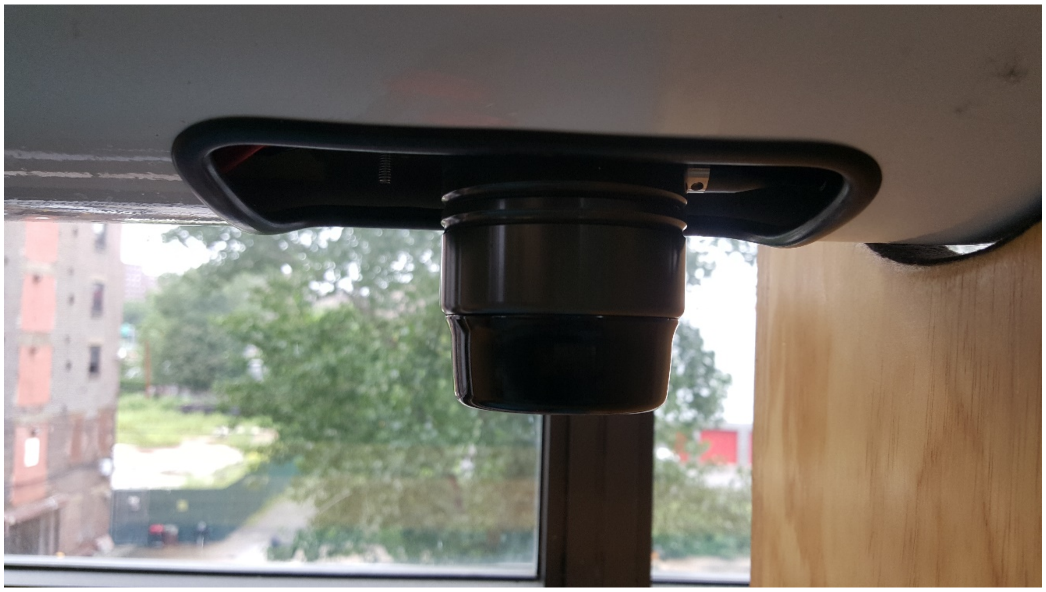


Figure 5. The AUV payload bay, showing the mounted end of the Echologger MRS900.

Communication: The main communication functions of the AUV are currently achieved in two ways. For initial testing of the prototype a tether is attached to the vehicle through a penetrator in the computer vessel that allows for operation and communication with the vehicle. The tethered control also allows for real time feedback on the AUV during initial tests. An Xbee RF Module (Figure 6) that operates at 2.4ghz is currently used for above ground communication. Pool testing of autonomous navigation and control is currently underway, but was not fully achieved during initial tests carried out over the duration of this study. The focus for Duro UAS was to develop a stable platform that could collect reliable, high resolution imagery using a variety of sonars and cameras. Work is currently underway for incorporating high-speed acoustic modems for surface vessels for AUV communication and data transfer.

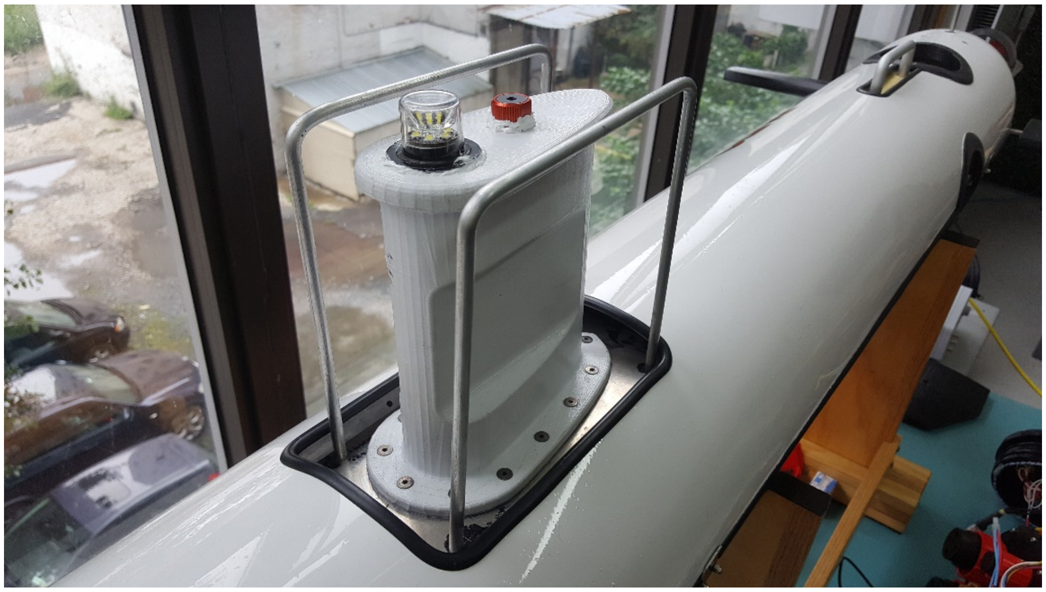


Figure 6. AUV Mast, which houses a pressure sensor, strobe, Xbee module, Dorji RF module and GPS antenna.

Setbacks: Numerous setbacks slowed initial progress in the project. The main setbacks included issues with thruster functionality and power distribution, as described below.

- **Thruster issues:** Initially, the thrusters used on the AUV (Blue Robotics) had technical issues related to their manufacturing process. Ultimately, in-house thrusters were designed and manufactured, as described in previous sections.

- **Power distribution:** Issues were encountered with the power system, resulting in certain systems turning off or powering down due to back electromagnetic frequency or spikes in voltage. This was solved by fabricating in-house power distribution boards, an example of which is shown in Figure 7.

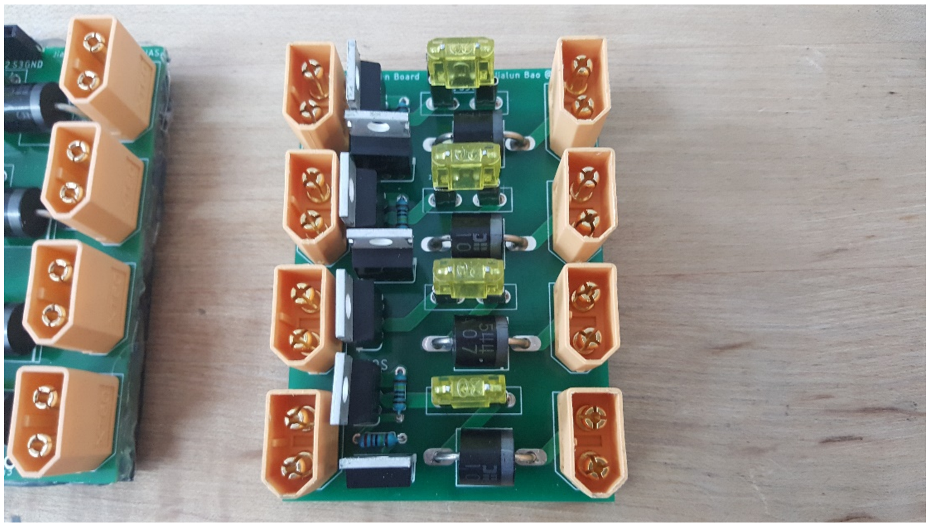


Figure 7. In-house power distribution board mounted on the AUV.

Future developments: The main developments underway for the AUV include integration of DVL into the system for more accurate autonomous navigation, achieving full autonomy, incorporating machine vision capabilities developed as part of this study into the onboard processors, further reducing power consumption, and optimization of vehicle power distribution, among others.

5. IMAGE PROCESSING ALGORITHMS FOR SCOUR IDENTIFICATION

Because of the poor lighting conditions and opaqueness of the water environment due to its turbidity, efforts were focused on sonar imaging. Sonar images are notoriously noisy and limited in resolution. This requires significant signal processing to extract reliable and measurable information. Robotics researchers have been advancing Simultaneous Location and Mapping (SLAM) technology. This study aimed to leverage SLAM research, particularly in navigation and occupancy mapping, which allowed the authors to focus on image processing of the sonar signal. While the AUV can be equipped with video imaging capabilities, the scope of the present work was limited to navigation technology and programming, as well as processing of sonar images. The developed image processing algorithms are presented in this section, and navigation simulation is described in the next section. Because this work was done before the vehicle could be equipped with the sonar and cruised in the water, it was based upon images produced by fixed imagers from commercially available sources and other existing literature. Six cases were considered, shown in Figure 8 and Figure 9.

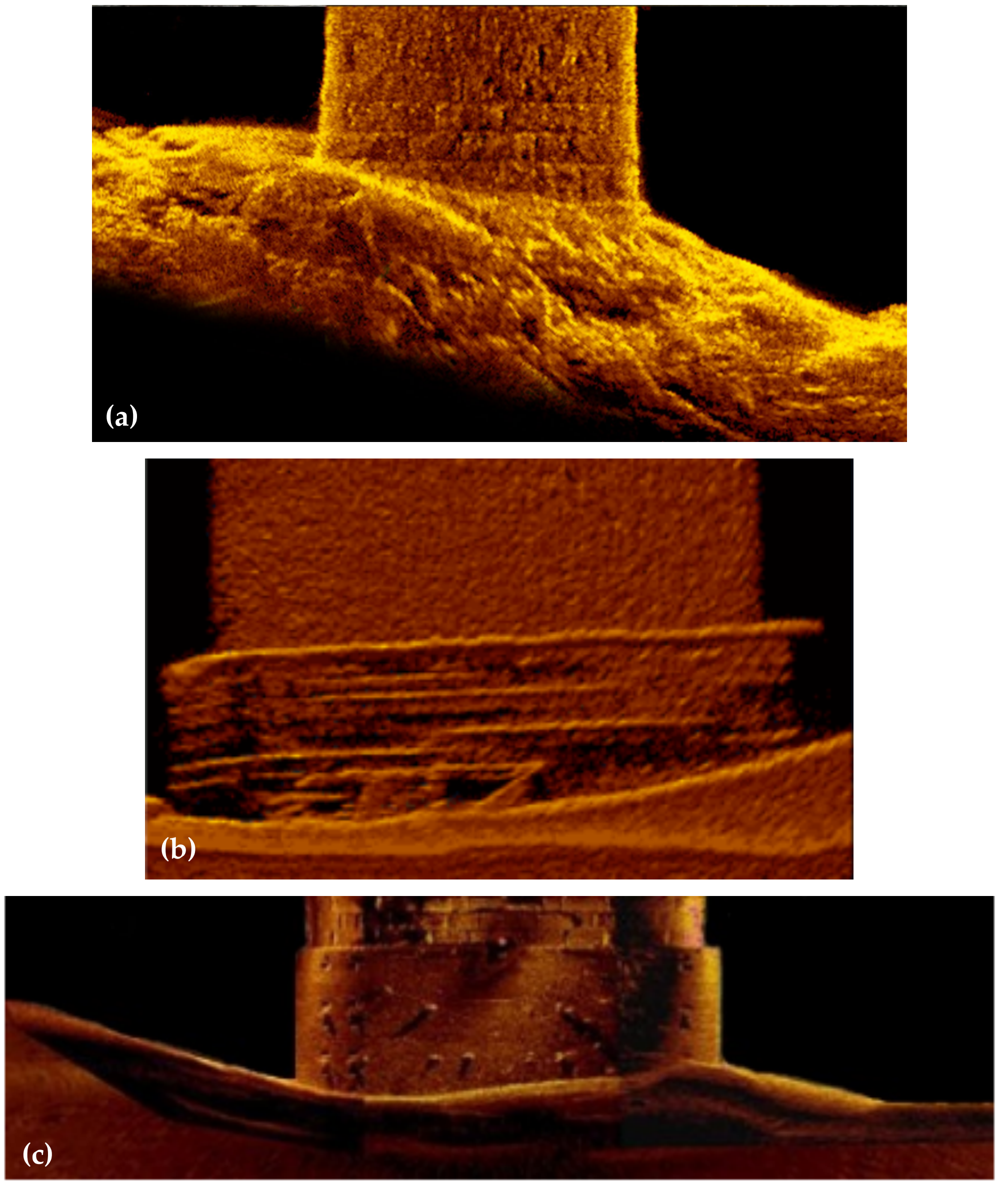


Figure 8. Acoustic images for cases one through three used to develop image processing algorithms in this study[1].

1 Source: (a) Sonar Image Finland; http://www.tcbusinessnews.com/wp-content/uploads/2015/06/Sonar.jpg; (b) New Sonar Technology; https://www.508mystery.com/single-post/2015/11/26/NEW-SONAR-TECHNOLOGY; (c) Underwater Bridge Inspection; http://www.prweb.com/releases/2014/05/prweb11841442.htm.

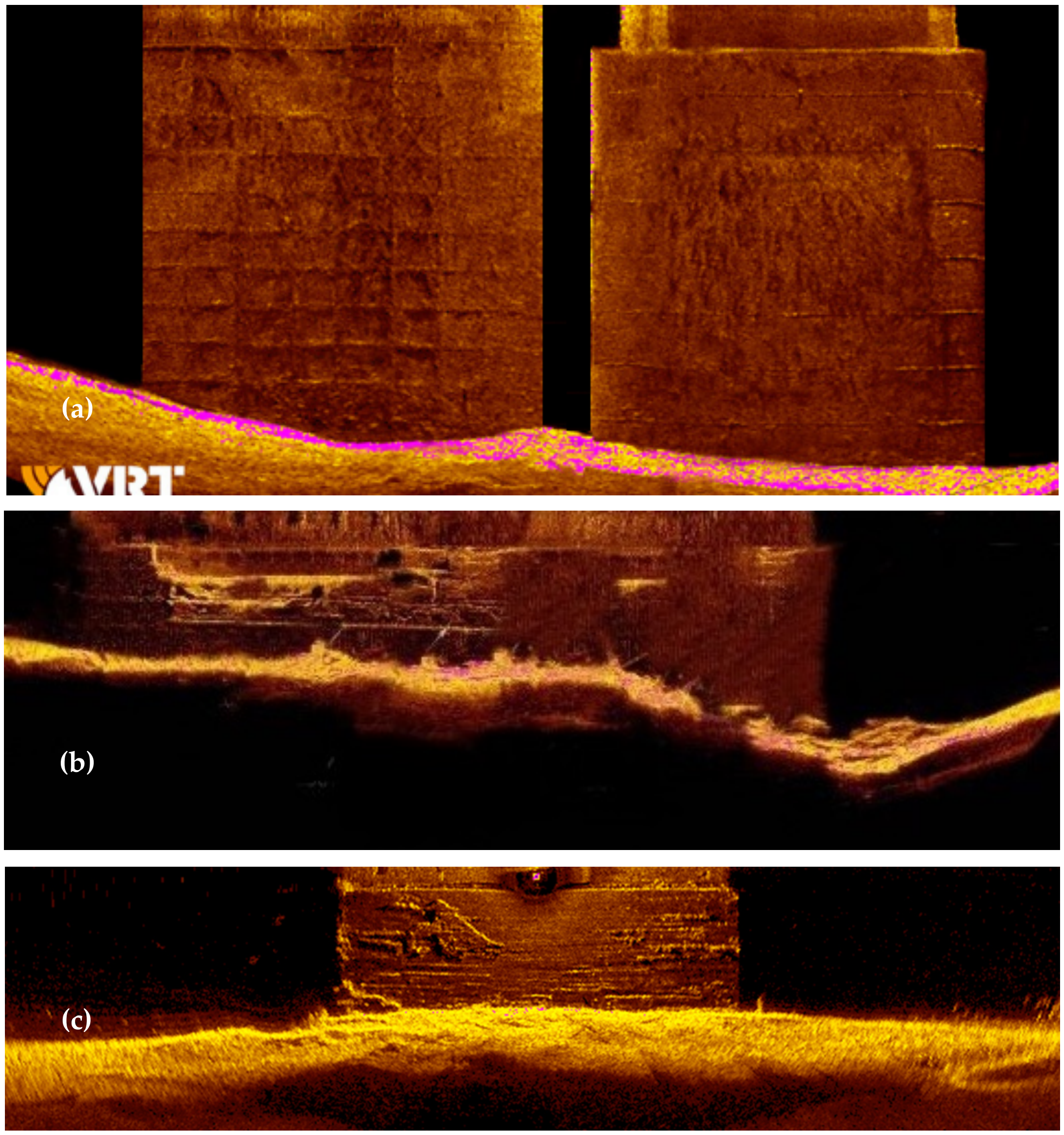


Figure 9. Acoustic images for cases four through six used to develop image processing algorithms in this study[2].

The image processing algorithm consisted of three steps:

2 Source: (a) Bridge in 3D; http://www.goodnewsfinland.com/wp-content/uploads/2015/08/VRT_bridge_3D.jpg; (b) Bridge Pier Elevation with Underwater Sonar Imaging ; http://www.pcs-civil.com/services/underwater-inspections/sonar-imaging-system; (c) Usługi - Badania obiektów i konstrukcji hydrotechnicznych; http://www.escort.com.pl/badania-obiektow-i-budowli-hydrotechnicznych

(1) In the first step of the algorithm, sonar images were filtered and the bridge pier features of interest, that is, the bridge pier structure and the river bed, were extracted.
(2) In the second step, texture-based segmentation was used to separate the bridge pier and riverbed from the background.
(3) Finally, in the third step the outline of the riverbed was analyzed and classified for tagging into the GIS database following typical AUV missions.

These steps were implemented in Matlab and were tested on the sample images shown in Figure 8 and Figure 9. After the algorithm was established and characterized, key parts were converted to OpenCV in C++ to run in real-time on a Jetson TX1 embedded processor mounted in the AUV. The three parts of the image processing algorithm are described next.

5.1. Image processing algorithm step I: bridge feature extraction

In the first part of the algorithm, the vertical outline of the bridge pier was identified. The image processing components used are shown in Figure 10. The key techniques used in this process were Savitzky-Golay Filtering (Schaffer 2011; Gander and Matt 1997) and statistical analyses for preprocessing, and finally edge detection and the Hough Transform (Duda and Hart 1972; Hart 2009) to extract the aforementioned features. In the following sections, the theoretical basis for each of the components is presented. Results of the image processing applied to the six images shown in Figure 8 and Figure 9 are presented in Section 5.4.

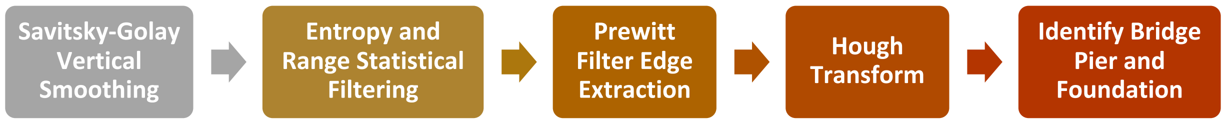

Savitzky-Golay Vertical Smoothing
Entropy and Range Statistical Filtering
Prewitt Filter Edge Extraction
Hough Transform
Identify Bridge Pier and Foundation

Figure 10. Step 1 of the algorithm – bridge feature extraction.

5.1.1. Savitzky-Golay filtering: data reduction and signal enhancement

The procedure to reduce or smooth the noise of a measured signal is commonly known as low pass filtering. High frequency components within the image are removed to allow for more gradual variations in the image, thus smoothing the image information. An array of smoothing techniques was considered including Gaussian Filtering, Moving Average filtering, and ensemble Filtering. The Savitzky-Golay Filter was recognized as the most suitable for the current input data. Savitzky-Golay smoothing filters are used to smooth out a noisy signal where the noiseless frequency span is large. With this kind of application, Savitzky-Golay smoothing performance is superior to standard averaging finite impulse response (FIR) filters (Shaffer 2011), which are prone to filtering out a large portion of the signal's high frequency content along with the noise. The Savitzky-Golay filter, like the moving average filter, calculates an average value of the neighboring data at every pixel. The Savitzky-Golay filter enriches the concept of a moving average by fitting a polynomial through the fixed number of points, as defined by the window, or frame length (Press et al. 1997).

Images were first converted to grayscale for processing. Next, the Savitzky-Golay filter was applied to the grayscale image. This can be done in the vertical, horizontal or both directions. Because the object of interest was the bridge pier, the filter was applied only in the vertical direction to smooth imperfections in the boundary pixels in the original edges, thus eliminating horizontal noise and enhancing vertical edges. The Savitzky-Golay filter smooths the image much like a moving average filter, where a polynomial of a certain order is fit to the data within a window size. Thus, the order and window size are unique to the application. For all of the images considered, a window size of approximately 15% of the image was experimentally found to be sufficient. A third order smoothing fit was implemented because it had a greater impact on the smoothing.

5.1.2. Entropy and range: statistical and morphological preprocessing

Entropy is a method to measure the randomness of intensity distribution in a digital image, often to designate between areas of uniform and irregular intensities (Jeynes 1957). The method innately lends itself well to edge detection and texture analysis. Entropy was used to separate the foreground from the background and then, range filtering was applied to extract a preliminary bridge boundary. Entropy measures the difference amongst pixels through a measure of randomness in a local neighborhood, which was set to nine pixels in this study. Range locally characterizes the range of pixel values according to a local neighborhood, which was set to a finer three-pixel neighborhood. This is unique in its use of wider to finer area of applying the statistical filter. A wavelet approach was considered as an alternative to these statistical measures but was deferred to future work.

Shannon Entropy was implemented for the bridge pier images, which is the entropy metric implemented in Matlab. Entropy and range preprocessing were used dually in this algorithm: first, to clean the image for edge detection and secondly, to segment the image via texture analysis. Entropy alone is commonly used to preprocess and define thresholds in these domains; however, the combined use of entropy and range as a preprocessing technique was original to this algorithm.

Statistics

Statistical methods play a key role in feature extraction and identification techniques in image processing. To apply these techniques successfully, the data must be formatted as a statistical measure. This was achieved through gray-level Histogramming. From the histogram, a quantification of higher-order statistical properties can be derived to characterize texture (Materka and Strzlecki 1998).

The entropy is a measure of histogram uniformity through the energy, or information, of the image. Thus, in image analysis, the intensity information is measured as Shannon's entropy where, instead of applying an unknown probability distribution, the normalized histogram function $p(i)$ is used. The entropy measures evaluated allow for the discernment of texture patterns which are based on their entropy values. They can also be used for edge detection, thresholding, and as discussed, texture analysis. Entropy values of the image are generally evaluated globally, but, by using Entropy filtering, a measure of entropy within regions of the image can be found.

Entropy Filtering

Entropy filtering requires the substitution of pixel values in the image by values of entropy. Entropy computations are made in the user-specified area, in the pixel-neighborhood of the input image. For small neighborhoods the local disturbances will be given weight, and the output image will be too noisy. Conversely, an excessively large value will not facilitate the perseveration of details, and the output image will be blurred. Neighborhood selection is a trade-off between a noisy and blurry image.

Other possible features derived from the histogram are the minimum, the maximum, median value, and the range. The latter was used in conjunction with entropy to act as a combined filter for image preprocessing in both edge detection and texture analysis, which is used in Step 2 of the Algorithm. There is an analogous relationship between range and range filtering. Here, range gives insight into the statistical dispersion of the data. Range mainly depends on the neighborhood size of observations, as it is used in representing the dispersion of small datasets. The main effect achieved by the range is to amplify the effect of entropy filtering onto the image as a two-stage preprocessing technique (Flemming 2009). Although entropy filtration strongly relies on the size of the neighborhood considered and can visually produce a noisy or blurry output, it was found to be the most effective for this study.

Entropy in edge detection

Edge detection is the process of finding sharp discontinuities in an image. The discontinuities are from various scene features, for example: changes in material properties, variations in scene illumination, discontinuities in depth, and discontinuities in surface orientation. In this study a combined entropy and range filtering was applied as a preprocessing technique to amplify local edges and textures for a Prewitt edge detector, discussed below. Because information from independent events can be additively decomposed, the probability distributions of the background, $p_{\,b}(i)$ and object $p_o(i)$ can be partitioned from the normalized histogram $p(i)$ (El-Sayed 2011).

The relationships between $p_o(i)$, $p_{\,b}(i)$ and $p(i)$ are shown in Eq. 1-4 (El-Sayed 2011). The change in the limits of the summation are the main differences between *P_o* the histogram of the object and *P_B*, the histogram of the background:

$$\boldsymbol{p_o(i): \frac{p_1}{P_o}, \frac{p_2}{P_o}, \dots \frac{p_t}{P_o}} \tag{1}$$

$$\boldsymbol{p_b(i): \frac{p_{t+1}}{P_B}, \frac{p_{t+2}}{P_B}, \dots \frac{p_k}{P_B}} \tag{2}$$

$$\boldsymbol{P_O = -\sum_{i=1}^{t} p(i)} \tag{3}$$

$$\boldsymbol{P_B = -\sum_{i=t+1}^{k} p(i)} \tag{4}$$

where *t* is the threshold value. The entropy of pixels and the entropy of the pixels defining the object of interest and the background, respectively, are expressed in order, as follows:

$$\boldsymbol{H = -\sum_{i=0}^{G-1} p(i) \log_2 p(i)} \tag{5}$$

$$\boldsymbol{H^o = -\sum_{i=1}^{t} p_i(i) \log_2 p_i(i)} \tag{6}$$

$$\boldsymbol{H^B = -\sum_{i=t+1}^{k} p_i(i) \log_2 p_i(i)} \tag{7}$$

Entropy is therefore dependent upon the threshold value (*t*) for the background and the object, and is calculated as the sum of all entropies. Maximizing the amount of information between the two classes (object and background) is the goal (Leung 1996). With entropy maximized, the luminance level, *t*, which maximizes the function, is known as the optimum threshold value:

$$\boldsymbol{t^{opt} = \max\,[H^o(t) + H^B(t)]} \tag{8}$$

Instead of using entropy to customize thresholding values of the edge detector based on the input, applying entropy and range as a preprocessing technique to a static threshold in the edge detector was done. In deployment, updating the threshold based on the detected entropy and image noise would be of use to improve the accuracy of the edge detector, scalability of the algorithm and improve robustness to sources of noise.

Implementation of entropy and range

Matlab's entropy function can be used to calculate the entropy of a grayscale image. The Matlab entropy function outputs a scalar value for the global entropy of the image. As a statistical measure of randomness, entropy here is used to represent the input image texture. Matlab's **entropyfilt** function was used to retain the dimensionality of the image by allowing for entropy to be applied as a mask. Around each pixel in the image, the **entropyfilt** function returns the entropy of a 9x9 default neighborhood. The formula for the entropy calculation in each neighborhood is: -sum(*p*.*log2(*p*)), where the value *p* is histogram counts from the Matlab histogramming function, **imhist**.

Preprocessing of images, such as the application of light normalization, is essential when using thresholds, particularly when objects are not well separable from the background. The output image computed by entropy filtration is highly dependent upon the area selected. For small neighborhoods, the local disturbances will result in weights that are sufficient enough to render the output image as too noisy. Conversely, an excessively large neighborhood value will not retain details rendering the output image as blurred. Thus, the main issue to be considered in the filtration method is the selection of a neighborhood. The compromise between a noisy or blurry image is a common tradeoff for value selection. Matlab's floor function rounds the output matrix elements to the nearest integer that is less than or equal to it. Entropy filtering floors the image to half the neighborhood's size. Matlab's **strel** (structural element class) can be used to define neighborhoods of many different shapes. In this case, a **strel** object is created; the **getnhood** (get neighborhood) function is used to extract the neighborhood for that **strel** object. In contrast, the **rangefilt** function returns an array where each output pixel contains the range value, which is the maximum value – minimum value of a 3-by-3 neighborhood around the corresponding pixel in the input image. Like entropy filtering, the input can be of any dimension, but grayscale was chosen for this study. Also, like **entropyfilt**, **rangefilt** defines the center element of the neighborhood as the position at the midpoint of the neighborhood's width (in both dimensions).

Matlab's **rangefilt** function uses **imdilate** and **imerode** (morphological functions) to find the maximum and minimum values in a specified neighborhood. In this study, this was achieved by implementing the padding behavior of these morphological functions to calculate the range through a **strel** object. The neighborhood of values is extracted from the structuring element object's neighborhood property, similar to **entropyfilt**.

5.1.3. Prewitt Operator

After Entropy and Range preprocessing, the edge detection filter was applied. A variety of edge detection filters were surveyed to extract the bridge boundaries. These included the Sobel, Canny, Roberts, Laplacian of Gaussian, and Prewitt filters. In general, edge detection filters detect edges using first derivatives, through a gradient calculation, or using second derivate, through Gaussian or zero-crossing detection methods. In the Laplacian, Laplacian of Gaussian (LOG), and Gradient, edge detection is based on the derivative of the pixels of the original image (Sunaga 2009). The Prewitt, Sobel and Roberts operators are gradient-based edge detection methods. They employ two dimensional (2-D) linear filters to detect and process vertical edges, as well as horizontal edges separately. This facilitates the approximation of first-order derivatives of the pixel values of the subject image. In addition to being more computationally complex, Gaussian methods such as the Laplacian of Gaussians and Difference of Gaussians exhibit heightened noise sensitivity (Dim and Takamura 2013). The Canny method integrates elements of both types by implementing a Gaussian filter to smooth noise and then a first order derivative to detect edges.

The lack of feasible directional filtering was the main reason the canny filter was not chosen for this study. Although a direction can be applied to the gradient, the Gaussian smoothing phase does not retain directional resolution and separation, and it is more computationally complex than implementing a one-dimensional gradient filter such as the Prewitt (Prewitt 1970). On the other hand, the LoG method filters were dismissed due to their extremely low sensitivity. These edge detectors are in the second derivative family and look for more intense changes, that is, stronger edges, in the image through a second derivative calculation than simple first order classical gradient filters, like Sobel and Prewitt. Hence, simple gradient filters like the Sobel and

Prewitt were selected in this study. It was found that filtering with respect to the vertical direction was the most important design criterion for this phase of the study. Sobel and Prewitt were tested in the vertical directions and were found to be more effective than the Canny in enhancing vertical edges. It was found that the Prewitt had a better sensitivity than the Sobel due to the lower values in its kernel. This could be an issue if there is much noise present in the image, but because horizontal noise is filtered out and vertical features are enhanced by Savitzky-Golay, and the Prewitt is preset to extract edges in the vertical direction, it is of more use to have a better sensitivity in this application. Because the main edges being detected in this study were large substantial signals, the Prewitt filter produces reasonably accurate results with lightweight computations. If detection of finer edges is needed, an edge detection filter in the Gaussian family could be chosen.

Several edge detection algorithms were tested to evaluate their applicability to bridge pier images. Each was tested with thresholds of 0.1, 0.3 and 0.8. The Prewitt and Sobel were set in the vertical direction. The Laplacian of Gaussian filter did not operate at these high thresholds, so a lower 0.05 threshold and a default selection were shown to characterize its behavior. The Hough Space description of the peaks derived from these filters was included to prove their effectiveness. The Prewitt with a vertical direction and threshold of 0.3 was ultimately used for this study.

5.1.4. Hough transform

The Hough transform is used for the identification of lines, circles, ellipses and other shapes in an image. This technique finds instances of objects categorized by a defined class-shape through a voting scheme in a parameter space (Hart 2009).

Acoustic images were fed into the **hough** Matlab function to detect the presence of straight line segments. This was done by finding the Standard Hough Transform (SHT) via the parametric representation of a line in Hough Space. The function returns ρ, the distance from the origin to the line along a vector perpendicular to the line, and θ, the angle in degrees between the x-axis and this vector. The angle θ was set within a threshold of 15 degrees on either side from 90 degrees to ensure that the lines of the image were vertical. The function can also return the

Standard Hough Transform, SHT. SHT is a parameter space matrix whose rows and columns correspond to ρ and θ values, respectively. This set of values was useful in practice as the input to detecting the peak Hough values, *P*. The **houghpeaks** function was used to extract the peak values of *H*. The peak *H* values represent the points where the curves intersect; *H* gives a distance and angle. This distance and angle describe the line which intersects the points being tested.

In this study, the maximum number of peaks was set to 8 as a default to account for the greatest amount of line segments present in the image, with a threshold of 30% of the maximum value of *H* to define the peak transform magnitudes. It was found that modifying this value greatly affected the results. It was essential to consider the size of the suppression neighborhood. The neighborhood around each peak that is set to zero after peak identification is called the *suppression neighborhood*. The default value of the suppression neighborhood is the maximum value of *H* divided by 50. Instances where this default lacked will be discussed in the results section. The algorithm was highly sensitive to changes in this value because it was the input to the **houghlines** function which ultimately yields the bridge width.

Plotting and bridge width calculations

The **houghlines** function accepts the Hough arguments, *H, T, R*, and *P*, discussed above. However, this function allows for error minimization in the Hough line plotting by accepting minimum length and gap margin arguments. Minimum length can ensure that the segments are likely to fall within a certain range. This is useful to eliminate noise and image artifacts that may have been amplified during preprocessing. A concern with setting a minimum length at this phase of development is that the pier can be a small segment within the full image of the pier and foundation. Nonetheless, the algorithm first identifies the greater (longer) structure and then determines its identity. If this length does not have any segments above it, then it is a foundation or a pier. If both are present, then, the algorithm considers both. Similarly, the fill gap capability of the **houghlines** function achieves error correcting by connecting small gaps between line segments that are likely to be the same structure.

Finally, it was found that the width calculation differed according to the type of image type detected. For instance, if the image contains both a bridge foundation and a pier, the algorithm requires a separate approach for extracting and plotting these values. There are two sets of widths and heights of interest in this case and must be distinguished through the distances and lengths of the Hough line segments. The same identification decision needs to be made when both extracting and plotting the geometric features of the bridge.

5.2. Image processing algorithm step II: riverbed extraction

The goal of this part of the algorithm was to extract the riverbed boundary from the rest of the image. This was done via texture segmentation primarily relying on entropy preprocessing, Gabor filtering, and k-means clustering. This part of the algorithm consisted mainly of Gabor filtering of the image to create a feature set that can be clustered (via k-means) to segment the textures present in the image. The components of the algorithm are shown in Figure 11, and are discussed next.

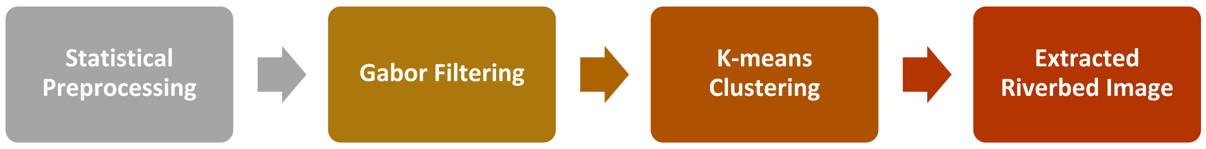


Figure 11. components of Step 2 of the image processing algorithm.

A classic approach to texture segmentation with Gabor filters was successful. It was found that applying a binarization in noisier images was more effective than plain entropy preprocessed images. The unique approach in this study was applying an entropy filter as a preprocessing method in Gabor filter feature extraction. Combined entropy and range filtering was not effective in the preprocessing.

5.2.1. Statistical preprocessing

As in step one of the algorithm, statistics were applied to the image to prepare the image for edge extraction. This was part of the preprocessing phase. A variety of edge statistical analyses, such as variance and standard deviation were also considered, but yielded similar results to the entropy or range filtering techniques. At first, entropy and range were considered to enforce a symmetrically similar preprocessing design to the first part of the algorithm. However, this was not effective. The range filtering disordered the intensities of each pixel, which had the effect of incorrectly discriminating textures. Entropy alone proved to be the most effective in discriminating the textures across an accurate boundary in space. However, in noisier images, applying a binary threshold image was the most beneficial in extracting a preliminary bridge boundary.

5.2.2. Gabor filtering

The Gabor Filter is a linear edge detector used for edge detection and texture analysis. Gabor filters share similarities in frequency and orientation response to those of the human visual system (Malik and Perona 1990). They are particularly suitable for discrimination and texture representation. A two-dimensional Gabor filter is a Gaussian kernel function modulated by a sinusoidal plane wave in the spatial domain (Matlab 2017; Atherton 2011). The Gabor filter impulse response is characterized by a sinusoidal wave, using a cosine in the real plane and a plane wave in two-dimensional space. The sinusoidal is multiplied by a Gaussian function that acts as the kernel, or basis function. Thus, it is apparent that the Fourier transform of a Gabor filter's impulse response is defined as the convolution of the transformed harmonic function and the transformed Gaussian basis function (Clausi and Jernigan 2000). In addition, the Gabor filter has a real and an imaginary component demonstrating orthogonal directions (Jain and Farrokhnia 2000).

As with the Fourier transform, a Gabor-filtered image is formed by superimposing a series of sinusoidal waves of different frequencies that are orientated in several directions. By examining pixels of the image, the effect of the transform can be physically realized. The pixel value shows the intensity of the sinusoidal wave, while its position shows the frequency and orientation of the wave (Malik and Perona 1990; Daugman 2017). The selection process of these waves is further discussed below. The Gabor transform acts as a bandpass filter cutting off the Fourier transform at certain frequencies to isolate specific information within the input image.

Feature extraction is an important function of Gabor filters. A set of Gabor filters with different frequencies and orientations can be used for extracting features of interest in an image. Based on the scanning sonar images chosen, segmentation is visually obvious because of the difference in texture between the consistent, repeating pattern of the bridge pier structure, and the irregular, rough texture of the seafloor (Reed and Hussong 1989). These expectations are an idealization; presence of debris, protective structures, and modifications to piers further complicate image segmentation.

Before applying the Gabor filter, several parameters including the wavelength and orientation were determined. The **Gabor** function used these parameters to create a Gabor filter bank with filters of the defined sinusoidal wavelength in pixels/cycle via the row and wavelength parameters, while orientation was defined in degrees normal to the sinusoidal plane wave. In implementation, these parameters were supplemented by minimum and maximum wavelengths, and the angle step size to prepare the bank of filters that are then applied to the image (Jain and Farrokhnia 1991). In addition, the spatial-frequency bandwidth and the aspect ratio of the Gaussian in the spatial domain were considered. The spatial-frequency bandwidth is the cutoff frequency of the filter response (Mehala and Dahaya 2008). This is realized when the frequency content in the input image diverges from the set frequency and is therefore defined as the inverse of the set wavelength (Lee 1996). Typical values for spatial-frequency bandwidth range from 0.5 to 2.5 in the units of octaves (Mathworks 2017). The aspect ratio of the Gaussian in spatial domain is defined as a vector of numbers that expresses the Gaussian envelope as a ratio of its semi-major and semi-minor axes. In practice, this parameter controls the ellipticity of the Gaussian envelope. These values typically range from 0.23 to 0.92. For the current study, the default values of 1.0 and 0.5, respectively, were found to be sufficient for the spatial frequency bandwidth and aspect ratio.

In general, the set of frequencies and orientations used to create a Gabor filter bank is designed to localize dissimilar, orthogonal, subsets of frequency and orientation in the input image. In this study, the orientation was sampled from 0 to 135 degrees in steps of 45 degrees and the wavelengths were sampled in increasing powers of two starting from $\frac{4}{\sqrt{2}}$ up to the hypotenuse length of the input image. The angle step size was tested at a smaller value to enhance discrimination and resolution of boundaries in particularly noisy images. This did not have an impactful effect on the results, however, and the angle step size was kept at 45 degrees. Similarly, minimizing the size of the wavelength step size also did not greatly impact the results. Therefore, the filter bank design by Jain and Farrokhnia (1991) was implemented in this study.

Feature set post-processing and clustering

The magnitude response of each of the Gabor filter bank components, also referred to as Gabor Energy, was a unique feature, or principle component of the input used in the algorithm to cluster and segment the textures within the image. However, post-processing is required before the Gabor magnitude responses can be used in clustering and classification. This post-processing includes Gaussian smoothing, which adds positional information to the feature set. This reshapes the feature set to the form expected by the k-means function, as well as normalizing the features to a common variance and mean.

Each image of a Gabor magnitude contained local variations, even within regions of uniform texture. These local variations corrupt the segmentation process. In terms of a bridge pier structure, presence of debris, irregular alterations to the pier structure, instrumentation, and scour protection measures within the image can alter the acoustic footprint of the texture. These areas would have darker or more irregular acoustic shadows cast upon them as a result of weaker returns to the transducer due to scattering of the incident beam at these irregular areas (Reed and Hussong 1989; Atherton 2011). This weaker signal appears as a shadow, or duller intensity, despite being the overall same texture as the rest of the bridge.

Smoothing the Gabor magnitude features through a Gaussian low pass filter counteracts the variations due to the presence of the above-mentioned irregular features. Thus, a sigma that is matched to the Gabor filter that extracted each feature was chosen. This is crucial since it is adaptable to the image and feature input (Lee 1996). Moreover, a smoothing term, *K* (which may be tuned), is determined to allow the algorithm greater freedom to control the quantity of smoothing applied to the Gabor magnitude responses. If the AUV detects the aftermath of a stormy condition, where the bridge structures could be obstructed by debris, this smoothing could retain the overall texture characterization of the bridge against the background and riverbed.

Finally, when the Gabor feature sets were created, for texture classification, it was found to be of great use to add spatial location information as column (x) and row (y) coordinates. This additional mapping information adds another two dimensions to cluster the data with, and causes the classifier to group textures which are in the same spatial vicinity. Thus, the attributes being fed to the classifier include the Gabor Energies at various filter wavelengths and orientations to classify the true intensity characterizations of the texture, as well as spatial locations to enhance the accuracy in localization and discrimination of textures. A total of 26 distinct features was stored in the variable **featureSet** in Matlab. In terms of the feature space, each pixel in the image was represented as a distinct data point, within a plane representing each feature to which it relates (Mathworks 2017). In order to feed these features into a classifier, they were reshaped into a matrix form. The features were normalized to have a zero mean and exhibit unit variance. This post-processing was done in order to further enhance the differences and similarities within the feature set.

Applying a Principle Components Analysis (PCA) is a method of confirming that there is sufficient variance in the Gabor feature data. This information is often plotted visually to quickly and intuitively assess the effectiveness of the Gabor features in classification and clustering. This was useful in developing the algorithm. For instance, with this PCA, it was realized that entropy alone was not sufficient to segment at least three textures (bridge structure, seafloor, and background), from the image. Mathematically, this implementation of PCA maps the data from a 26-dimension representation of each input pixel image into a 1-dimension intensity value for each pixel. Having a distinctive variance among the data set is imperative in obtaining an accurate characterization of textures. An example of one of the test cases applied is shown in Figure 12.

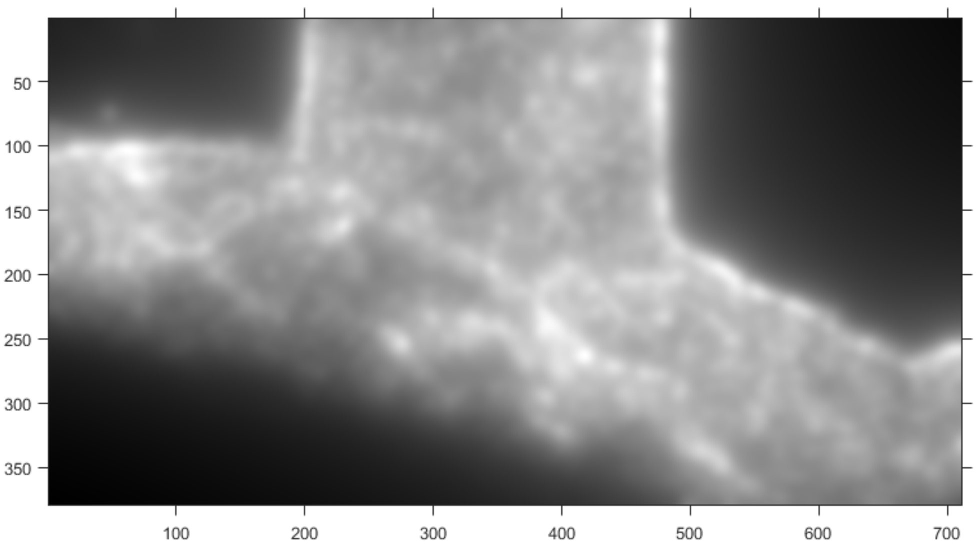


Figure 12. PCA analysis of feature set (axes in pixels) applied to case 1.

5.2.3. K-means clustering

The textures were ultimately segmented using the K-Means clustering algorithm of Matlab's built-in function. K-means clustering is a partitioning method based on the average squared distance between points in the same cluster (k++) (Ghaemi et al. 2009; Arthur and Vassilvitskii 2006). Its accuracy has been improved with use over the years, resulting in k-means++ algorithm, discussed below. K-means analysis is a non-hierarchical cluster definition method driven-by an iterative process (Ortega et al. 2009; Mathworks 2017). That is, in every iteration of the algorithm, the value of each member in a cluster is reassessed. This reassessment and update process is founded on the current centroid of the clusters (Chen 2009). In general, this process is repeated until the chosen number of clusters is reached at an appropriate convergence-level (Peeples 2011; Kintigh and Ammerman 1982).

The Matlab implementation partitions data into a predefined number of k, mutually exclusive, clusters. The value of k was predefined as the number of expected textures to be segmented in the image, which is three. By default, k-means uses the squared Euclidean distance measure and the k-means ++ for cluster center initialization. K-means clustering operates on live observations to create one level of clusters. Older clustering techniques used hierarchical clustering, which operate on a decisional set of dissimilarity measures (Arthur and Sergei 2007; Kumar and Wasan 2010). This difference is the reason that k-means clustering is often more appropriate in handling big data (Ferguson 1982). This includes large sets of images, and was thus selected as the classifier and clustering technique in this study. Based on the data set, the cluster centroids and the maximum number of iterations could be set to save computational resources; however, care must be taken to avoid inaccurate convergence. By default, the Matlab k-means function uses the k-means++ algorithm to initialize cluster centers. Because the image data set was variable, the initialization of centroids is not predefined. Therefore, the more appropriate method was the kmeans++ algorithm, which was utilized in this study.

The k-means++ algorithm extends k-means with a mathematically simple and randomized seeding technique to initialize centroid locations (Arthur 2007). Even though this introduces an overhead in the initialization of the algorithm, it provides an important benefit, namely, reducing the probability of a bad initialization, which would lead to a bad clustering result. Thus, k-means++ improves both the speed and the accuracy of k-means.

The only prior information required in the algorithm are the number of texture blocks to be segmented in the image (Sorensen and Ludvigsen 2015). This is a limit to clustering algorithms in general and is a popular area of research in Machine Learning. In implementing the algorithm proposed in this study, this limitation could lead to error in the presence of debris at the pier-riverbed interface. In general, the textures to be segmented were the bridge, the riverbed, and the background, for a value of k =3 clusters. In the presence of debris, this would make the debris appear as a part of the riverbed. Normally, this would not be as issue because the smoothing process eliminates such noise from the extracted seafloor. However, if this debris fills a scour hole, the randomness of both the debris and the hole could make them appear as a texture, resulting in a large error in predicting scour hole depth. Based on the geometry and quality of the input image, the number of clusters was variably set to either 3 or 4. The riverbed segment was extracted by storing it as its own image with a null, black, background. This segment was one of the inputs in Step 3, along with the bridge information extracted in Step 1.

5.3. Image processing algorithm step III: reconstruction through bridge scour model

The reconstruction of the image consisted of integrating the key results from Steps one and two. Specifically, this included plotting the Hough lines to signify the bridge pier and, plotting a boundary curve correlating to the extracted riverbed, with minima and maxima easily localized and relatable to the Bridge. This information can be exported into the geographic information system databased developed as part of this study, and can be tagged to a specific bridge following an AUV mission. The flowchart of this part of the algorithm consisted of converting the riverbed image to a curve and plotting the bridge pier and foundation system.

The algorithm first read in the boundaries segmented by the texture segmentation portion of this study. The boundary was then converted from a two-dimensional image to a one-dimensional signal representing the height of the riverbed. If a one-dimensional signal is desired as an output, then it should be acquired by a sensor that is made for that purpose, like a bathymetric sonar. However, acquiring two-dimensional images from a sonar includes more information about the environment as a nearly full documentation of the bridge condition. This allows research in high-level environmental assessments, which is the aim of future work.

In order to achieve the aforementioned conversion, an algorithm was developed to search the image matrix and find the areas where the image changed from low (0), or background, to high (1), or foreground, or the riverbed. The image was first converted from grayscale to a binary image to emphasize boundaries. Then, nested for loops were used to iterate through the pixels until a change was detected. Once a change from low to high was detected, the pixel location was stored in a vector of length equal to the number of columns in the image and the algorithm continued its search over the image. The column indices were the independent variables. The row values ultimately became the dependent variables which were plotted over the image to show the variations in the riverbed in space. Subsequently, smoothing was applied using the Savitzky-

Golay filter to remove local oscillations. Finally, the Hough line results from step one were simply plotted over the riverbed curve. These line segments were stored in matrix form.

5.4. Results

The results of each step of the algorithm are presented in this section. The advantages, limitations, and challenges in each of the components of the algorithm are identified, along with outputs in each step.

The Matlab implementation of the codes described in previous sections is not appropriate for run-time usage with AUV missions. In practice, some of the analysis can be done offline after an AUV mission is completed. The ideal implementation allows for real-time decision making, in particular in recognizing and navigating around bridge piers in scour monitoring/assessment missions. For this purpose, the Nvidia Jetson TX1 platform was chosen. This is a development board based upon a Tegra processor with both ARM CPU cores and GPU cores. Low level access to the GPU cores for computation is through CUDA. An OpenCV port is provided by Nvidia, which accesses the GPU cores for parallel processing.

Some of the computer vision algorithms developed in Matlab were directly available in OpenCV, but many others required custom development. These codes were developed as part of this study, and are provided in Appendix A. While the OpenCV library on the Jetson is tuned for GPU processing, future work can more completely leverage the performance enhancement possible with parallelization on GPUs for optimization of computing resources onboard the AUV.

5.4.1. Step one results

The goal of step one was to detect and extract the bridge pier features from an acoustic image. The main methods used were Savitzky-Golay filtering to reduce horizontal components and enhance vertical components of the image; specifically, these were the edges of the bridge pier. Next, the entropy and range filters were applied locally to these edges. Then, a Prewitt filter in the vertical direction was used to detect and specifically extract vertical edges. Finally, a Hough transform was used to gather and determine these edges as a set of true edges or Hough lines, as a vertical feature set. The parameters used for Step 1 in processing of the images for the six bridge piers considered in this study are shown in Table 1.

Table 1. Summary of parameters set for step one of the image processing algorithm.

Case	Image Width x Height (pixels)	Savitzky-Golay Filter Window Size (pixels)	Number of Set Hough Peaks	Minimum Size of Peak (pixels)	Fill Gap Correction (pixels)	Prewitt Filter Threshold
1	710 x 380	31	2	150	70	0.3
2	310 x 180	11	6	75	90	0.3
3	410 x 130	11	4	70	70	0.3
4	570 x 270	11	8	175	70	0.3
5	520 x 170	11	2	110	10	0.3
6	1000 x 240	11	2	100	70	0.3

A great level of variation was not needed for the Savitzky-Golay filtering phase of the algorithm. It was found that over-smoothing eliminated too many key frequency components, or worse, created false positive results. In most cases, a Savitzky-Golay filter order of 3 and a window size of about 5-10% of the image were sufficient. As mentioned in the Algorithm, a lower filter order and a higher window size have a greater effect on smoothing. The filter design was verified experimentally. In cases where there were more bridge components, such as case two, case three, and case four, less smoothing was needed to preserve target discrimination for each of these edges. Hence, a higher filter order of 3 was used. Furthermore, it was found that noise had the greatest effect on Savitzky-Golay filter design. The image used in case one had less background noise reduction applied to it compared to the other images used in the study, and, therefore, needed a wider window size of smoothing applied to it than the other cases. Nonetheless, a scalable filter design of a conservative 5% of height Window Size and third order was found to be scalable in this study with respect to image height and noise level. Parameters used for the filtering are shown in Table 2.

Table 2. Entropy-range results: Savitzky-Golay filter rules.

Case	Savitzky-Golay Filter Window Size (pixels)	Image Height (pixels)	Window Size to Image Height (%)	Savitzky-Golay Filter Order
1	31	380	8.16%	3
2	11	180	6.11%	3
3	11	130	8.46%	3
4	11	270	4.07%	3
5	11	170	6.47%	3
6	11	240	4.58%	3

The main settings and variables that were modified in this step were the number of Hough peaks to detect, the minimum size of a peak and the fill gap correction. Table 3 shows how the rule set was defined for each of these variables. The number of Hough peaks was set to match the number of true bridge edges in each case. Because the minimum size of peak and fill gap correction were in terms of pixels, it was reasonable to define these in terms of percentage to the full image height. While the minimum size of peak percentage was roughly 50% of the image height, the percentage of the fill gap correction to the image height varied with each case. This indicates that this correction is not related to the image size. Instead, a requirement for a lower fill gap correction is related to greater image noise. This is because image artifacts could be misread as new or extensions to the true bridge edges. In addition, the similarity in textures of the bridge to the seafloor could also create a false peak if the fill gap correction was too high. Hence, a strong noise reduction and vertical edge enhancement phase is crucial to step 1.

The fill gap correction was set to be proportional to the noise level. Although the Hough ceiling parameter could have also been altered at each step, it was found that maintaining a constant 0.3 ceiling was sufficient for the examples considered in this study.

Table 3. Hough peaks and line settings

Case	Image Height (pixels)	Number of Set Hough Peaks	Minimum Size of Peak (pixels)	Minimum Peak/Image Height (%)	Fill Gap Correction (pixels)	Fill Gap Correction/Image Height (%)
1	380	2	150	40%	70	47%
2	180	4	75	42%	90	50%
3	130	4	70	65%	70	65%
4	270	6	175	65%	70	26%
5	170	2	110	65%	10	6%
6	240	2	100	42%	70	29%

Table 4 lists the effectiveness of the fixed settings described in Table 3. It can be seen that although not all peaks were identified, as can be seen in the results of cases three and four, the detected peaks were only true peaks. That is, although not all six true peaks were identified, the two peaks that were not identified were incorrectly detected as a more dominant true peak. This signifies that the algorithm did not produce a false result. Rather, the algorithm showed a weakness to peaks that were under a certain percentage of the full image height. This can be directly incorporated into an automatic scour image analysis algorithm in order to avoid detection of false peaks.

Table 4. Entropy results: Hough peaks in the examples considered

Case	Image Width (pixels)	Image Height (pixels)	#of True Peaks in Case	# of Matches to Truth	# of False Peaks	Correct Matches (%)	Peak Height Accuracy (%)	Horizontal Spatial Accuracy (%)
1	710	380	2	2	0	100	96%	98%
2	310	180	4	4	0	100	100%	100%
3	410	130	4	2	0	79%	85%	83%
4	570	270	6	4	0	94%	97%	100%
5	520	170	2	2	0	100	100%	100%
6	1000	240	2	2	0	100	100%	100%

Taking this under consideration the percent correctness measures were weighted to reflect that the algorithm did not identify an incorrect peak, but, doubly represented a true instance. These weights were found by taking a ratio of the length of the missed vertical peak to the total height of the image. Next, the sum of these ratios was found. Then, the ratio of the missed peaks to the total number of peaks was found. Finally, the sum of height ratios was multiplied by the peak number ratio. This was the total percent in the detection algorithm per image.

As an example, the missed peaks for case three were: 25 pixels and 30 pixels with an image height of 130 pixels, therefore,

$$\left\{\left(\frac{25}{130}\right) + \left(\frac{30}{130}\right)\right\} \times \mathbf{100} = \mathbf{42\%}\ \boldsymbol{Weighting} \tag{9}$$

Applying this correction,

$$\left(\frac{\mathbf{2}\ \boldsymbol{Missed\ Peaks}}{\mathbf{4}\ \boldsymbol{Total\ Peaks}} = \mathbf{50.0\%}\ \boldsymbol{Error}\right) \times \mathbf{0.42} = \mathbf{21\%}\ \boldsymbol{Error} \tag{10}$$

Also, for case four, the missed peaks were: 25 pixels and 30 pixels with an image height of 270 pixels, therefore,

$$\left\{\left(\frac{25}{270}\right) + \left(\frac{25}{270}\right)\right\} \times \mathbf{100} = \mathbf{18.5\%}\ \boldsymbol{Weighting} \tag{11}$$

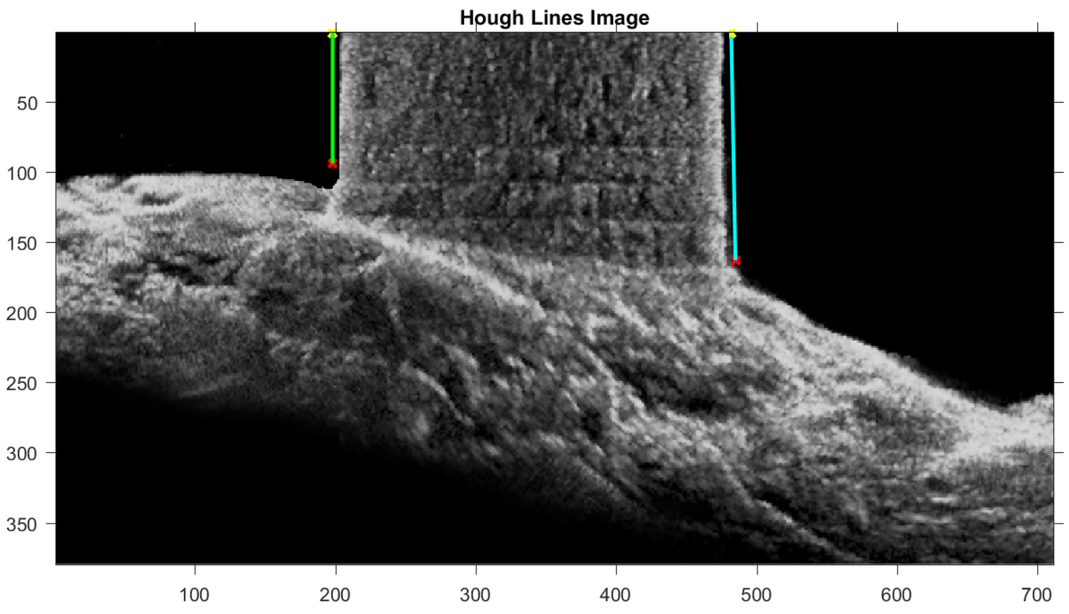
Applying the correction,

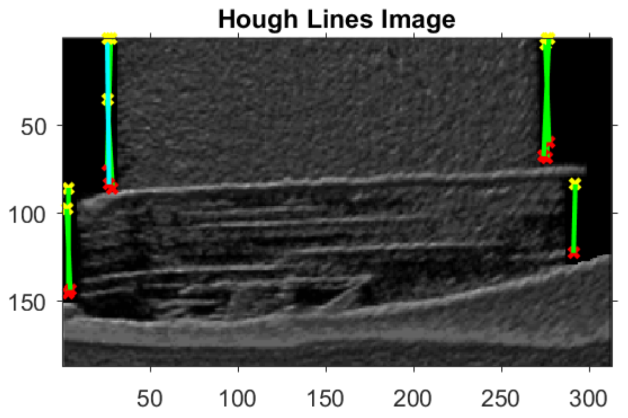
$$\left(\frac{\mathbf{2}\ \boldsymbol{Missed\ Peaks}}{\mathbf{6}\ \boldsymbol{Total\ Peaks}} = \mathbf{33.3\%}\ \boldsymbol{Error}\right) \times \mathbf{0.185} = \mathbf{6.17\%}\ \boldsymbol{Error} \tag{12}$$

These results indicate that if a pier and foundation are present in an image, a reasonable aspect ratio must be preserved to accurately depict all components of the bridge. This is seen through case three and case four, where the peaks of the bridge that were not detected were missed but, were not false. Cases two and three are reasonably comparable in their target, their seafloor curves seemingly being the main difference in the images. However, the bridge detection algorithm in case two succeeds where case three does not. Each image has four Hough peaks, and case two detects them all, while case three does not. This difference was found to be attributed to the aspect ratio of the image. For all components of the bridge to be detected, they must be nearly 25% of the height of the image. In case three, the smaller peaks represented the bridge pier. They were 25 and 30 pixels each over a 130-pixel height, yielding an average peak to height ratio of 21 percent. In case two, the smaller peaks represented the bridge foundation. They were 50 and 40 pixels each over a 180-pixel height, yielding an average peak to height ratio of exactly 25 percent.

It was found that the horizontal placement of the piers as detected by the algorithm also varied. These errors were for the most part negligible. Nonetheless, it was found that images with greater noise in the background were more likely to display errors in the horizontal spatial accuracy of the image. Reducing the horizontal noise and enhancing the features was found to be key in eliminating these errors.

The conclusions from the results gathered in Step One indicate that quality background noise removal was needed for accurate detection of edges while excess data reduction is needed to enhance edges. Smoothing was extremely significant. Smoothing was used in this phase to not only eliminate background noise but also to enhance the bridge edges, which the signals of interest. Hence, a balance between eliminating background noise and enhancing edges was key to this study. Over-smoothing led to noise being detected as pier edges (false positives) and under-smoothing led to both pier edges to be missed (false negatives) and background noise to be detected (false positives). The metrics for success in this phase of the algorithm were first and foremost the column accuracy of the line segments on top of the bridge widths and the length of the detected segments. It was found that a suitable solution was using a cascade of direction-sensitive filters. Hence, globally applied filters were replaced with the Savitzky-Golay filter for its more tunable input arguments as well as directional smoothing and the Prewitt direction-sensitive edge detector. Results of the bridge pier identification are graphically shown in Figure 13 and Figure 14, where the automatically detected vertical boundaries of the bridge piers are shown using lines.





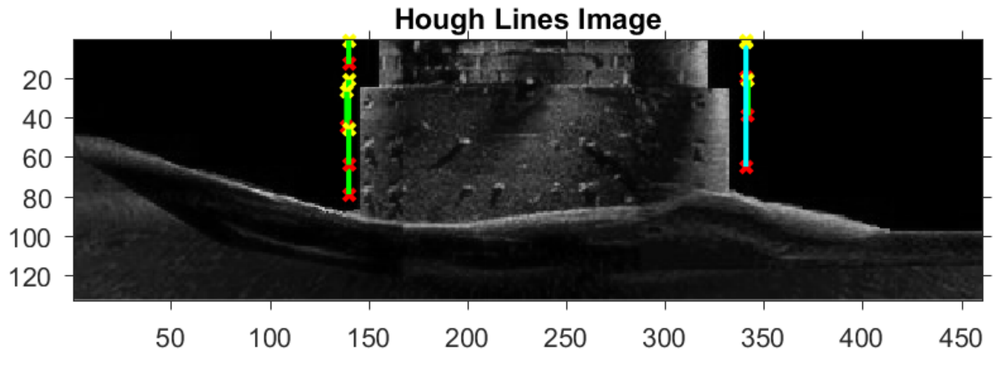


Figure 13. Automatic detection of vertical pier components in cases 1-3.

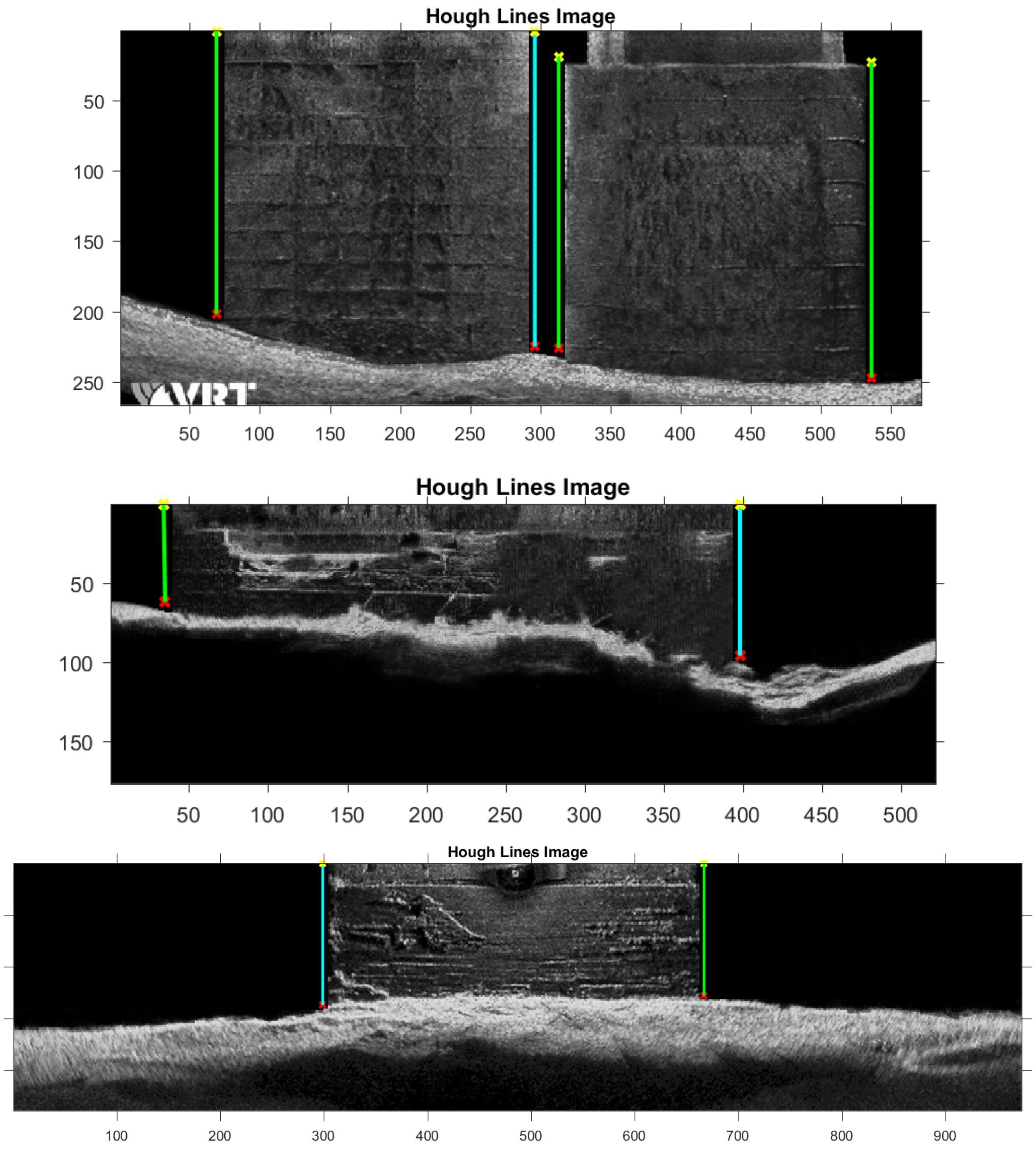


Figure 14. Automatic detection of vertical pier components in cases 4-6.

5.4.2. Step two results

The goal of Step 2 was to extract the seafloor from the rest of the image. This was successfully achieved via a texture segmentation approach. This process is inherently application-specific. However, entropy alone as a preprocessing technique was generally useful in most cases. It was found that applying a binary threshold also improved partitioning. After preprocessing, a Gabor Filter bank was used to create an image feature set that was then applied as the input to a kmeans++ clustering algorithm. The clustering algorithm segmented textures based on their Gabor Filter responses and pixel locations. As the seafloor texture is more random than the bridge or background of the image, a texture segmentation technique was successful in most cases.

The texture algorithm remained constant in the majority of the bridges studied, with entropy preprocessing applied to a k-means cluster solution of k=3 clusters measured through Euclidean distance. However, in cases two, four, and five, the clustering solution needed to be set to 4. Results of the segmentation techniques applied to the six bridge pier cases are shown in Figure 15.

5.4.3. Step three results

In this final step, the results of step 1 and step 2 were combined to a displayable format. There were a few key design challenges in this part of the algorithm. As previously discussed, for each case, the segmented boundaries were binarized to emphasize their edges for accurate discrimination of the boundary, as can be seen in Figure 16 for case 1. Challenges included selecting an accurate yet, effective smoothing filter. The Savitzky-Golay proved to be the most appropriate as it fits a polynomial to set of data points locally without over smoothing. The process of creating a boundary curve meant creating vector of discrete points to represent the edge variations of the seafloor boundary. Application of a smoothing filter was found to be effective in these cases. This can be seen in Figures 17 and 18, which show the effect of smoothing at different orders and window sizes of the function on the resulting edge detection at the riverbed. Ultimately, a frame length of 10% of the image width and order of 3 were chosen because they had the most median effects and computational complexity to the image.

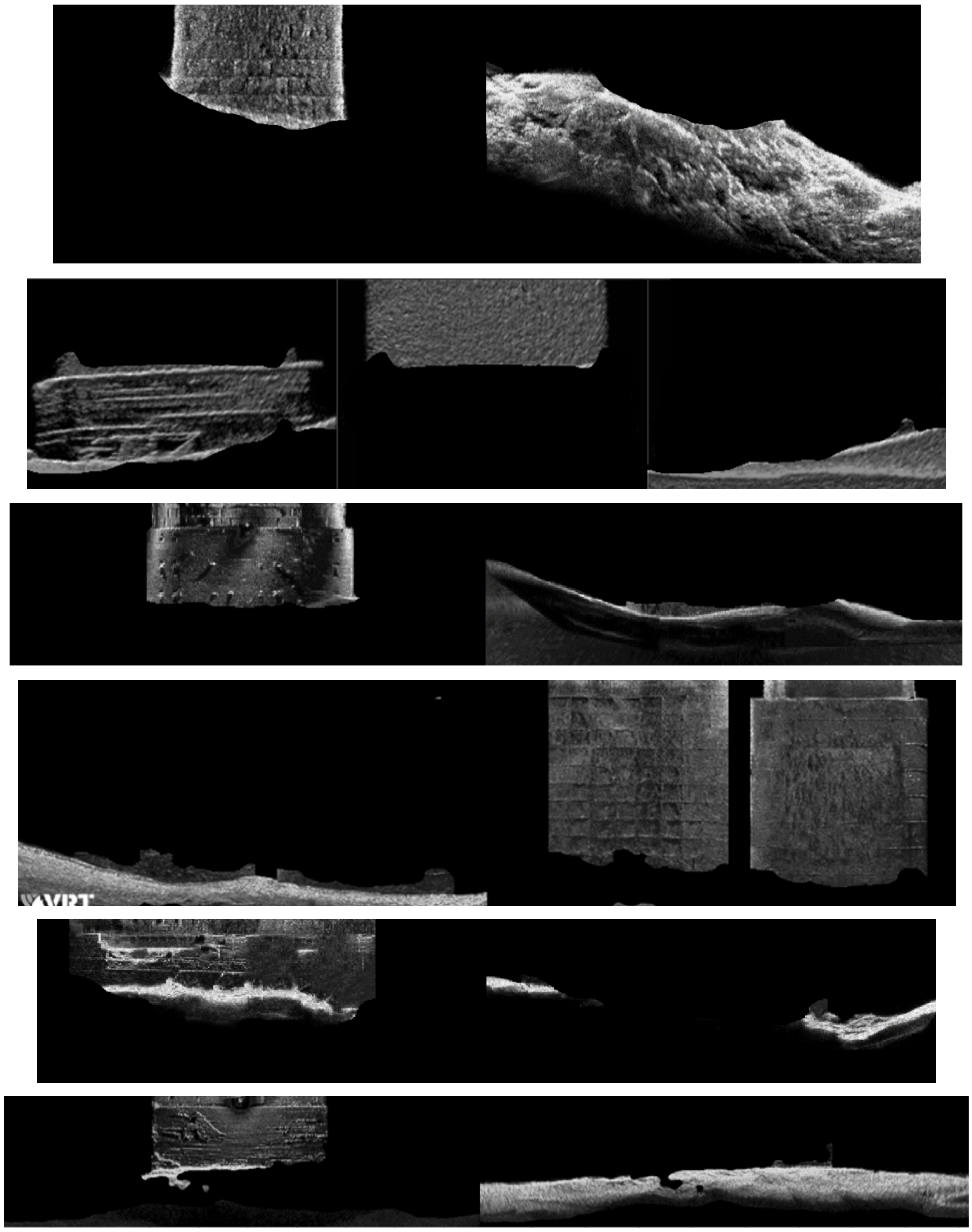


Figure 15. Results of segmentation techniques applied to separate the river bed from the bridge pier for cases 1 through 6.

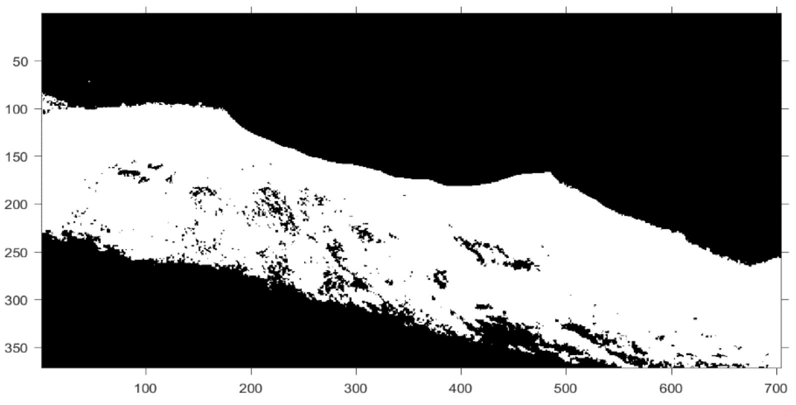


Figure 16. Thresholding applied for enhancement of edge detection applied to case 1.

Results of the image processing applied to the six cases considered in this study are presented in Figure 19 and Figure 20. The inaccuracies evident in some of the results are due to several factors, for which solutions were devised, but the raw processing was presented to illustrate the shortcomings in the original algorithms. For example, it can be seen that because of poor signal gradients in Case four, a false peak was detected on the right side of the pier. Further smoothing eliminated this and similar false peaks.

In case five, the poor quality of the image resulted in loss of important image signals during the thresholding process. In such cases, the upstream and downstream side of the image can be used to estimate the riverbed. Adopting larger window sizes in the smoothing algorithm can eliminate the majority of the resulting false peaks.

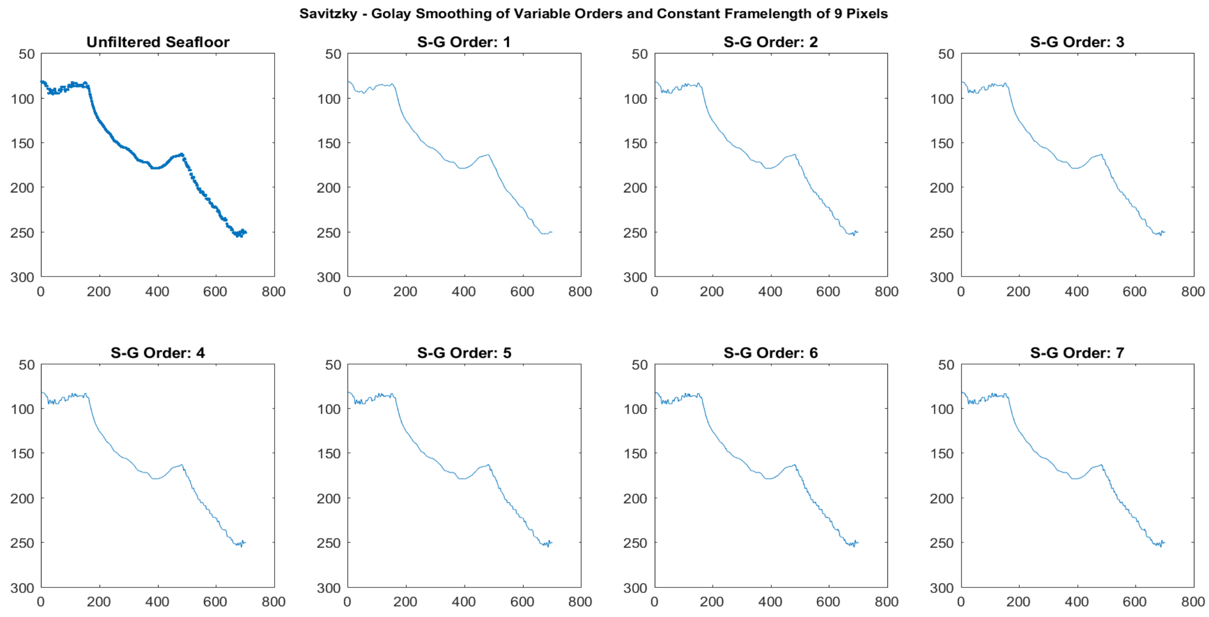


Figure 17. Savitzky-Golay filter at constant frame length and variable orders applied to case 1.

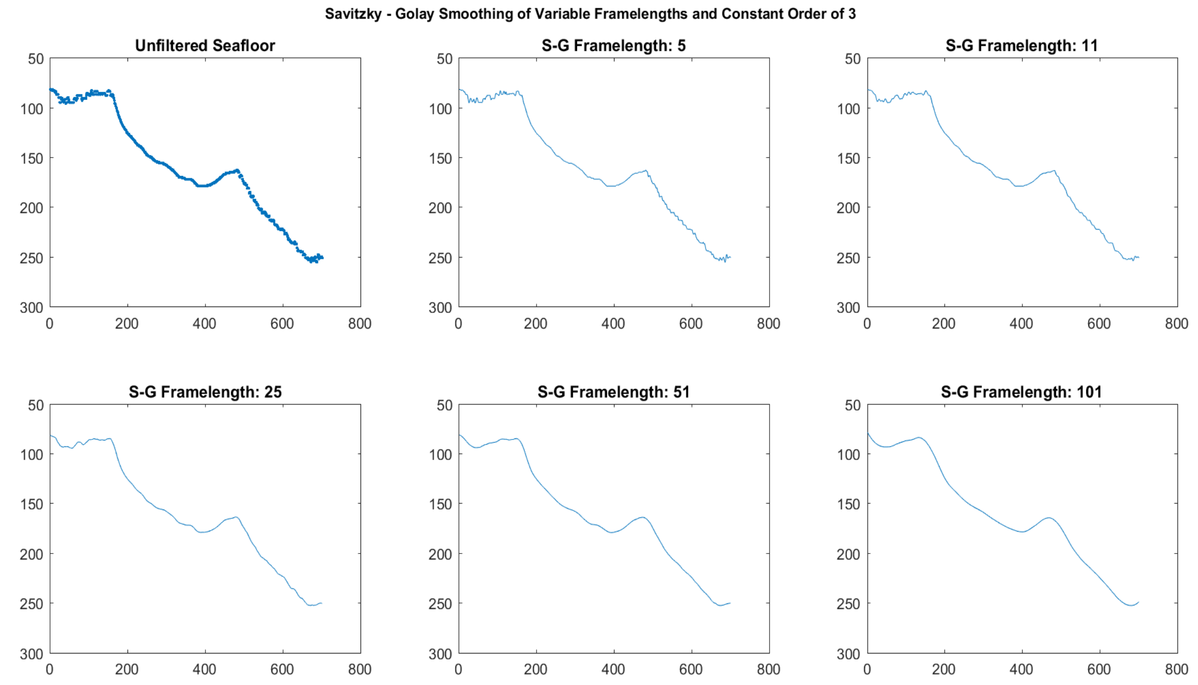


Figure 18. Savitzky-Golay filter at variable frame lengths and constant order of 3 applied to case 1.

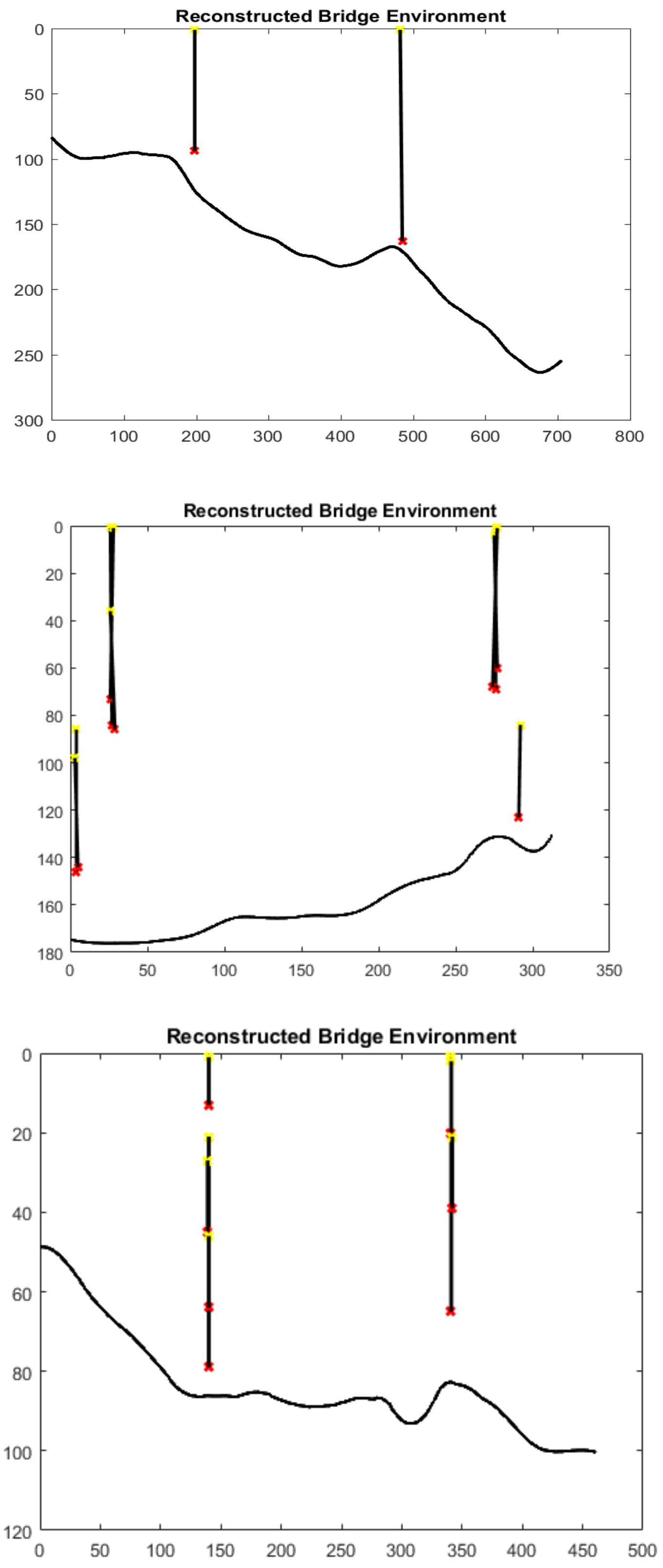
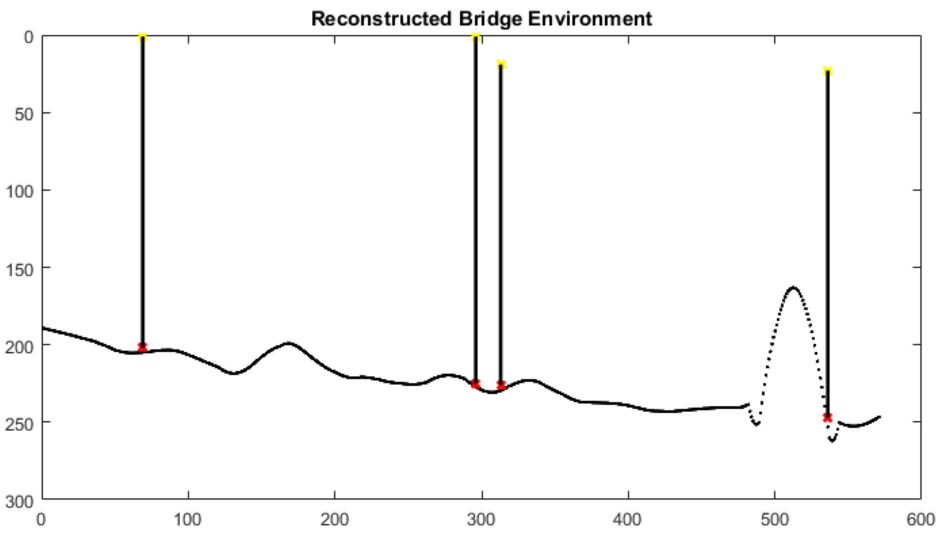
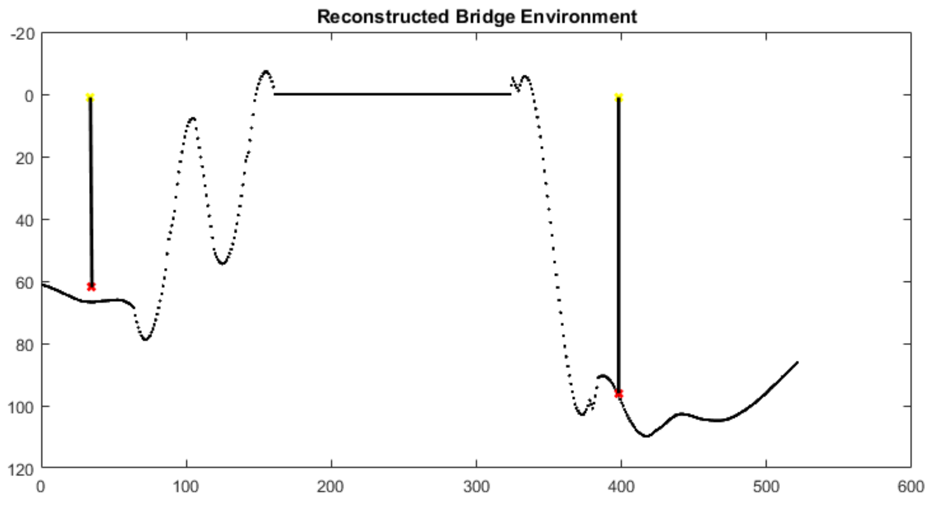


Figure 19. Depiction of bridge pier and riverbed in cases 1-3.





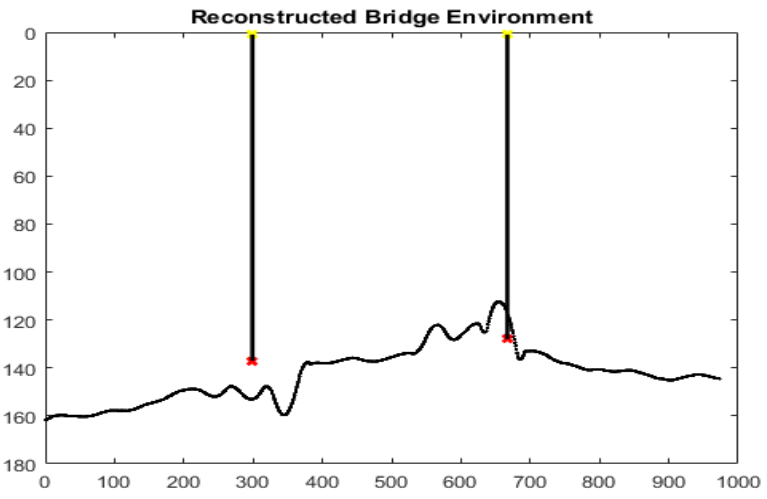


Figure 20. Depiction of bridge pier and riverbed in cases 4-6.

6. PATH FINDING

The path finding algorithm was developed using MOOS-IvP (Benjamin 2013), a multi-objective optimization system based on interval programming (MOOS: Mission Oriented Operating Suite). MOOS is an open source suite designed path simulation for underwater vehicles. The MOOS-IvP software was specifically designed to handle operations with AUVs and similar watercraft. The system is driven by three different design philosophies: the backseat driver design philosophy, the publish-subscribe design philosophy, and the behavior-based control design philosophy. The backseat driver design philosophy allows for the distinction between vehicle control and vehicle autonomy. Vehicle control is run on the main vehicle computer while vehicle autonomy is run on a separate payload computer. This design decouples the autonomy system from a vehicle's hardware. A navigation and control system is provided that is capable of streaming vehicle position to the main computer. The system also receives a stream of autonomy decisions. How the vehicle navigates is unspecified to the autonomy system running in the payload computer.

The publish-subscribe middleware design philosophy allows processes to communicate with each other through a database called MOOSDB, or MOOS Database. Messages are available in a variable-value pair, such as NAV SPEED. Parameters such as speed, heading, depth, and mode are available through the MOOSDB. This accessibility makes applications largely independent, and allows for the addition or removal of behaviors. Editing existing applications is also simplified as a result of this approach. The behavior-based control design philosophy allows for the creation of behaviors. Behaviors are software modules that control an aspect of vehicle autonomy. The helm implements each behavior and allows for the configuration of a set of missions (Benjamin et al. 2009). The helm also contains mode spaces that determine which behaviors are active. When multiple behaviors are active, the IvP solver reconciles the behaviors. An example of AUV simulation using MOOS-IvP is shown in Figure 21.

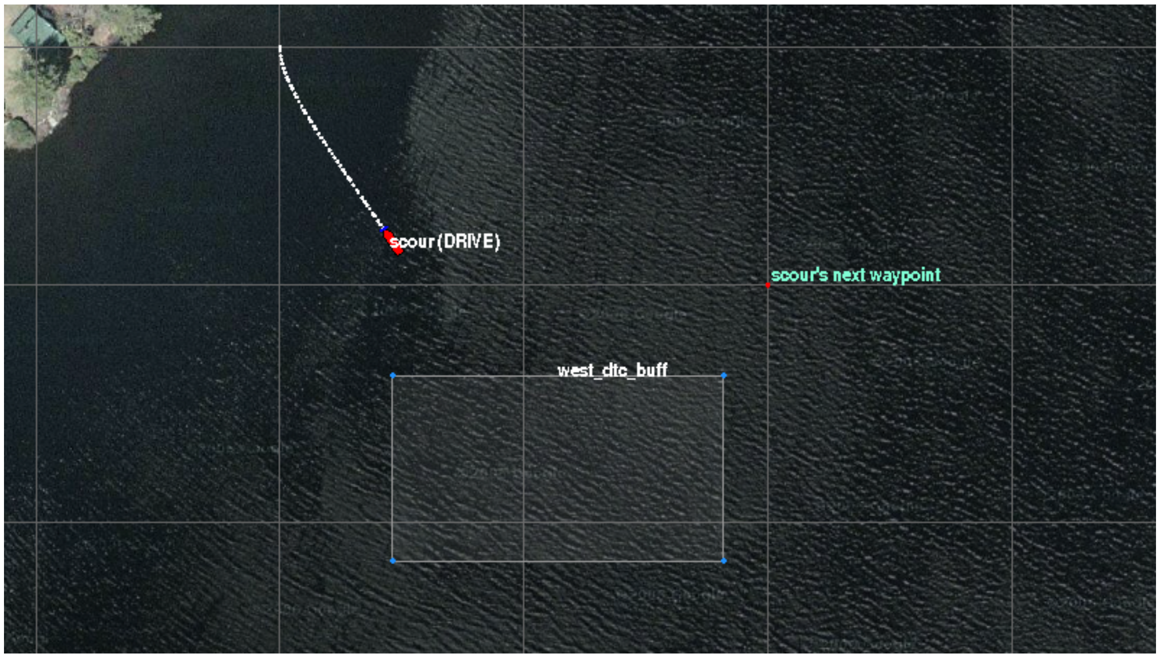


Figure 21. Example of navigation simulation in MOOS.

Each mission in the MOOS simulations is comprised of behaviors that are integrated into the helm system. They contain two C++ files, two C header files, a .bhv file, and a .moos file. The two C++ files contain the code for the behaviors and the C header files correspond to the C++ files. The .bhv file is where the specifics of the behaviors are set, such as initial location, speed, mode, depth, and other parameters specific to each behavior. The .moos file contains information about applications such as pHelm IvP, uSim Marine, and uHelm Scope. The behaviors used to create the bridge scour navigation mission were behaviors that were adapted from existing behaviors contained in the MOOS-IvP system. The behaviors used were Simple Waypoint, Avoid Obstale, Survey, Constant Depth, Constant Speed, and Station Keep. Avoid Obstacle was renamed Object Detect and Survey was renamed Loitering to reflect their new functionalities. The components of simulated AUV missions are described next.

Navigation algorithm: This simulation is run through deploy option in the software. When the AUV is deployed, Simple Waypoint is activated. When the survey button is pressed, Object Detect is activated. The AUV is aware of its first waypoint, but it detects that there is an object present and moves toward it.

The Object Detect behavior prevents the AUV from colliding with objects present in the water. The white line visible in Figure 21 is a track line that shows the AUV’s path. Additionally, the dotted line around the object represents a buffer zone which the AUV will not cross. When it reaches the object, the AUV switches to the Loitering behavior and begins its scanning pattern around the object. This pattern is characterized by decreased speed and implementation of the Station Keep behavior. The decreased speed allows a sonar to scan the pier and riverbed for signs of scour. The Station Keep behavior enables the AUV to stop at each side of the pier and shift its position and attitude so it can scan effectively. The AUV makes one pass around the object in this fashion. Once this pattern is completed, the AUV continues to the original waypoint and the simulation is finished.

Mode hierarchy: Each mission has its own mode hierarchy. Modes dictate the state the AUV is in and can potentially allow an AUV to dynamically switch between behaviors. Examples of modes that are generally used in behaviors are ACTIVE, INACTIVE, DEPLOY, and RETURN. A new mode was created for this mission with the intent of detecting bridge scour. The mode, named NAV BSA MODE, has several states that it can switch between. In this simulation, the AUV begins in the IDLE state. Once it is deployed, the AUV switches to the ACTIVE mode. Simple Waypoint and Object Detect are initialized to begin when the mode is ACTIVE. A state diagram depicting the mode hierarchy is shown in Figure 22.

NAV BSA MODE is initialized to SCOUT which indicates the Object Detect behavior is engaged and the AUV is actively searching for objects in the water as seen in Figure 23. Once it detects an object, the AUV will move towards it until it reaches the buffer zone around the object. When it reaches the buffer, the mode will switch to SENSE as seen in Figure 24. In the SENSE mode, the AUV will begin the scour detection pattern around the base of the pier. After the AUV has made one pass around the object, NAV BSA MODE switches from SENSE mode to SKEEP mode, as shown in Figure 25. In the SKEEP mode, the Station Keep behavior is active. Once it has completed all Station Keep points, the AUV will continue to the original waypoint. The key to the AUV’s autonomy is mode switching. Through these algorithms, anew method of mode switching was developed in this study that gives the AUV more decision-making power and brings the mission closer to full autonomy. The implementation of these algorithms is described next.

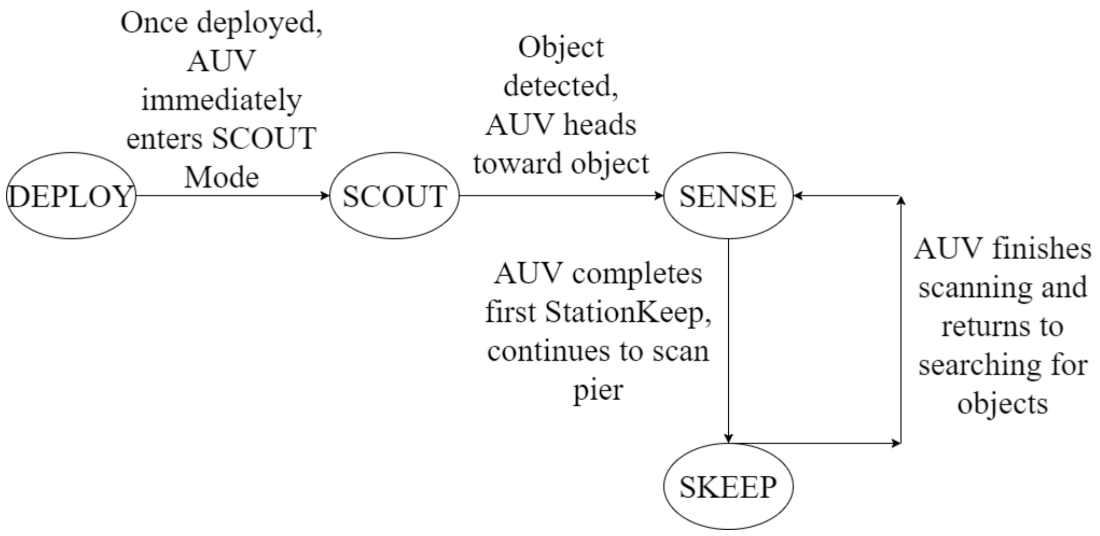
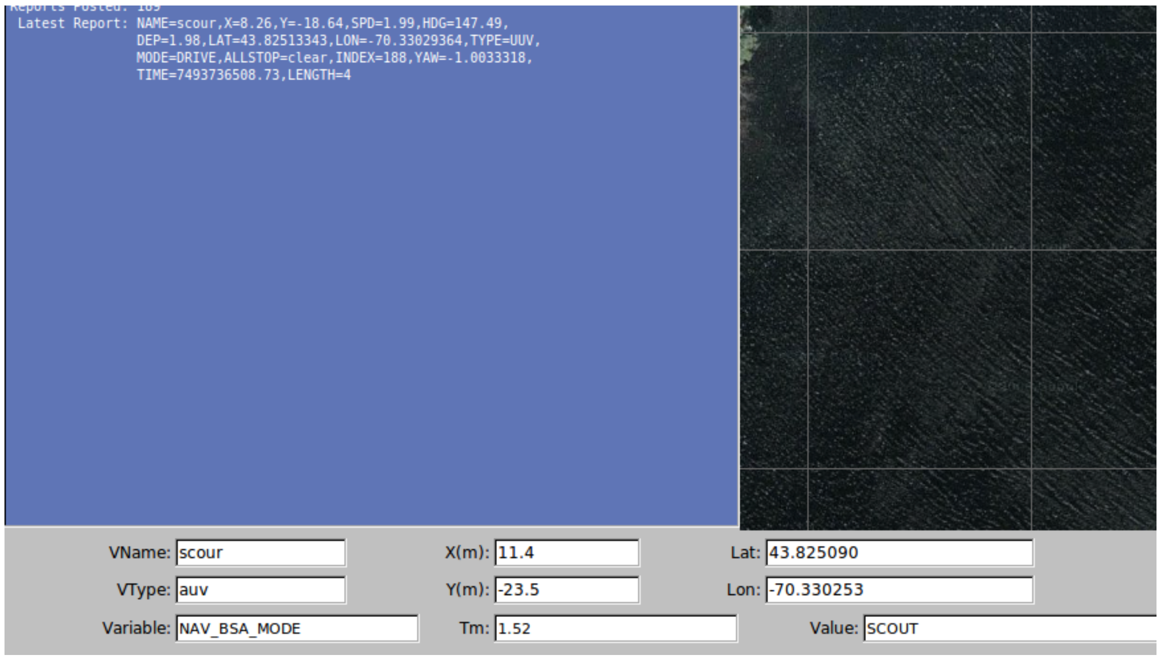

Once deployed, AUV immediately enters SCOUT Mode
DEPLOY
SCOUT
Object detected, AUV heads toward object
SENSE
AUV completes first StationKeep, continues to scan pier
AUV finishes scanning and returns to searching for objects
SKEEP

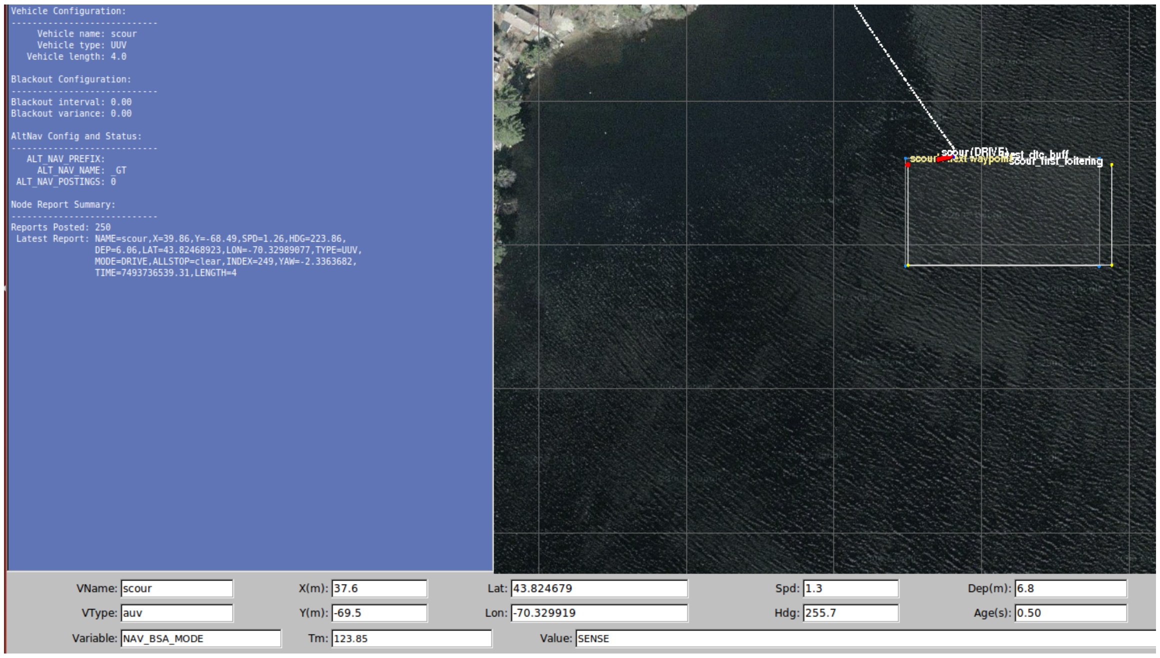
Figure 22. Mode hierarchy in the simulated AUV navigation.



```
Reports Posted: 189
 Latest Report: NAME=scour,X=8.26,Y=-18.64,SPD=1.99,HDG=147.49,
                DEP=1.98,LAT=43.82513343,LON=-70.33029364,TYPE=UUV,
                MODE=DRIVE,ALLSTOP=clear,INDEX=188,YAW=-1.0033318,
                TIME=7493736508.73,LENGTH=4
```

VName:	scour	X(m):	11.4	Lat:	43.825090
VType:	auv	Y(m):	-23.5	Lon:	-70.330253
Variable:	NAV_BSA_MODE	Tm:	1.52	Value:	SCOUT

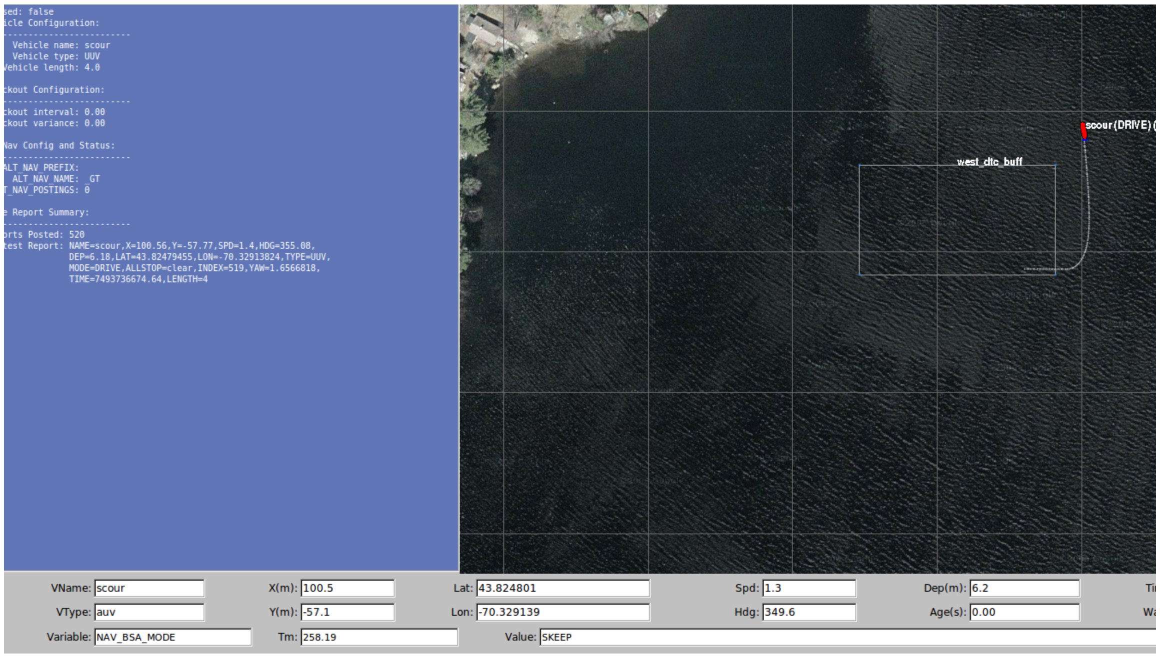
Figure 23. Snapshot of SCOUT Initialization in the simulated AUV navigation.



```
Vehicle Configuration:
----------------------------
     Vehicle name: scour
     Vehicle type: UUV
   Vehicle length: 4.0

Blackout Configuration:
----------------------------
Blackout interval: 0.00
Blackout variance: 0.00

AltNav Config and Status:
----------------------------
   ALT_NAV_PREFIX:
     ALT_NAV_NAME: _GT
 ALT_NAV_POSTINGS: 0

Node Report Summary:
----------------------------
Reports Posted: 250
 Latest Report: NAME=scour,X=39.86,Y=-68.49,SPD=1.26,HDG=223.86,
                DEP=6.06,LAT=43.82468923,LON=-70.32989077,TYPE=UUV,
                MODE=DRIVE,ALLSTOP=clear,INDEX=249,YAW=-2.3363682,
                TIME=7493736539.31,LENGTH=4
```

VName:	scour	X(m):	37.6	Lat:	43.824679	Spd:	1.3	Dep(m):	6.8
VType:	auv	Y(m):	-69.5	Lon:	-70.329919	Hdg:	255.7	Age(s):	0.50
Variable:	NAV_BSA_MODE	Tm:	123.85	Value:	SENSE				

Figure 24. Snapshot of a change to the SENSE mode in the simulated AUV navigation.



```
sed: false
icle Configuration:
-------------------------
  Vehicle name: scour
  Vehicle type: UUV
Vehicle length: 4.0

ckout Configuration:
-------------------------
ckout interval: 0.00
ckout variance: 0.00

Nav Config and Status:
-------------------------
ALT_NAV_PREFIX:
  ALT_NAV_NAME: _GT
T_NAV_POSTINGS: 0

e Report Summary:
-------------------------
orts Posted: 520
test Report: NAME=scour,X=100.56,Y=-57.77,SPD=1.4,HDG=355.08,
             DEP=6.18,LAT=43.82479455,LON=-70.32913824,TYPE=UUV,
             MODE=DRIVE,ALLSTOP=clear,INDEX=519,YAW=1.6566818,
             TIME=7493736674.64,LENGTH=4
```

VName:	scour	X(m):	100.5	Lat:	43.824801	Spd:	1.3	Dep(m):	6.2
VType:	auv	Y(m):	-57.1	Lon:	-70.329139	Hdg:	349.6	Age(s):	0.00
Variable:	NAV_BSA_MODE	Tm:	258.19	Value:	SKEEP				

Figure 25. Snapshot of a change to the SKEEP mode in the simulated AUV navigation.

Implementation of modes: Modes were set in the .bhv file of each behavior. Modes were either set to ACTIVE or DEPLOY so the behaviors began running as soon as the simulation started or they were initialized using the onscreen buttons. This type of mode switching is inefficient and contrary to the types of autonomous decisions the AUV will need to make when surveying for bridge scour.

A new method of mode switching was developed to remedy the above-stated shortcoming. A new variable named m BSA MODE was created that can be added to the MOOSDB publish-subscribe database as seen in Figure 26. Each behavior that requires information about this variable can find it by subscribing to the helm, as shown in Figure 27. The new variable is initialized to ACTIVE. When it is published to the MOOSDB, it is given the name NAV BSA MODE. When the AUV reaches the buffer zone, the mode changes to SENSE. SENSE engages two more behaviors that were previously idle, Constant Speed and Station Keep. Constant Speed subscribes to the MOOSDB for information about the variable m BSA MODE, so when the mode switches, Constant Speed is engaged and the speed decreases. Similarly, the first instance of Station Keep is engaged on SENSE mode. When the AUV completes its first Station Keep waypoint, the mode is again switched. The new mode, SKEEP, activates three more Station Keep points. When the AUV reaches the object and switches to SENSE mode, the AUV will also dive to a lower depth. This new method of mode switching is caused by the AUV reacting to stimuli in its environment. This method is more representative of the decisions the AUV will have to make when completing a mission in real time. The described implementation therefore efficiently simulates autonomous missions of the AUV to investigate bridge pier scour.

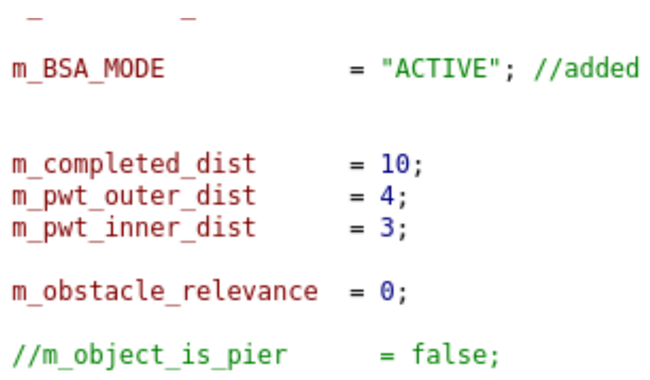
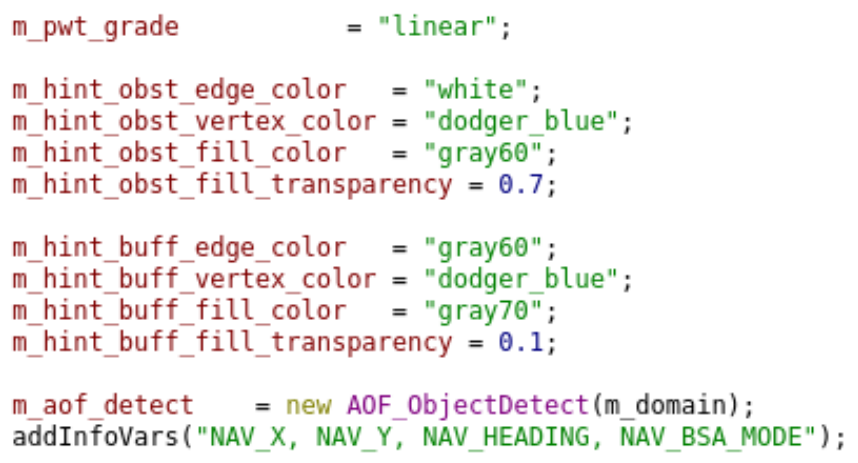

```
m_BSA_MODE           = "ACTIVE"; //added

m_completed_dist     = 10;
m_pwt_outer_dist     = 4;
m_pwt_inner_dist     = 3;

m_obstacle_relevance  = 0;

//m_object_is_pier      = false;
```

Figure 26. New variable created in MOOS-IvP for improved efficiency in AUV decision-making.

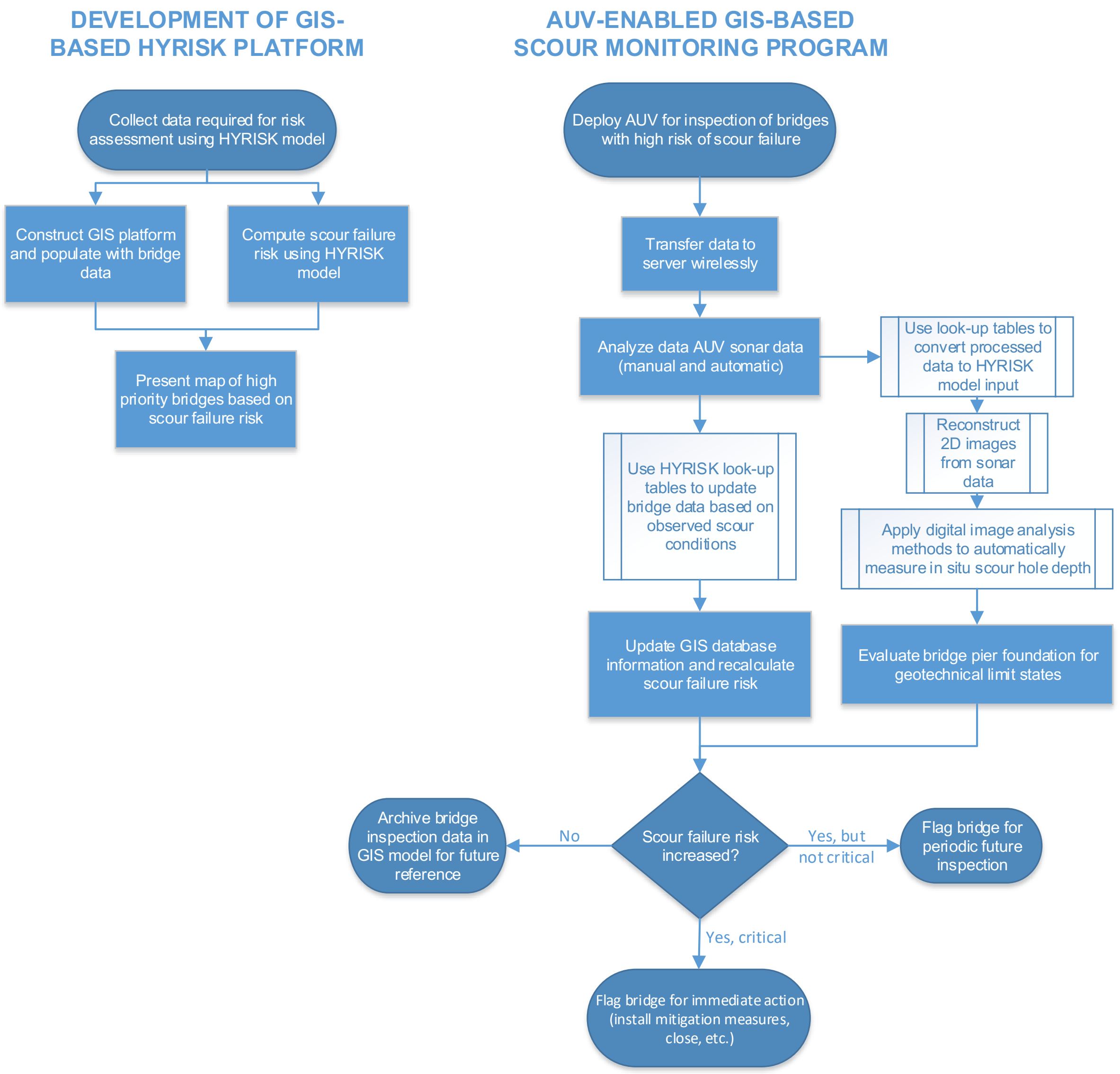


```
m_pwt_grade           = "linear";

m_hint_obst_edge_color   = "white";
m_hint_obst_vertex_color = "dodger_blue";
m_hint_obst_fill_color   = "gray60";
m_hint_obst_fill_transparency = 0.7;

m_hint_buff_edge_color   = "gray60";
m_hint_buff_vertex_color = "dodger_blue";
m_hint_buff_fill_color   = "gray70";
m_hint_buff_fill_transparency = 0.1;

m_aof_detect    = new AOF_ObjectDetect(m_domain);
addInfoVars("NAV_X, NAV_Y, NAV_HEADING, NAV_BSA_MODE");
```

Figure 27. Addition of the variable described in Figure 26 to the Helm.

7. GIS-BASED SCOUR RISK ASSESSMENT FOR PRIORITIZATION OF SCOUR MONITORING

A risk assessment methodology was implemented in this study to prioritize bridges for scour monitoring using the AUV. Once the AUV reaches the deployment stage, the risk assessment described in this section can be used to identify bridges with the highest risk, in dollar amount. The risk assessment methodology adopted was a geographic information system (GIS)-based implementation of the HYRISK (Stein and Sedmera 2006) model for scour failure risk assessment. The GIS-based risk assessment tool can be integrated with data collected by the AUV, as proposed in Figure 28, to update scour risk, particularly following high storm events. In the immediate aftermath of such events, often limited state and local resources need to be prioritized to ensure safety of existing bridge infrastructure. AUVs can be employed to rapidly provide data required for such prioritization efforts. The GIS data platform presented herein will allow immediate and automatic updating of gathered data, and can facilitate complex planning and decision-making processes by incorporating all available data into spatially distributed maps of bridges within the state jurisdiction. Risk maps can automatically prioritize deployment of further monitoring programs as well as scour countermeasures. The developed platform is outlined in the following sections.



DEVELOPMENT OF GIS-BASED HYRISK PLATFORM
Collect data required for risk assessment using HYRISK model
Construct GIS platform and populate with bridge data
Compute scour failure risk using HYRISK model
Present map of high priority bridges based on scour failure risk
AUV-ENABLED GIS-BASED SCOUR MONITORING PROGRAM
Deploy AUV for inspection of bridges with high risk of scour failure
Transfer data to server wirelessly
Analyze data AUV sonar data (manual and automatic)
Use look-up tables to convert processed data to HYRISK model input
Reconstruct 2D images from sonar data
Apply digital image analysis methods to automatically measure in situ scour hole depth
Use HYRISK look-up tables to update bridge data based on observed scour conditions
Update GIS database information and recalculate scour failure risk
Evaluate bridge pier foundation for geotechnical limit states
Scour failure risk increased?
No
Archive bridge inspection data in GIS model for future reference
Yes, but not critical
Flag bridge for periodic future inspection
Yes, critical
Flag bridge for immediate action (install mitigation measures, close, etc.)

Figure 28. Flowchart for implementation of proposed scour monitoring program.

7.1. Development of a GIS data platform for scour risk assessment

The risk assessment method employed in this study was the HYRISK model developed by Pearson et al. (2002) with modifications by Stein and Sedmera (2006) and Khelifa et al. (2013), as well as other modifications to accommodate available data for New York State. Since scour monitoring is often administered by state and local officials, the implementation presented herein can be emulated for other states based on available data. It is important to note that the HYRISK model does not include actual measurement of scour hole depth for bridges. Instead, it relies on auxiliary related information to compute a probability of failure. It then associates a cost with a bridge failure. The product of the probability of failure and cost of failure, along with adjustment factors, define risk of scour failure for a bridge (Stein et al. 1999; Pearson et al. 2002; Stein and Sedmera 2006). The model is therefore a risk ranking system for scour failure. The HYRISK model can be expressed in mathematical form as:

$$Risk = (Risk\ Adjustment\ Factors) \times (Probability\ of\ Failure) \times (Cost\ of\ Failure) \tag{13}$$

The model quantifies the probability of failure by statistical evaluation of historical data for bridges across the united states. The probability of failure is numerically mapped to (1) the likelihood that a given storm event will cause flooding and overtopping of the bridge, or the overtopping frequency, and (2) the vulnerability of a given bridge to scouring of its piers. In order to quantify the probability of failure, these two parameters need to be inferred from NBI data. Cost of failure is quantified by summing the various costs associated with the failure of a bridge in service, as described in the next sections.

The data required for GIS implementation of the modified HYRISK model were collected from a number of different sources. The main source for bridge data was the FHWA (FHWA 1996) National Bridge Inventory (NBI), which is a database of information on all the bridges in the United States. The database stores numeric data on the bridges such as bridge geographical location, dimensions, structural information, use level, traffic capacity, and channel conditions, among others. The database uses a unique classification index that ranks certain bridge properties such as scour vulnerability and channel protection based on a 9-0 scale with 9 assigned to the best condition and 0 assigned to the poorest condition (the letter N is assigned if a given classification does not apply to a bridge) (FHWA 1996). The HYRISK model uses the NBI data to create correlations between a given bridge’s available data to various unmeasurable information, relying on statistics to adequately determine probabilities.

7.1.1. Probability of failure

The overtopping frequency is related to two NBI items: functional class (NBI item 26) and Waterway adequacy (NBI item 71). Functional class refers to the type of roadway that the bridge supports and is a function of the location of the bridge and correlates to factors such as the ADT and the travel speed along the roadway. Waterway adequacy appraises the waterway opening with respect to flow conditions under the bridge, and is ranked from zero through nine. The resulting overtopping frequency is categorized into five groups as never (N), remote (R), slight (S), occasional (O), and frequent (F) overtopping. For example, an interstate highway with a water adequacy number of 9 (best waterway conditions under bridge) results in an overtopping frequency rating of remote. Overtopping frequency can therefore be programmed to be computed for each bridge using the aforementioned parameters.

Vulnerability is a measure of how resilient a bridge is against scouring. The parameter has a rank of zero through nine for each bridge in the HYRISK model. The two NBI items used to quantify vulnerability are channel protection (NBI item 61) and substructure condition (NBI item 60). Channel protection describes the nature and quality of protective measures in the riverbed and bridge pier against scouring due to water flow. Elements such as stream stability and the condition of the channel, riprap, slope protection, or stream control devices are quantified into a channel protection rank (0 through 9). The substructure condition describes the condition of the pier, piles, or other types of support or foundation. As expected, a channel or substructure in worse condition will result in a more scour vulnerable foundation. For example, according to the HYRISK model, a bridge with a substructure condition rank of 9 and a channel protection rank of 9 also has a vulnerability of 9. Vulnerability can therefore also be programmed to be computed for each bridge using NBI items 60 and 61.

Once the overtopping frequency and vulnerability factors have been quantified for a given bridge, its probability of scour failure can be determined using a mapping table, as shown in Table 5. The table presented by Stein et al. (2006) based on statistical analysis of scour failure data, with minor subsequent adjustments by Khelifa et al. (2013).

Table 5: Probability of failure based on overtopping frequency and scour vulnerability (after Stein and Sedmera 2006, with modifications by Khelifa et al. 2013).

		Overtopping Frequency			
		Remote	**Slight**	**Occasional**	**Frequent**
Scour Vulnerability	0	1	1	1	1
	1	0.01	0.01	0.01	0.01
	2	0.005	0.006	0.008	0.009
	3	0.0011	.0013	0.0016	0.002
	4	0.0004	0.0005	0.0006	0.0007
	5	0.0003	0.0004	0.0005	0.0006
	6	0.00018	0.00025	0.0004	0.0005
	7	0.00018	0.00025	0.0004	0.0005
	8	0.000004	0.000005	0.00002	0.00004
	9	0.0000025	0.000003	0.000004	0.000007
	N	0	0	0	0

7.1.2. Cost of failure

The cost associated with failure of a bridge due to scouring is the sum of four costs: cost of loss of life, rebuilding cost, running cost, and time loss cost, described next.

***Rebuilding cost*:** The rebuilding cost is the cost involved in rebuilding the bridge after failure. This function of the risk can be determined using Eq. (14) and parameters developed from the HYRISK model, which computes running cost as

$$\textit{Rebuilding Cost} = C_1 \times e \times W \times L \tag{14}$$

where C_1 is the cost associated with reconstructing the bridge (in units of \$/ft^2), e is an early replacement cost multiplier, W is the width of the bridge (NBI Item 52, in units of ft), and L is the length of the bridge (NBI Item 49, in units of ft). The early replacement cost is a function of average daily traffic (NBI item 29) and ranges between one and two as per Stein and Sedmera (2006).

The reconstruction costs (C_1) were determined for New York State bridges using preliminary cost estimating worksheets for new and replacement bridges published by the New York Department of Transportation (NYDOT). The worksheet contains a variety of information on each contract such as the cost estimate and includes the structure and superstructure type. Several adjustments were made to the worksheet values to ensure reasonable estimates for reconstruction costs. For example, the values in the worksheet were reduced, based on engineering judgement and experience, for continuous span concrete bridges, as the worksheet yielded excessively high values. Moreover, the NBI data includes older bridges with material types not listed in the worksheet, including timber, masonry, and aluminum. It was assumed in these cases that if a bridge were to fail, the new design would be according to local practice. Therefore, reconstruction costs were computed for materials listed in Table 6. For reference, reconstruction costs reported by Stein and Sedmera (2006) are also listed in Table 6.

Table 6. Bridge reconstruction cost

NBI Value	Material	Stein and Sedmera (2006) cost ($/ft^2)	Cost used in this study for New York ($/ft^2)
1	Concrete	50-65	515
2	Concrete, continuous span	60-80	515
3	Steel	62-75	460
4	Steel, continuous span	70-90	460
5	Prestressed	50-70	328
6	Prestressed, continuous span	65-110	328
7	Wood	N/A	515
8	Masonry	N/A	515
9	Aluminum, Iron	N/A	460
0	Other	N/A	515

***Running cost*:** The running cost can be defined as the cost associated with keeping a constant vehicular flow rate. It is calculated by splitting vehicular traffic into passenger vehicles and trucks, and estimating running cost for alternative routes while the bridge is being reconstructed. Running cost can be written as:

$$\text{Running Cost}=\left[C_2\left(1-\frac{T}{100}\right)+C_3\left(\frac{T}{100}\right)\right]\times D\times A\times d$$
$$\text{Time Loss Cost}=\left[C_4\times O\times\left(1-\frac{T}{100}\right)+C_5\times\left(\frac{T}{100}\right)\right]\times\frac{D\times A\times d}{S} \tag{15}$$

where C_2 and C_3 are cost of running passenger vehicles and trucks, respectively, A is the length of the detour (NBI Item 19), A is the average daily vehicular traffic (NBI Item 29), T is the average daily truck traffic (NBI Item 109), and d is the duration of the detour. An average vehicular cost, C_2, of \$0.535/mile was used based on analysis of the Internal Revenue Service data, and the value of C_3 was chosen as \$2.12/mile based on data obtained from DAT ReteReview, which allows users to obtain current contract freight rates across the country[3].

The duration of the detour (d) is not a factor that can be found through the NBI data. Pearson et al. (2002) developed a modified version of the HYRISK model that estimates the average time that a motorist may spend being detoured. The product of this estimate and the ADT of the bridge yields the total time spent on the detour.

Time loss cost**:** The time loss cost in the cost equation can be summarized as the cost associated with the time lost from a bridge failure and a driver being forced into an alternative, longer route. Unlike the running cost which is a value based on the vehicle itself, the time loss cost is based upon the monetary value of the driver's time. Similar to the running cost equation, the time loss cost equation uses two constants, C_4 and C_5, to represent the value of time for passenger car and truck drivers, respectively. The equation can be written as follows:

$$\text{Time Loss Cost}=\left[C_4\times O\times\left(1-\frac{T}{100}\right)+C_5\times\left(\frac{T}{100}\right)\right]\times\frac{D\times A\times d}{S} \tag{16}$$

where S is the average detour speed, O is the occupancy rate for a passenger car, and the rest of the parameters are similar to Eq. (15). The occupancy rate for a passenger car was taken from the National Household Travel Survey to be 1.55 and the detour speed to be 40 mph. Based on

[3] https://www.dat.com/freight-rates

analysis of New York State Department of Labor data, an average value of time of $12.82/hour was selected.

***Cost of life*:** The final term in the total cost equation is the cost of life, which represents any legal cost or compensation associated with a life lost due to a bridge failure. This cost is the product of the cost of each life lost, C_6, and the expected number of deaths, X. The expected number of deaths was estimated using the equation proposed by Khelifa et al. (2013), as the product of three terms: the occupancy rate for a bridge, O, the time to clear a bridge, TC, and the arrival rate of the bridge, AR. The time to clear the bridge was computed using the length of the bridge (NBI item 49) and the average speed limits in New York State[4]. Speed limits were correlated to the functional classification of bridges (NBI item 26) to reflect varying speed limit across different roads. The arrival rate is the ADT for a given bridge.

***Risk adjustment factors*:** The HYRISK model incorporated a risk adjustment factor, K, to account for uncertainties resulting from unknown bridges. Accordingly, $K=K_1 \times K_2$, where K_1 is a factor relating to the length of the bridge, and K_2 is the foundation adjustment factor, ranging from 0.2 for massive rock foundations and one for unknown foundations. In the present study, the uncertainties in the bridge foundation were ignored, with a value of one assigned for both K_1 and K_2.

7.2. Risk calculation for New York State bridges

The first step in GIS implementation of the HYRISK model is to collect available data for New York State bridges from NBI. The data was downloaded from FHWA into a spreadsheet for further analysis. Not all the available data from NBI is needed in the HYRISK model. Bridges which did not have scour susceptibility were eliminated from the dataset. A list of bridge deletions used in this study is presented in Table 7. It can be seen that culverts, interchanges, bicycle bridges, and bridges for which NBI has determined scour susceptibility is not applicable were removed from the database. Of the 17,461 total bridges in the 2016 NBI database, a total of

[4] http://www.speed-limits.com/newyork.htm

7,261 bridges were removed, leaving 10,200 bridges for risk analysis and implementation in the GIS database.

Table 7. A table sowing list of bridges deleted from the dataset for this study

Bridge removed	NBI item and definition	Reason for deletion	Notes
Culvert	60: Bridge substructure	Not susceptible to the type of scour that is the focus of this study, *i.e.*, local scour.	Bridges with a code of N in item 60 correspond to culverts.
Interchange	42A: Type of service 42B: Type of structure under bridge	Water not flowing under bridge	Bridges with code 6 for item 42A are interchanges, and are removed. Bridges with codes 1-4 under item 42B removed for same reason.
Bicycle bridge	42A: Type of service	Not focus of this study	Code 3 in item 42A indicates bicycle bridge.
Not susceptible to scour as per NBI	113: scour criticality	Bridge is removed if it is deemed not susceptible to scour in NBI.	Code N for NBI item 113 indicates bridge is not susceptible to scour.

Once the data were collected in a spreadsheet format, required calculations were made to compute risk for each bridge. The resulting risk parameter ranged between zero and one. The data were transferred to the GIS software ARCGIS Pro, and was superimposed on New York State map delineating county borders. Each bridge was tagged on the map using its longitude and latitude information. All subsequent calculations and other relevant information and data such as images, scour inspection logs and reports can be tagged uniquely to each bridge on the map as needed. The focus of the present study was to visualize risk of scour failure for bridges across New York. The resulting map is presented in the next section.

7.3. Risk maps for New York State bridges

The procedures outlined in previous sections were used to compute risk of scour failure for the 10,200 bridges considered in New York State. From the bridges investigated, 30 yielded a probability of overtopping of 0 indicating an excellent condition for the bridge. The majority of bridges had an annual risk less than $10,000. The risk scores can be used to perform a range of analyses, including ranking of bridges, based on risk, for prioritization of post-storm inspections, for distribution of state funds among counties and localities based on scour failure risk, and for planning routes for inspection and investigation of scour conditions.

Results of the risk analysis performed following methods described in this section are presented in Figure 29. The map shows mean risk for all counties in New York state. The highest risk was computed for Kings county, while multiple other counties in upstate New York also posed high risks. A plot of total risk of each county is also presented in Figure 30, showing a readjustment of county rankings with total risk. This adjustment reflects the different total number of bridges within counties, and can have important implications for planning and distribution of funds for scour monitoring and mitigation programs.

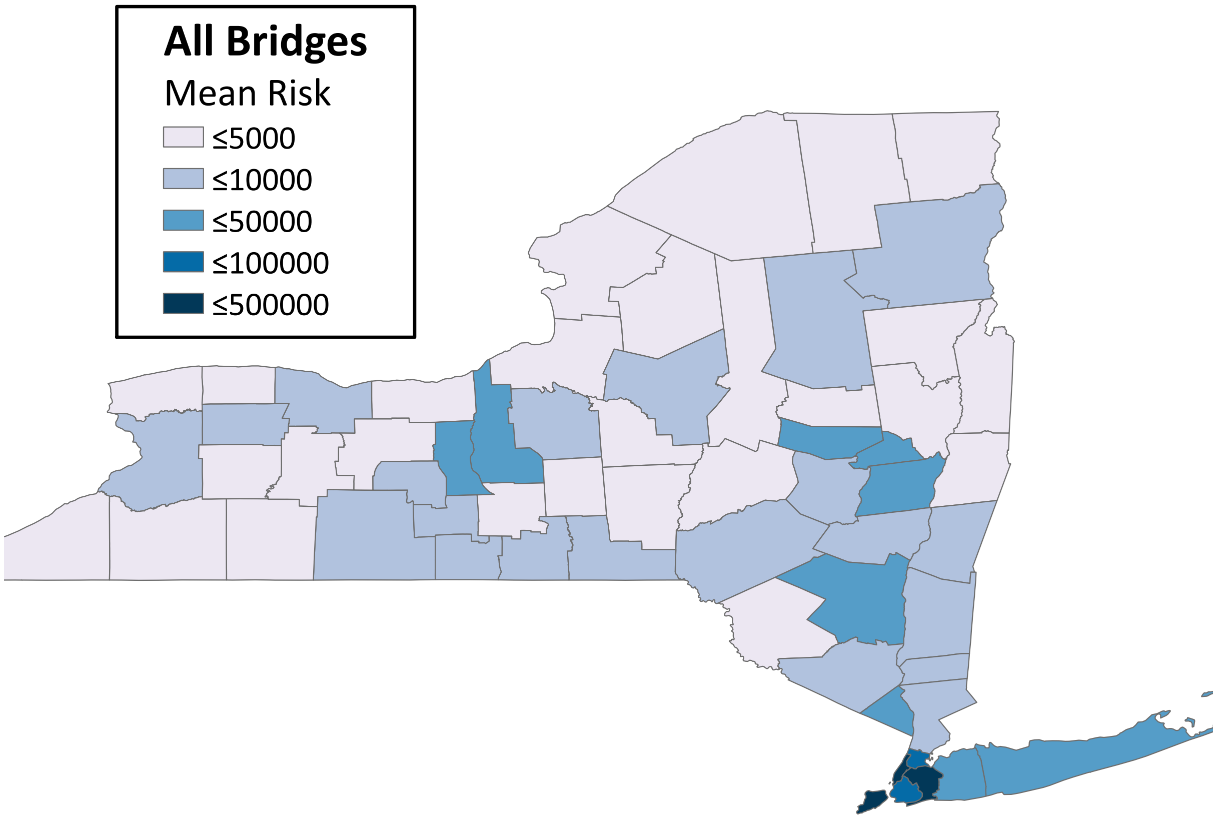


Figure 29. Mean risk map of all bridges in New York.

A point plot of one hundred bridges with the highest risk of scour failure in the state are shown in Figure 31. Each dot is color-coded to show the level of risk. The total risk for each bridge was normalized by the total risk of the bridge with the highest risk. Therefore, a normalized risk of one means the bridge has the highest risk in the state, and all other bridges have a normalized risk less than one, and relative to the highest risk. The normalized risk was divided into five categories, based on statistical analysis of the total risk range for the top one hundred bridges. These categories can be adjusted to reflect traditional classifications such as scour-criticality. Among the bridges considered, one bridge had a normalized risk score greater than 0.5, seven bridges had a normalized score between 0.1 and 0.5, and 24 bridges had a normalized risk score between 0.05 and 0.1. It is important to note that these rankings are to be used for prioritization and planning purposes; the low normalized risk score does not indicate low global risk of scour failure. The inset in Figure 31 shows a close-up of the Kings county area, revealing the significant number of bridges with moderate to high risk in this area.

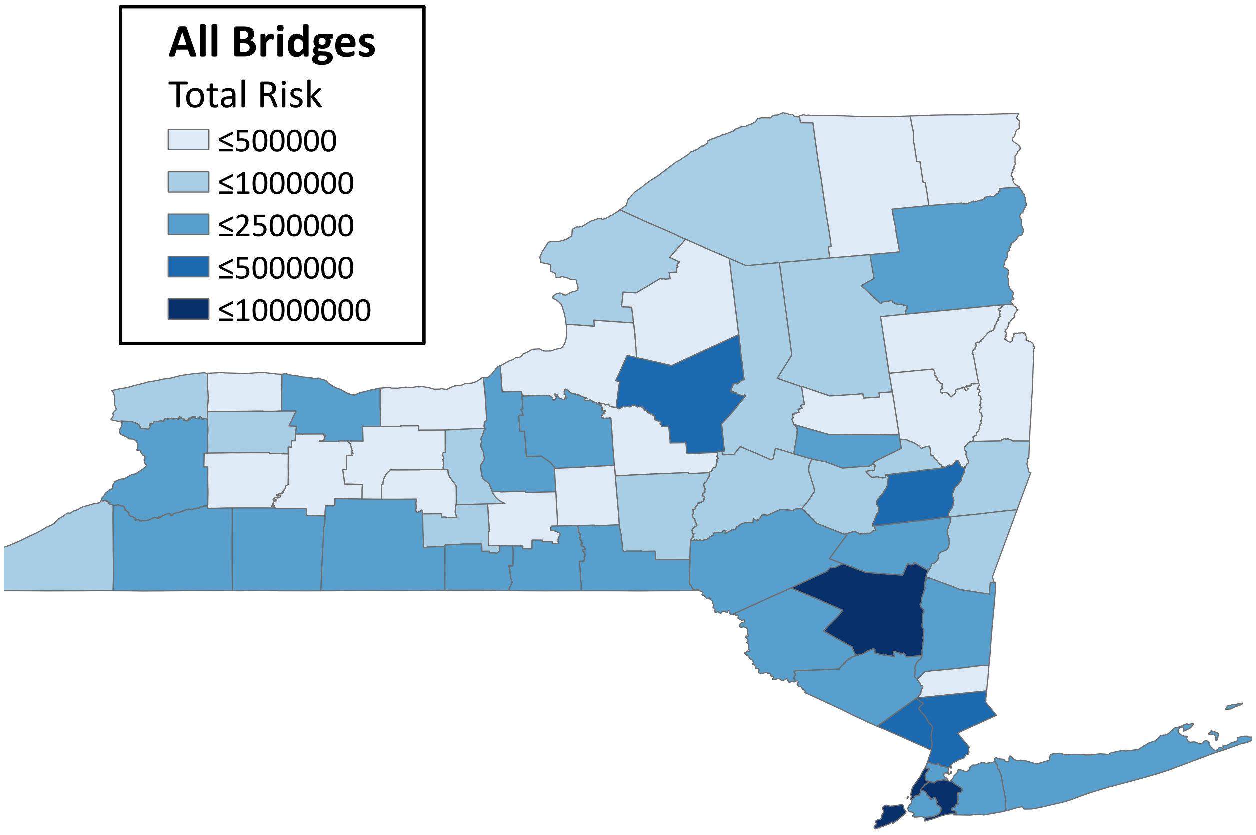


Figure 30. Total risk map of all bridges in New York.

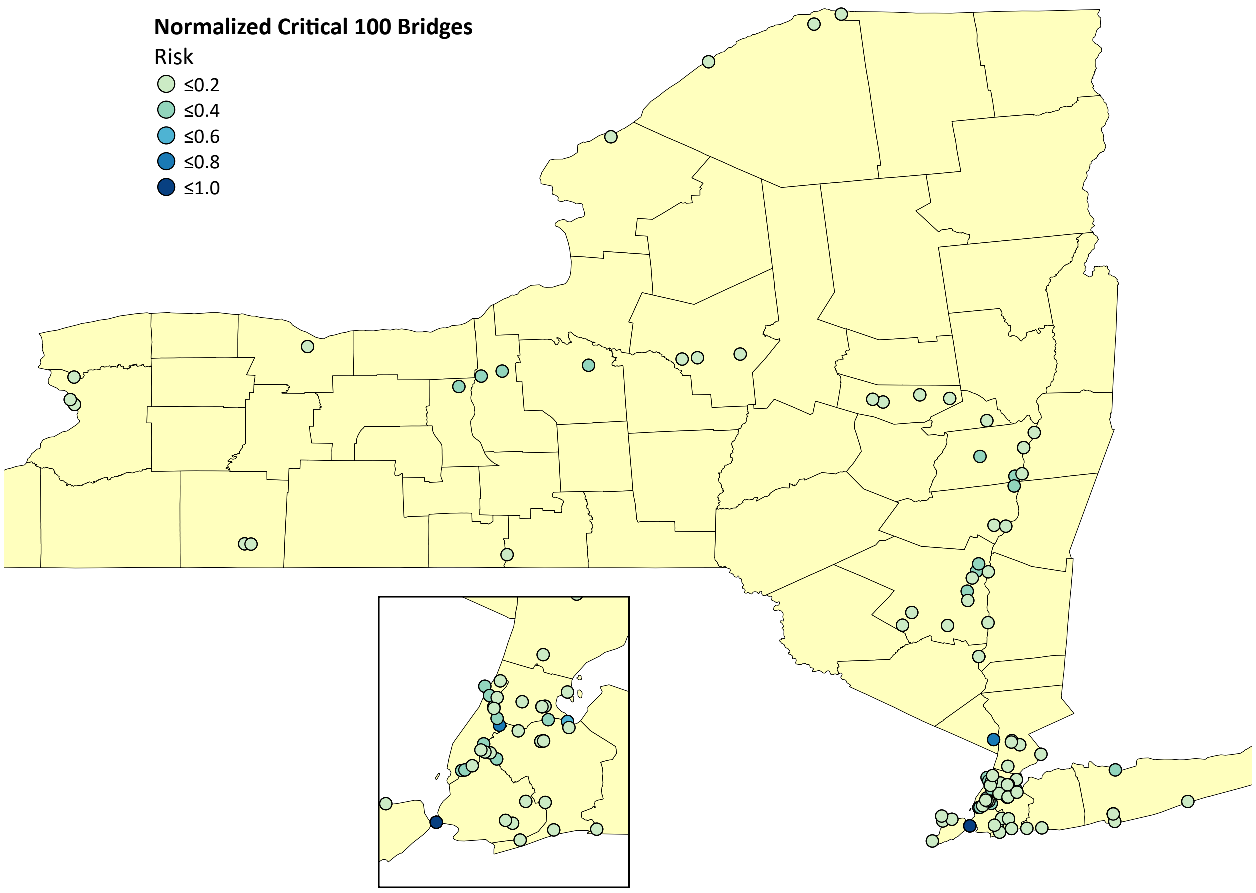


Figure 31. Normalized risk map of top 100 bridges with highest risk of scour failure in New York.

8. CONCLUSIONS AND FUTURE DIRECTIONS

In this study, technological advances were made in terms of hardware and software to facilitate a geographic information system (GIS)-based scour risk assessment using autonomous underwater vehicles (AUV). The contributions of the work included:

(1) Adaptation of an existing AUV for bridge scour monitoring and inspection. The AUV adaptations included hardware upgrades, such as improvements on fabrication methods, upgrades on motor components for navigation in riverine environments, navigation, onboard processors, and instrumentation to accommodate collection of bathymetric data from a bridge site.

(2) Development of codes and algorithms to guide navigation of the AUV, and to process typical digital images collected in AUV missions at bridge pier sites. Specifically, codes were developed to analyze typical acoustic images of bridge piers for extraction of features of interest to scour monitoring, including the bridge pier structure and the riverbed outline. Methods developed included preprocessing using the Savitsky-Golay filter and entropy and range filtering, edge detection algorithms using the Prewitt operator, the Gabor filter, and K-means clustering, and the use of Hough transform. The algorithms were implemented in Matlab and OpenCV. Details of the image processing algorithms, the underlying theory, and results applied to a set of sample acoustic images were presented.

(3) Simulation of AUV path finding and navigation using the state-of-the-art Mission Oriented Operating Suite (MOOS) simulation environment, which is a multi-objective optimization system based on interval programming. A navigation algorithm was programmed to allow for typical AUV scour monitoring missions. The platform can be used to program autonomous missions for the AUV in future work.

(4) Development of a GIS-based platform to prioritize bridge scour monitoring and inspection programs using AUVs and other methods. A risk assessment model based on the HYRISK model by Stein et al. (2006), and extended by Khelifa et al. (2013), was implemented by computing probability of failure and cost of failure for over 10,200 bridges in New York State for which scour susceptibility was applicable. Risk of scour-related failure was defined as the product of the probability of failure and the cost of failure, along with several risk adjustment factors. Data from the National Bridge Inventory (NBI) was used to compile a database of bridge properties relevant to the HYRISK model. The model used several NBI items to quantify the probability of failure for a given bridge using overtopping frequency and scour vulnerability interpretations from the NBI data. The NBI items used to compute the probability of failure included the functional class (NBI item 26), waterway adequacy (NBI item 71), scour vulnerability (NBI item 61) and substructure conditions (NBI item 60). Cost of failure was quantified for each bridge as the product of the rebuilding cost, the running cost, time loss cost, and the cost of life. All relevant data was compiled in a GIS map

of New York State, using the software ArcGIS. A set of risk maps were generated to demonstrate the efficacy of the developed model in visualizing risk distribution throughout the state, which can be useful in decision-making and planning for post-storm prioritization of AUV deployment for scour assessment and other mitigative actions.

While the majority of the objectives of the study were achieved, setbacks in development f the AUV prevented field deployment during the course of this study. In future work, the AUV will be tested in riverine environments to assess its capabilities in autonomous navigation, acoustic image acquisition, and transfer of information for further scour analysis. Once the AUV is tested for autonomy, scour assessment missions will be carried out to test the efficacy of the image processing algorithms developed as part of this study. The OpenCV implementation of the codes will be used for onboard processing of the acquired images. MOOS simulations will be used to aid autonomous missions. Once missions are completed, the information can be relayed to a central server, where the relevant scour data including images of the pier, scour conditions, riverbed parameters, bathymetric data, and scour hole depth can be tagged to the developed GIS model. In future work, this information can be used to automatically update the GIS-based HYRISK risk assessment model. Risk maps can therefore be automatically updated following AUV scour assessment missions. Continuous deployment of AUVs throughout the state may be a cost-effective means of monitoring scour conditions in the state's bridges, particularly in the wake of extreme weather events.

REFERENCES

Antonelli G. *Underwater Robots*. Springer. Third Edition, 2014.

Arthur D, Vassilvitskii S. K-means++: The Advantages of careful seeding. Online publication: http://ilpubs.stanford.edu:8090/778/1/2006-13.pdf, Stanford University.

Arthur D, Vassilvitskii S. K-means++: the advantages of careful seeding. Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms. Society for Industrial and Applied Mathematics, New Orleans, Louisiana, January 7-9, 2007, pp. 1027-1035.

Atherton MW. *Echoes and Images: The Encyclopedia of Side-Scan and Scanning Sonar Operations.* Vancouver, BC: OysterInk Oublications, 2011.

Benjamin MR, Newman PM, Schmidt H, Leonard JJ. An Overview of MOOS-IvP and a Brief Users Guide to the IvP Helm Autonomy Software. MIT Computer Science and Artificial Intelligence Lab, MIT-CSAIL-TR-2009-028, June, 2009.

Benjamin MR. MOOS-IvP autonomy tools users manual. Release 13.2, MIT Computer Science and Artificial Intelligence Lab, www.moos-ivp.org/docs/, February, 2013.

Chen K. On coresets for k-median and k-means clustering in metric and Euclidean spaces and their applications. *SIAM Journal on Computing* 2009; 39(3):923-947.

Clausi D, Jernigan E. Designing Gabor filters for optimal texture separability. *Pattern Recognition* 2000; 33:1835-1849. 10.1016/S0031-3203(99)00181-8.

Daugman J. 2018. Information Theory Lecture Series. University of Cambridge. [Link: https://www.cl.cam.ac.uk/teaching/1718/InfoTheory/materials.html. Last accessed August 20, 2018.]

Dim JR, Takamura T. 2013. Alternative approach for satellite cloud classification: edge gradient application. *Advances in Meteorology*; 2013: 1–8. ISSN 1687-9309. doi:10.1155/2013/584816.

Duda RO, Hart PE. Use of the Hough Transformation to Detect Lines and Curves in Pictures, *Communications of the ACM* 1972; Vol. 15, pp. 11–15.

El-Sayed MA. A new algorithm based entropic threshold for edge detection in images. *International Journal of Intelligent Computing and Information Science* 2011; 8(5-1):71-78.

El-Sayed, MA, Ahmed MAO. Using Renyi's entropy for edge detection in level images. *International Journal of Intelligent Computing and Information Science* 2011; 11(2): 1-10.

Ferguson TS. An inconsistent maximum likelihood estimate. J*ournal of the American Statistical Association* 1982; 77(380):831–834.

FHWA. Recording and Coding Guide for the Structure Inventory and Appraisal of the Nation's Bridges. Report No. FHWA-PD-96-001, 1996.

Fleming WGK. *Piling Engineering*, Taylor and Francis, 2009.

Gander W, Matt U. *Solving Problems in Scientific Computing Using Maple and MATLAB*. Springer. 1997, 412 p.

Gee, K. W. Action: National Bridge Inspection Standards- Scour Evaluations and Plans of Action for Scour Critical Bridges. Federal Highway Administration, Washington, DC; 2008.

Ghaemi R, Sulaiman MN, Ibrahim H, Mustapha N. A Survey: clustering ensembles techniques. Proceedings of World Academy of Science, Engineering and Technology 2009; 38: 2070-3740.

Gilmour B, Niccum G, O'Donnell T. Field resident AUV systems- Schevron's long-term goal for AUV development, in Autonomous Underwater Vehicles (AUV), 2012 IEEE/OES, 2012: 1–5.

Hart PE. How the Hough Transform was Invented. *IEEE Signal Processing Magazine* 2009; 26(6): pp 18 – 22.

Hunt BE. Monitoring scour critical bridges. NCHRP Synthesis of Highway Practice 396, Transportation Research Board, Washington, DC, 2009.

Jain AK, Farrokhnia F. Unsupervised texture segmentation using Gabor filters. *Pattern Recognition* 1991; 24(12): 1167-1186.

Jeynes ET. Information Theory and Statistical Mechanics, *Physical Review* 1957; 106(4): 620-630.

Johnson PA, Whittington RM. Vulnerability-Based Risk Assessment for Stream Instability at Bridges. *Journal of Hydraulic Engineering* 2011; 137(10): 1248-1256.

Khelifa A, Garrow LA, Higgins MJ, Meyer MD. Impacts of Climate Change on Scour-Vulnerable Bridges: Assessment Based on HYRISK. *Journal of Infrastructure Systems* 2013; 19(2): 138-146.

Kintigh KW, Ammerman A. Heuristic approaches to spatial analysis in archaeology. *American Antiquity* 1982; 47:31-63.

Kumar P, Wasan SK. Comparative analysis of k-mean based algorithms. *International Journal of Computer Science and Network Security* 2010; 10(4):314 -318.

Lagasse PF, Ghosn M, Johnson PA, Zevenbergen LW, Clopper PE. Reference Guide for Applying Risk and Reliability-Based Approaches for Bridge Scour Prediction. NCHRP REPORT 761, Transportation Research Board, Washington, DC, 2013.

Lee TS. Image representation using 2D Gabor wavelets. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 1996; 18(10).

Leung CK, Lam FK. Performance analysis for a class of iterative image thresholding algorithms. *Pattern Recognition* 1996; 29(9): 1523-1530.

Malik J, Perona P. Preattentive texture discrimination with early vision mechanisms. *Journal of Optical Society of America A* 1990; 7(5):923-933.

Malik J, Perona P. Preattentive texture discrimination with early vision mechanisms. *Journal of the Optical Society of America* 1990; 7(5):923-933.

Materka A, Strzelecki M. Texture Analysis Methods – A Review, technical University of Lodz, Institute of Electronics, COST b11 report, Brussels, 1998.

Mathworks. 2017. *Matlab Documentation*.

Mehala N, Dahiya R. Comparative study of FFT, STFT and wavelet techniques for induction machine fault diagnostic analysis. In: Proc. of the 7th WSEAS Int. Conf. on Computational Intelligence, Man-Machine Systems and Cybernetics (CIMMACS '08), Cairo, Egypt, December 29-31, 2008.

Melville, B. W., and Coleman, S. E. Bridge Scour. Water Resources Publications. Highlands Ranch, Co.; 2000.

Ortega J, Del Rocío M, Rojas B, Somodevilla García M. Research issues on K-means algorithm: an experimental trial using Matlab. CEUR Workshop Proceedings, 534:83-96, 2009.

Pearson D, Stein S, Jones JS. HYRISK Methodology and Users Guide. Report FHWA-RD-02-XXX. Federal Highway Administration, Washington, D.C, 2002.

Peeples MA. R script for k-means cluster analysis. [online]. Available at: http://www.mattpeeples.net/kmeans.html. (Last accessed September 9, 2017).

Prasad VN, Domke J. Gabor Filter Visualization. University of Maryland Internal Report, 2017.

Prendergast LJ and Gavin K. A review of bridge scour monitoring techniques. *Journal of Rock Mechanics and Geotechnical Engineering* 2014; 6:138-149.

Press WH, Teukolsky SA, Vetterling WT, Flannery BP. *Numerical Recipes in Fortran 77: the Art of Scientific Computing*. Second Edition, Cambridge University Press. 1997.

Prewitt JMS. Object enhancement and extraction. *Picture Processing and Psychopictorics* 1970; 10(1): 15–19.

Reed IV TB, Hussong D. Digital image processing techniques for enhancement and classification of SeaMARC II side scan sonar imagery. Jou*rnal of Geophysical Resources* 1989; 94(B6):7469–7490. doi:10.1029/JB094iB06p07469.

Schaffer, RW. What Is a Savitzky-Golay Filter?, *IEEE Signal Processing Magazine*, 2011; *Digital Object Identifier* 10.1109/MSP.2011.941097, pp 111- 117.

Schall JD and Price GR. Portable scour monitoring equipment. NCHRP Report 515, Transportation Research Board, Washington, DC, 2004.

Sørensen AJ and Ludvigsen M. Towards Integrated Autonomous Underwater Operations. Plenary lecture. IFAC Workshop on Navigation, Guidance, and Control of Underwater Vehicles, April 28-30, 2015. Girona, Spain, IFAC-PapersOnLine 12/2015; 48(2):107-118. DOI: 10.1016/j.ifacol.2015.06.018.

Sørensen AJ and Ludvigsen M. Towards Integrated Autonomous Underwater Operations. Plenary lecture. IFAC Workshop on Navigation, Guidance, and Control of Underwater Vehicles, Girona, Spain, IFAC 48(2):107-118. DOI: April 28-30, 2015.

Stein SM, Young GK, Trent RE, Pearson DR. Prioritizing scour vulnerable bridges using risk. *Journal of Infrastructure Systems* 1999; 5(3): 95-101.

Stein, S. and Sedmera, K. Risk-Based Management Guidelines for Scour at Bridges with Unknown Foundations. NCHRP Project 24–25 Final Report; 2006.

Sunaga T. Theory of interval algebra and its application to numerical analysis. In: Research Association of Applied Geometry (RAAG) Memoirs, Ggujutsu Bunken Fukuy-kai. Tokyo, Japan, 1958, 2:547-564. *Reprinted in Japan Journal on Industrial and Applied Mathematics*, 2009, 26(2-3):126–143.

APPENDIX A: OPENCV IMPLEMENTATION OF MATLAB CODES DEVELOPED IN THIS STUDY

Listing 1: Implementation of Entropy in OpenCV:

```
#include "opencv2/highgui.hpp"
#include "opencv2/imgproc.hpp"
#include <iostream>
#include <stdio.h>
#include <time.h>
using namespace std;
using namespace cv;
static int sub_to_ind(int *coords, int *cumprod, int num_dims)
{
    int index = 0;
    int k;

    assert(coords != NULL);
    assert(cumprod != NULL);
    assert(num_dims > 0);

    for (k = 0; k < num_dims; k++)
    {
        index += coords[k] * cumprod[k];
    }

    return index;
}

static void ind_to_sub(int p, int num_dims, const int size[],
    int *cumprod, int *coords)
{
    int j;

    assert(num_dims > 0);
    assert(coords != NULL);
    assert(cumprod != NULL);

    for (j = num_dims - 1; j >= 0; j--)
    {
        coords[j] = p / cumprod[j];
        p = p % cumprod[j];
    }
}

void getLocalEntropyImage(cv::Mat &gray, cv::Rect &roi, cv::Mat &entropy)
{

    clock_t func_begin, func_end;
    func_begin = clock();
    //1.define neighborhood model,here it's 9*9
    int neighborhood_dim = 2;
    int neighborhood_size[] = {9, 9};
    //2.Pad gray_src
    Mat gray_src_mat(gray);
    Mat pad_mat;
    int left = (neighborhood_size[0] - 1) / 2;
    int right = left;
    int top = (neighborhood_size[1] - 1) / 2;
    int bottom = top;
    copyMakeBorder(gray_src_mat,   pad_mat,   top,   bottom,   left,   right,
BORDER_REPLICATE, 0);
    Mat *pad_src = &pad_mat;
    roi = cv::Rect(roi.x + top, roi.y + left, roi.width, roi.height);

    //3.initial neighborhood object,reference to Matlab build-in neighborhood
object system
    //        int element_num = roi_rect.area();
    //here,implement  a  histogram  by  ourself  ,each  bin  calcalate  gray  value
frequence
    int hist_count[256] = {0};
    int neighborhood_num = 1;
    for (int i = 0; i < neighborhood_dim; i++)
        neighborhood_num *= neighbood_size[i];

    //neighborhood_corrds_array is a neighbors_num-by-neighborhood_dim array
containing relative offsets
    int            *neighborhood_corrds_array             =             (int
*)malloc(sizeof(int)*neighborhood_num * neighborhood_dim);
    //Contains  the  cumulative  product  of  the  image_size  array;used  in  the
sub_to_ind and ind_to_sub calculations.
    int *cumprod = (int *)malloc(neighborhood_dim * sizeof(*cumprod));
    cumprod[0] = 1;
    for (int i = 1; i < neighborhood_dim; i++)
        cumprod[i] = cumprod[i - 1] * neighborhood_size[i - 1];
    int *image_cumprod = (int*)malloc(2 * sizeof(*image_cumprod));
    image_cumprod[0] = 1;
    image_cumprod[1] = pad_src->cols;
    //initialize neighborhood_corrds_array
    int p;
    int q;
    int *coords;
    for (p = 0; p < neighborhood_num; p++){
        coords = neighborhood_corrds_array + p * neighborhood_dim;
        ind_to_sub(p, neighborhood_dim, neighborhood_size, cumprod, coords);
        for (q = 0; q < neighborhood_dim; q++)
            coords[q] -= (neighborhood_size[q] - 1) / 2;
    }
    //initlalize neighborhood_offset in use of neighbornood_corrds_array
    int *neighborhood_offset = (int *)malloc(sizeof(int) * neighborhood_num);
    int *elem;
    for (int i = 0; i < neighborhood_num; i++){
        elem = neighborhood_corrds_array + i * neighbood_dim;
        neighborhood_offset[i] = sub_to_ind(elem, image_cumprod, 2);
    }

    //4.calculate entropy for pixel
    uchar *array = (uchar *)pad_src->data;
    //here,use entroy_table to avoid frequency log function which cost losts
of time
    float entroy_table[82];
    const float log2 = log(2.0f);
    entroy_table[0] = 0.0;
    float frequency = 0;
    for (int i = 1; i < 82; i++){
        frequency = (float)i / 81;
        entroy_table[i] = frequency * (log(frequency) / log2);
    }
    int neighborhood_index;
    //        int max_index=pad_src->cols*pad_src->rows;
    float e;
    int current_index = 0;
    int current_index_in_origin = 0;
    for (int y = roi.y; y < roi.height; y++){
        current_index = y * pad_src->cols;
        current_index_in_origin = (y - 4) * gray.cols;
        for  (int  x  =  roi.x;  x  <  roi.width;  x++,  current_index++,
current_index_in_origin++) {
            for (int j = 0; j<neighborhood_num; j++) {
                neighborhood_index = current_index + neighborhood_offset[j];
                hist_count[array[neighborhood_index]]++;
            }
            //get entropy
            e = 0;
            for (int k = 0; k < 256; k++){
                if (hist_count[k] != 0){
                    //                                                   int
frequency=hist_count[k];
                    e -= entroy_table[hist_count[k]];
                    hist_count[k] = 0;
                }
            }
            ((float *)entropy.data)[current_index_in_origin] = e;
        }
    }
    free(neighborhood_offset);
    free(image_cumprod);
    free(cumprod);
    free(neighborhood_corrds_array);

    func_end = clock();
    double func_time = (double)(func_end - func_begin) / CLOCKS_PER_SEC;
    std::cout << "func time" << func_time << std::endl;
}
int main(int argc, char** argv)
{
    cv::Mat src;

    /// Load image
    src = cv::imread("/home/ubuntu/Downloads/bridge.jpeg");
    if (!src.data)
    {
        std::cout << "Usage: EntropyFilter <path_to_image>" << std::endl;
        return -1;
    }
    /// Convert to grayscale
    cvtColor(src, src, cv::COLOR_BGR2GRAY);

    //Calculate Entropy Filter
    cv::Rect roi(0, 0, src.cols, src.rows);
    cv::Mat dst(src.rows, src.cols, CV_32F);
    getLocalEntropyImage(src, roi, dst);
    cv::normalize(dst, dst, 0, 255, cv::NORM_MINMAX);
    cv::Mat entropy;
    dst.convertTo(entropy, CV_8U);

    /// Display results
    namedWindow("Original", cv::WINDOW_AUTOSIZE);
    namedWindow("Entropy Filter", cv::WINDOW_AUTOSIZE);

    imshow("Original", src);
    imshow("Entropy Filter", entropy);

    /// Wait until user exits the program
    cv::waitKey(0);
    return 0;
}
```

Listing 2: Savitsky-Golay custom implementation in OpenCV:

```
#include <opencv2/highgui/highgui.hpp>
#include <opencv2/imgproc/imgproc.hpp>
#include<iostream>
#include<cmath>
using namespace cv;
using namespace std;

int main(int argc, char** argv)
{
Mat im_rgb ; //original image
Mat im_gray; //grayscale image
Mat im_rgb_out;
Mat im_dst; //output image

int ddepth;
Point anchor;
double delta;
int kernel_size;
int order; //polynomial order
int terms; //total number of terms according to order

ddepth=-1; //indicates depth od destination image is similar to original
delta = 0; //value to be added to each pixel after convolution
anchor = Point(-1, -1); //indicates anchor is center that is center point
will be changed by output value
kernel_size=5; //5X5 window (ideal)
order=3; //ideal
terms=10; //for order 3

char* window_name1 = "Savitzky Filter";
char* window_name2 ="Original";
char* window_name3= "Grayscale image";
```

```
char* window_name4= "Grayscale Output After Smoothing";

namedWindow(window_name1, CV_WINDOW_AUTOSIZE); //create window
namedWindow(window_name2, CV_WINDOW_AUTOSIZE); //create window
namedWindow(window_name3, CV_WINDOW_AUTOSIZE); //create window
namedWindow(window_name4, CV_WINDOW_AUTOSIZE); //create window

im_rgb =imread("/home/ubuntu/Downloads/bridge.jpeg");

//check image
if(!im_rgb.data)
return -1;

cvtColor(im_rgb, im_gray, CV_RGB2GRAY); //convert in grayscale

//create X for 3rd order

Mat X(kernel_size*kernel_size, terms, CV_32FC1);

int i=0; //data

while(i<kernel_size*kernel_size)
{
  for(int j=-kernel_size/2;j<=kernel_size/2;j++)
  {
    for(int k=-kernel_size/2; k<=kernel_size/2; k++)
    {
      X.at<float>(i,0)=1;
      X.at<float>(i,1)=j;
      X.at<float>(i,2)=k;
      X.at<float>(i,3)=pow(j,2);
      X.at<float>(i,4)=pow(k,2);
      X.at<float>(i,5)=j*k;
      X.at<float>(i,6)=pow(j,3);
      X.at<float>(i,7)=pow(k,3);
      X.at<float>(i,8)=(pow(j,2))*k;
      X.at<float>(i,9)= j*(pow(k,2));
      ++i;
    }
  }
}

//create kernel auto values
//float  data[5][5]=  {{-0.0743,  0.0114,  0.0400,  0.0114,  -0.0743},{0.0114,
0.0971,  0.1257,  0.0971,  0.0114},{0.0400,  0.1257,  0.1543,  0.1257,  0.0400},
{0.0114, 0.0971, 0.1257, 0.0971, 0.0411},{-0.0743, 0.0114, 0.0400,  0.0114,
-0.0743}};

//Mat kernel(kernel_size, kernel_size, CV_32FC1, data); // (size, size, type:
CV_,NO_ofbits,type_of_bits,channel,number_of_channel)

Mat kernel(kernel_size,kernel_size,CV_32FC1);

Mat C= ((X.t()*X).inv())*X.t(); //kernel mathematics

vector<float> vec=C.row(0); //only C00 for smoothing C01 for 1st derivative
```

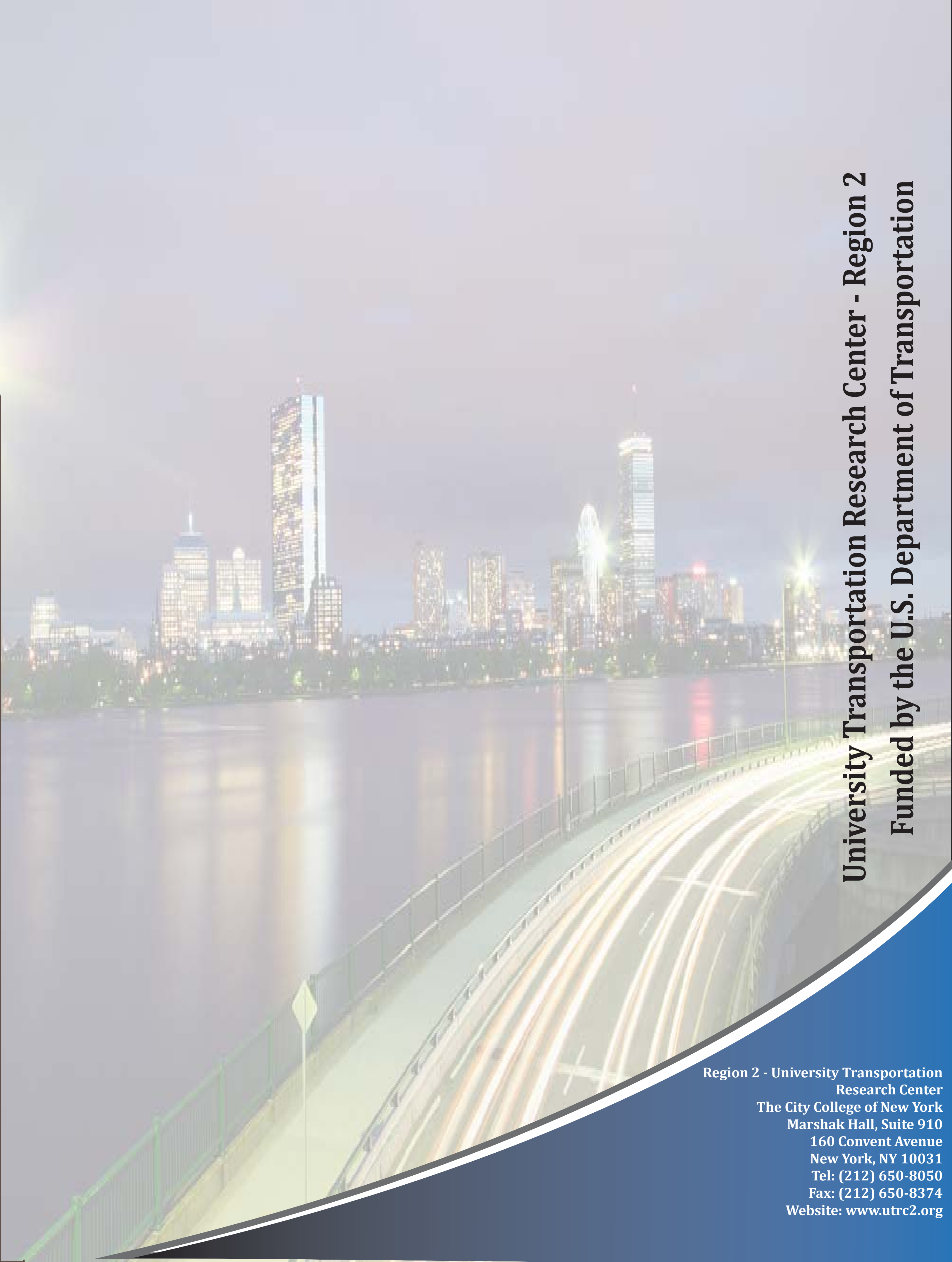
```
memcpy(kernel.data, vec.data(), vec.size()*sizeof(float)); //storing C00 into
kernel

float s= cv::sum(kernel)[0];
kernel=kernel/s; //kernel to be convoluted

//float s= cv::sum(kernel)[0]; //0 for 1 channel
//kernel=kernel/s;

filter2D(im_rgb, im_rgb_out,ddepth, kernel, anchor, delta, BORDER_DEFAULT);

filter2D(im_gray, im_dst, ddepth, kernel, anchor, delta, BORDER_DEFAULT);

imshow(window_name1, im_rgb_out );
imshow(window_name2, im_rgb);
imshow(window_name3, im_gray);
imshow(window_name4,im_dst); //show manual created kernel for differentiating
cv::waitKey(0);
return 0;
}
```

Listing 3: Hough transform example in OpenCV:

```
#include<iostream>
#include<opencv2/highgui/highgui.hpp>
#include <opencv2/imgproc/imgproc.hpp>

using namespace std;
using namespace cv;
int main(int argc, char** argv)
{
char* window_name1="Original";

namedWindow(window_name1, CV_WINDOW_NORMAL); //create window

Mat im_rgb =imread("/home/ubuntu/Downloads/bridge.jpeg");
Mat dst, cdst;
Canny(im_rgb, dst, 50,200,3);
cvtColor(dst, cdst, CV_GRAY2BGR);
vector<Vec4i> lines;
HoughLinesP(dst, lines, 1, CV_PI/180, 100,40,10);
for(size_t i=0; i<lines.size(); i++)
{
   Vec4i l=lines[i];
   line(cdst,  Point(l[0],  l[1]),  Point(l[2],l[3]),  Scalar(0,0,255),  3,
CV_AA);

}
imshow(window_name1, im_rgb );
imshow("detected", cdst);

waitKey();
return 0;
}
```

Listing 4: Range filtering OpenCV algorithm:

```
#include<iostream>
#include "opencv2/imgproc/imgproc.hpp"
#include "opencv2/highgui/highgui.hpp"
#include "highgui.h"
#include<stdlib.h>
#include <stdio.h>

using namespace std;
using namespace cv;

int main()
{
Mat im_orig; //input image
Mat im_erode, im_dilate; //output image
Mat im_diff;

im_orig=imread("/home/ubuntu/Downloads/bridge.jpeg");
if(!im_orig.data)
return -1;

cvtColor(im_orig, im_orig, COLOR_BGR2GRAY);
namedWindow("Erosion", CV_WINDOW_AUTOSIZE);
namedWindow("Dilation", CV_WINDOW_AUTOSIZE);
namedWindow("Difference", CV_WINDOW_AUTOSIZE);

 cout<<"Implementing Erosion: "<<endl;
Mat element_erode = getStructuringElement( MORPH_RECT,
                                       Size( 5, 5 ),
                                       Point(-1,-1) );

  /// Apply the erosion operation
  erode( im_orig, im_erode, element_erode );
  imshow( "Erosion", im_erode );
cout<<"Implementing Dilation" <<endl;
Mat element_dilate = getStructuringElement( MORPH_RECT,
                                       Size( 5, 5 ),
                                       Point(-1,-1 ) );
  /// Apply the dilation operation
  dilate( im_orig, im_dilate, element_dilate );
  imshow( "Dilation", im_dilate );

cout<<"Range Filtered Image:"<<endl;
im_diff=im_dilate-im_erode;
imshow("Difference",im_diff);

waitKey(0);
return 0;
}
```



University Transportation Research Center - Region 2
Funded by the U.S. Department of Transportation

Region 2 - University Transportation Research Center
The City College of New York
Marshak Hall, Suite 910
160 Convent Avenue
New York, NY 10031
Tel: (212) 650-8050
Fax: (212) 650-8374
Website: www.utrc2.org