

5-12-2023

## Automated mapping of oblique imagery collected with unmanned vehicles in coastal and marine environments

Jacob B. Freeman

Mississippi State University, Jbf209@msstate.edu

Follow this and additional works at: <https://scholarsjunction.msstate.edu/td>



Part of the [Environmental Monitoring Commons](#), and the [Other Environmental Sciences Commons](#)

---

### Recommended Citation

Freeman, Jacob B., "Automated mapping of oblique imagery collected with unmanned vehicles in coastal and marine environments" (2023). *Theses and Dissertations*. 5836.

<https://scholarsjunction.msstate.edu/td/5836>

This Graduate Thesis - Open Access is brought to you for free and open access by the Theses and Dissertations at Scholars Junction. It has been accepted for inclusion in Theses and Dissertations by an authorized administrator of Scholars Junction. For more information, please contact [scholcomm@msstate.libanswers.com](mailto:scholcomm@msstate.libanswers.com).

Automated mapping of oblique imagery collected with unmanned vehicles in coastal and marine  
environments

By

Jacob B. Freeman

Approved by:

Adam Skarke (Major Professor)  
John Cartwright (Committee Member)  
Qingmin Meng (Committee Member)  
Jamie L. Dyer (Committee Member)  
Andrew E. Mercer (Graduate Coordinator)  
Rick Travis (Dean, College of Arts & Sciences)

A Thesis  
Submitted to the Faculty of  
Mississippi State University  
in Partial Fulfillment of the Requirements  
for the Degree of Master of Science  
in Environmental Geoscience  
in the Department of Geosciences

Mississippi State, Mississippi

May 2023

Copyright by  
Jacob B. Freeman  
2023

Name: Jacob B. Freeman

Date of Degree: May 12, 2023

Institution: Mississippi State University

Major Field: Environmental Geoscience

Major Professor: Adam Skarke

Title of Study: Automated mapping of oblique imagery collected with unmanned vehicles in coastal and marine environments

Pages in Study: 195

Candidate for Degree of Master of Science

Recent technological advances in unmanned observational platforms, including remotely operated vehicles (ROVs) and small unmanned aerial systems (sUAS), have made them highly effective tools for research and monitoring within marine and coastal environments. One of the primary types of data collected by these systems is video imagery, which is often captured at an angle oblique to the Earth's surface, rather than normal to it (e.g., downward looking). This thesis presents a newly developed suite of tools designed to digitally map oblique imagery data collected with ROV and sUAS in coastal and marine environments and quantitatively evaluates the accuracy of the resultant maps. Results indicate that maps generated from oblique imagery collected with unmanned vehicles have highly variable accuracy relative to maps generated with imagery data collected with conventional mapping platforms. These results suggest that resultant maps have the potential to match or even surpass the accuracy of maps generated with imagery data collected with conventional mapping platforms but realizing that potential is largely dependent upon careful survey design.



## DEDICATION

I dedicate this thesis to my friends and family who stood by my side throughout this monumental time in my life. Most of all I would like to thank Thomas J. Freeman and Beverly B. Freeman – my supportive and wonderful parents who raised me to take leaps in life even when the outcome is uncertain. As one chapter ends, a new chapter opens. The lessons you taught me and the passion for the ocean and adventure both of you instilled in me since I was a young boy will only continue to grow and will be carried with me wherever I go. I truly would not be the man I am today without you.

## ACKNOWLEDGEMENTS

I would first like to thank my advisor, Prof. Adam Skarke, for being such a mentor throughout this process. Your office was always open to me for questions and guidance over the years; you provided me with constant support and presented me with research and academic opportunities that I will never forget, it has truly been a pleasure working by your side. I would especially like to thank Lee Hathcock at Mississippi State University's Geosystems Research Institute (GRI) for expanding my knowledge of computer programming and never hesitating to help no matter the issue or the time of day. You have taught me so much and I cannot thank you enough for being involved in this research. There are so many influential people within the Department of Geosciences at Mississippi State University who have made my time here life-changing and special. This research was funded in part by two grants from the National Oceanic and Atmospheric Administration: (NA16OAR4320199) Geospatial analysis of deep-sea environments using ROV video data with the Coastal Marine Ecological Classification Standard and (NA19NOS4730207) Regional Geospatial Modeling. I would like to thank Rachel Basset and Peter Etnoyer of the NOAA National Centers for Coastal Ocean Science Deep Coral Ecology Laboratory for their support in the portion of this research focused on mapping ROV video data – without you all, none of this would have been possible.

## TABLE OF CONTENTS

DEDICATION .....	ii
ACKNOWLEDGEMENTS .....	iii
LIST OF TABLES .....	vi
LIST OF FIGURES .....	vii
CHAPTER	
I. INTRODUCTION .....	1
II. BACKGROUND .....	7
2.1 Remotely Operated Vehicles .....	7
2.2 Small Unmanned Aerial Systems .....	10
2.3 Oblique Imagery & Orthorectification .....	13
2.4 Remotely Operated Vehicle Video Annotation and Viewshed Mapping .....	16
III. METHODS .....	20
3.1 ROV Data Processing Framework .....	20
3.1.1 Extraction of Annotation & Navigation Data .....	21
3.1.2 Revision of Substrate Annotation to Fine Resolution CMECS Classes .....	22
3.1.3 Generation of ROV Viewshed Maps .....	23
3.2 Evaluation of ROV Viewshed Maps .....	27
3.3 sUAS Data Processing Framework .....	29
3.3.1 Video Frame extraction & timestamp scripts .....	32
3.3.2 Agisoft Metashape Orthorectification .....	34
3.3.3 Evaluation of Spatial Accuracy .....	36
IV. RESULTS .....	39
4.1 Automated ROV Substrate Maps .....	39
4.2 ROV Substrate Map Evaluation .....	41
4.3 sUAS Orthorectification .....	44
4.4 sUAS Map Evaluation .....	44
4.5 ROV Orthorectification .....	59

V.	DISCUSSION.....	63
5.1	ROV Automated Mapping Implementation .....	63
5.2	Suggested ROV Viewshed Mapping Improvements.....	65
5.3	ROV Orthorectification of Oblique Imagery .....	67
5.4	sUAS Orthorectification Implementation.....	74
5.5	Suggested sUAS Orthorectification Mapping Improvements .....	76
5.6	sUAS Applications of Orthorectified Oblique Imagery .....	77
5.6.1	Coastal Mapping.....	77
5.6.2	Marine Spatial Planning .....	79
5.6.3	Storm Damage Assessment .....	80
VI.	CONCLUSION .....	82
6.1	ROV Automated Mapping .....	83
6.2	ROV oblique Image Orthorectification .....	84
6.3	sUAS Oblique Image Orthorectification .....	85
6.4	Future Work.....	86
	REFERENCES .....	88
	APPENDIX	
A.	ROV AUTOMATED MAPPING PYTHON SCRIPTS .....	95
B.	ROV AUTOMATED MAPPING STANDARD OPERATING PROCEDURES .....	103
C.	FINAL ROV SUBSTRATE MAPS .....	130
D.	BACKSCATTER COMAPARISON OF ROV SUBSTRATE MAPS & DISTRIBUTION BOXPLOTS.....	165
E.	ROV AND sUAS FRAME EXTRACTION AND TIMESTAMP SCRIPTS.....	185

## LIST OF TABLES

Table 4.1	Comparison of RMSE Values for each sUAS Orthorectified Orthomosaic .....	50
-----------	--	----

## LIST OF FIGURES

Figure 2.1	ROV Deep Discoverer (NOAA OE) .....	9
Figure 2.2	NOAA Ship <i>Okeanos Explorer</i> (NOAA OE).....	9
Figure 2.3	DJI Phantom 4 ( <a href="https://www.dji.com/phantom-4-pro">https://www.dji.com/phantom-4-pro</a> ).....	12
Figure 2.4	Vertical and Oblique Imagery .....	14
Figure 2.5	Comparison of Orthorectification Software .....	15
Figure 3.1	NOAA ROV <i>Okeanos Explorer</i> Study Sites.....	21
Figure 3.2	CMECS compliant substrate scheme (Basset et al, 2017) .....	23
Figure 3.3	Deep Discoverer Viewshed Concept.....	25
Figure 3.4	Viewshed Mapping Approach.....	25
Figure 3.5	Example of the final ROV substrate map of EX 1903 L2 Dive 09 .....	26
Figure 3.6	General Automated ROV Viewshed Mapping Workflow .....	27
Figure 3.7	Mississippi State University North Farm Study Site with flight track lines. ....	31
Figure 3.8	Mississippi Gulf Coast Study Site (Waveland, MS) .....	32
Figure 3.9	General sUAS Workflow .....	36
Figure 4.1	Seafloor substrate map generated from observations recorded during EX 1903 L2 Dive 09.....	41
Figure 4.2	Backscatter Comparison of EX 1806 Dive 13 .....	42
Figure 4.3	Example Boxplot of EX 1806 Dive 13 Backscatter Distribution.....	43
Figure 4.4	Root Mean Square Error (RMSE) equation (ESRI, 2022) .....	45
Figure 4.5	Orthorectification of North Farm sUAS Flight 01 .....	46
Figure 4.6	Orthorectification of North Farm sUAS Flight 02 .....	47

Figure 4.7	Orthorectification of Mississippi Gulf Coast sUAS flight 01 .....	48
Figure 4.8	Orthorectification of Mississippi Gulf Coast sUAS Flight 02 .....	49
Figure 4.9	Orthorectification of North Farm Flight 01 Error Map .....	51
Figure 4.10	North Farm sUAS Flight 01 Error Graph.....	52
Figure 4.11	Orthorectification of North Farm Flight 02 Error Map .....	53
Figure 4.12	North Farm sUAS Flight 02 Error Graph.....	54
Figure 4.13	Error Between GPS Ground Control Points and 3cm Aerial Image .....	55
Figure 4.14	Error Between North Farm sUAS Flight 01 and 3cm Aerial Image .....	56
Figure 4.15	Error Between sUAS North Farm Flight 02 and 3cm Aerial Image .....	57
Figure 4.16	Error Between sUAS Coast Flight 01 and Satellite Image.....	58
Figure 4.17	Error Between sUAS Coast Flight 02 and Satellite Image.....	59
Figure 4.18	Orthorectified ROV Orthomosaic Comparison to Observed Substrate EX 1903 L2 Dive 06.....	61
Figure 4.19	Orthorectified ROV Orthomosaic Comparison to Observed Substrate EX 1903 L2 Dive 08.....	62
Figure 5.1	ROV Image of seafloor EX 1903 L2 Dive 06.....	69
Figure 5.2	ROV orthorectified orthomosaic image of the seafloor from EX 1903 L2 Dive 06 .....	70
Figure 5.3	ROV image of benthic features throughout EX 1903 L2 Dive 08 .....	71
Figure 5.4	Orthomosaic image section of EX 1903 L2 Dive 08. ....	72
Figure 5.5	ROV image of marine organisms during EX 1903 L2 Dive 16.....	73
Figure 5.6	Orthomosaic image section of EX 1903 L2 Dive 16 .....	74
Figure A.1	<i>CMECS classification script.py</i> .....	96
Figure A.2	Mapping Script.py .....	98
Figure A.3	<i>Map production script.py</i> .....	100
Figure B.1	ROV Automated Mapping SOP #1 .....	104

Figure B.2 ROV Automated Mapping SOP #2 .....	106
Figure B.3 ROV Automated Mapping SOP #3 .....	110
Figure B.4 ROV Automated Mapping SOP #4 .....	114
Figure B.5 ROV Automated Mapping SOP #5 .....	124
Figure C.1 Classification of substrate EX 1803 Dive 03 .....	131
Figure C.2 Classification of substrate EX 1803 Dive 04 .....	132
Figure C.3 Classification of substrate EX 1803 Dive 05 .....	133
Figure C.4 Classification of substrate EX 1803 Dive 07 .....	134
Figure C.5 Classification of substrate EX 1803 Dive 08 .....	135
Figure C.6 Classification of substrate EX 1803 Dive 09 .....	136
Figure C.7 Classification of substrate EX 1803 Dive 10 .....	137
Figure C.8 Classification of substrate EX 1803 Dive 11 .....	138
Figure C.9 Classification of substrate EX 1803 Dive 12 .....	139
Figure C.10 Classification of substrate EX 1803 Dive 14 .....	140
Figure C.11 Classification of substrate EX 1803 Dive 15 .....	141
Figure C.12 Classification of substrate EX 1806 Dive 02 .....	142
Figure C.13 Classification of substrate EX 1806 Dive 03 .....	143
Figure C.14 Classification of substrate EX 1806 Dive 04 .....	144
Figure C.15 Classification of substrate EX 1806 Dive 05 .....	145
Figure C.16 Classification of substrate EX 1806 Dive 06 .....	146
Figure C.17 Classification of substrate EX 1806 Dive 08 .....	147
Figure C.18 Classification of substrate EX 1806 Dive 10 .....	148
Figure C.19 Classification of substrate EX 1806 Dive 11 .....	149
Figure C.20 Classification of substrate EX 1806 Dive 12 .....	150
Figure C.21 Classification of substrate EX 1806 Dive 13 .....	151



Figure C.22	Classification of substrate EX 1806 Dive 14 .....	152
Figure C.23	Classification of substrate EX 1806 Dive 16 .....	153
Figure C.24	Classification of substrate EX 1806 Dive 17 .....	154
Figure C.25	Classification of substrate EX 1903 L2 Dive 01 .....	155
Figure C.26	Classification of substrate EX 1903 L2 Dive 02 .....	156
Figure C.27	Classification of substrate EX 1903 L2 Dive 05 .....	157
Figure C.28	Classification of substrate EX 1903 L2 Dive 06 .....	158
Figure C.29	Classification of substrate EX 1903 L2 Dive 08 .....	159
Figure C.30	Classification of substrate EX 1903 L2 Dive 09 .....	160
Figure C.31	Classification of substrate EX 1903 L2 Dive 10 .....	161
Figure C.32	Classification of substrate EX 1903 L2 Dive 11 .....	162
Figure C.33	Classification of substrate EX 1903 L2 Dive 16 .....	163
Figure C.34	Classification of substrate type throughout EX 1903 L2 Dive 19.....	164
Figure D.1	Backscatter Comparison of EX 1803 Dive 08 .....	166
Figure D.2	Backscatter Comparison of EX 1803 Dive 15 .....	167
Figure D.3	Backscatter Comparison of EX 1806 Dive 04 .....	168
Figure D.4	Backscatter Comparison of EX 1806 Dive 08 .....	169
Figure D.5	Backscatter Comparison of EX 1806 Dive 13 .....	170
Figure D.6	Backscatter Comparison of EX 1903 L2 Dive 05 .....	171
Figure D.7	Backscatter Comparison of EX 1903 L2 Dive 06 .....	172
Figure D.8	Backscatter Comparison of EX 1903 L2 Dive 09 .....	173
Figure D.9	Backscatter Comparison of EX 1903 L2 Dive 10 .....	174
Figure D.10	Backscatter Comparison of EX 1903 L2 Dive 19 .....	175
Figure D.11	EX 1803 Dive 08 Backscatter Distribution Boxplot .....	176
Figure D.12	EX 1803 Dive 15 Backscatter Distribution Boxplot .....	177

Figure D.13EX 1806 Dive 04 Backscatter Distribution Boxplot .....	178
Figure D.14EX 1806 Dive 08 Backscatter Distribution Boxplot .....	179
Figure D.15EX 1806 Dive 13 Backscatter Distribution Boxplot .....	180
Figure D.16EX 1903 Leg 2 Dive 05 Backscatter Distribution Boxplot.....	181
Figure D.17EX 1903 Leg 2 Dive 06 Backscatter Distribution Boxplot.....	182
Figure D.18EX 1903 Leg 2 Dive 09 Backscatter Distribution Boxplot.....	183
Figure D.19EX 1903 Leg 2 Dive 19 Backscatter Distribution Boxplot.....	184
Figure E.1 ROV Frame Extraction and Timestamp Script .....	186
Figure E.2 North Farm sUAS Frame Extraction and Timestamp Script.....	190
Figure E.3 Mississippi Gulf Coast sUAS Frame Extraction and Timestamp Script.....	193

## CHAPTER I

### INTRODUCTION

Marine and coastal environments provide important habitat, ecosystem services, and natural resources. Additionally, they are the location of substantial natural hazards, including tropical cyclones, coastal erosion, and tsunami (Wright et al., 2019). Given their relevance to a broad range of natural processes and societal concerns, scientists, environmental managers, and policymakers have focused a great deal of attention on understanding, predicting, and monitoring coastal and marine environmental dynamics (Taddia et al. 2020). The success of these efforts is largely dependent upon the ability of investigators to accurately survey and map features of interest in coastal and marine environments that are often rapidly changing (Patel et al. 2021). This in turn is dependent upon access to efficient and economical mapping platforms that can survey these environments with high temporal and spatial resolution and be deployed rapidly in response to events such as storm landfall.

Conventionally, subaerial portions of coastal and marine environments have been mapped with mosaiced still imagery collected with satellite or manned aircraft remote sensing platforms (Klemas, 2011) and submarine portions have been mapped with sonar acoustic reflectivity imagery collected from manned ships or boats (Lurton, 2010; Mayer, 2006). These approaches have been highly effective at generating accurate maps of coastal and marine environments but have associated limitations that have constrained their utility for many research and management applications. Most notably, these conventional mapping data collection

approaches are expensive, can be limited in their spatial resolution due to distance of the platform from the surveyed surface, and are highly constrained in their temporal resolution. For example, the temporal sampling frequency of satellite remote sensing for a specific location is generally fixed by the satellite orbital path (Prost, 2013), precluding higher frequency surveys or precisely timed surveys relative to an event like a landfalling tropical cyclone. Additionally, the utility of satellite and aerial platforms is often limited by the presence of clouds between the platform and the surveyed surface (Ju & Roy, 2008), which are particularly common in association with atmospheric events, like storms, that are most likely to rapidly modify coastal and marine environments.

The recent rapid development of aerial and marine unmanned vehicle technology has resulted in the broader availability of observational platforms that have the capacity to effectively map coastal and marine environments while overcoming some of the limitations of conventional mapping platforms. Both underwater remotely operated vehicles (ROV) and small unmanned aircraft systems (sUAS) provide professional and amateur users with the ability to collect large quantities of spatially referenced imagery data that has not previously been widely available. Notably, these unmanned systems are now routinely deployed with cameras that can collect video data at ultra-high resolutions (e.g., 4 k), which surpasses the spatial resolution of other tools commonly used for subaerial and submarine environmental imaging such as satellite remote sensing, traditional aerial photography, side scan sonar, and multibeam sonar. The costs of sUAS platform acquisition and operation are multiple orders of magnitude less than satellite or manned aircraft mapping platforms. Additionally, sUAS platforms can be rapidly deployed in a specific location in response to an environmental event of interest and the frequency of repeat surveys is essentially unlimited. Moreover, sUAS can operate below the cloud ceiling, yielding

more continuous and therefore useful mapping imagery. ROV platforms allow for seafloor mapping with orders of magnitude higher resolution relative to sonar imagery collected from a manned boat or ship. Additionally, ROV platforms allow focused optical image mapping, which yield substantially greater detail and insight into environmental conditions than acoustic image mapping from manned vessels.

Although ROV and sUAS platforms offer advantages relative to conventional mapping platforms, they also have some associated potential limitations that must be addressed and quantified. A substantial challenge to effectively mapping an area with ROV and sUAS imagery is that the platforms often require their primary camera to collect video imagery, with an outward-looking high-oblique camera angle (near horizontal), for effective vehicle piloting, navigation, and real-time environmental assessment. This is in contrast to most aerial and satellite mapping remote sensing platforms in which still imagery is collected with a vertical or near vertical (low-oblique) camera angle in order to minimize geometric image distortion (Prost, 2013). This is also in contrast to seafloor mapping in which acoustic reflectivity data is collected at a wide range of angles to the imaging instrument (sonar transducer) but the nature of the acoustic data precludes the image distortion associated with high-oblique optical camera angles. Notably, only limited research has focused on collection and processing of optical imagery data collected with sUAS and ROVs for mapping purposes (e.g. Dunford et al., 2009; Hugenholtz , 2012; Hugenholtz et al., 2013; Lalibertre et al., 2009; Yahyanejad et al., 2011; Zhou, 2009) and standardized collection procedures as well as processing workflows are not yet well established for these data, as they are for data collected with conventional mapping platforms.

The fundamental goal of mapping operations with an sUAS and ROVs is accurate orthorectification - positioning the near horizontal, high-oblique imagery on the Earth's surface

in a uniform geospatial context that accounts for geometric image distortion to the greatest degree possible. Achieving suitable accuracy in the orthorectification of imagery collected with sUAS and ROVs is a critical first step in quantitatively analyzing and understanding spatial relationships among the imaged features in the collected data. Given the unique advantages and challenges associated with ROV and sUAS collected high-oblique video imagery, relative to traditional remote sensing still imagery collected near vertically, it is necessary to quantitatively evaluate the spatial accuracy of maps generated with ROV and sUAS image data as well as their comparability to maps generated with traditional remote sensing platforms. Accordingly, **the goal of this thesis is to evaluate the capacity of ROV and sUAS platforms to accurately map coastal and marine environments with oblique video imagery.** This goal will be achieved through the execution of the following four objectives:

- 1) Develop an automated data processing framework for the generation of georeferenced maps from ROV video imagery and its derivative data.**
- 2) Quantitatively evaluate the spatial agreement between produced ROV maps and those produced with conventional sonar image mapping approaches and platforms.**
- 3) Develop an automated data processing framework for the generation of georeferenced maps from sUAS video imagery.**
- 4) Quantitatively evaluate the spatial accuracy of resulting sUAS maps and determine their comparability to that of maps produced with conventional remote sensing platforms.**

This thesis presents and evaluates the efficacy a newly developed automated framework for digital geospatial mapping of oblique video imagery and associated derivative environmental data, collected with unmanned vehicles in coastal and marine environments. The utility of resultant maps generated from ROV data was determined by assessing the degree to which ROV mapped seafloor substrate classification polygons were spatially coincident with seafloor substrate as indicated by independent sonar acoustic backscatter maps of the seafloor. It was not possible to evaluate the absolute spatial accuracy of resultant ROV maps because it is not feasible to collect the equivalent of GPS “ground truth” points on the seafloor. The spatial accuracy of resultant maps generated from sUAS collected data was evaluated through determination of the horizontal Root Mean Square Error (RMSE) of the mapped position of imaged features relative to their “ground truth” position as measured by a survey grade GPS. The comparability of resultant maps generated from sUAS collected data with maps generated by conventional remote sensing platforms was evaluated by comparing the determined horizontal accuracy of the sUAS and conventional remote sensing maps as well as the RMSE of the mapped position of imaged features on sUAS maps relative to matching features on conventional remote sensing maps. Quantifying the spatial accuracy of resultant sUAS and the agreement of resultant ROV and sUAS maps with maps generated by conventional mapping methods, collectively demonstrates the capacity of ROV and sUAS platforms to accurately map coastal and marine environments with oblique video imagery.

Results indicate that oblique imagery collected with unmanned vehicles in coastal and marine environments potentially can be used to map features in these environments with a spatial accuracy comparable to, or in some cases better than that of conventional mapping approaches. However, in some cases resultant accuracies are substantially poorer than those of conventional

mapping approaches. The comparability of accuracy or lack thereof appears to be largely dependent upon survey design. These results indicate that users potentially can realize the substantial advantages of ROV and sUAS platforms with respect to cost, resolution, sampling frequency, deployment speed, and survey altitude (below cloud ceiling) while achieving results with spatial accuracies comparable to conventional platforms, that lack these advantages. Additionally, this thesis demonstrates that accurate maps can be generated from ROV and sUAS data that was not originally collected for the purpose of generating maps. This indicates that the large volume of previously collected ROV and sUAS video data available in archives may be used for generating accurate maps of environments even when that was not a consideration of the original survey design. This also suggests that this approach may be a useful way for investigators to rapidly visualize the information contained in video data records when the time required for full video review is prohibitive.

Collectively, the results of this research yield methods that advance the capacity of the scientific community and the general public to efficiently create accurate maps of marine and coastal environments using data collected with unmanned vehicles. This will enable members of the scientific community to rapidly assess the value of existing publicly available video data to their research objectives, improving data accessibility and use. Additionally, it will enable professional and amateur sUAS pilots that have access to large quantities of oblique aerial imagery to generate georeferenced aerial images with greater resolution, efficiency, and frequency while doing so at a lower cost than satellite and traditional aerial remote sensing imagery. This outcome will support efforts to monitor long-term coastal environment evolution as well as rapid and targeted assessment of coastal change and impacts after natural hazards such as hurricane landfall.



## CHAPTER II

### BACKGROUND

#### **2.1 Remotely Operated Vehicles**

ROVs are unmanned submersibles equipped with camera systems and environmental sensors that allow research to be conducted in underwater environments. These unmanned robotic submersibles are connected to a support vessel by a cable tether. Operators onboard the support vessel pilot the ROV and control its data collection systems, which include cameras and environmental sensors. (Macreadie et al. 2018). ROVs are used for challenging underwater tasks such as infrastructure construction, and maintenance operations in offshore industries, including renewable energy and petroleum extraction industries, as well as oceanographic research, marine archeology, and naval defense operations worldwide (Castro et al. 2019). ROVs regularly deliver high-resolution video imagery of the seafloor and water column, supporting industrial applications and enabling exploratory scientific research in one of the most underexplored environments on Earth (Marsh et al, 2013).

As noted, ROVs are connected to a surface ship by a reinforced cable tether, which allows operators on the ship to control the vehicle as it maneuvers underwater and transmits operational and sensor data to the surface in real-time. These data include vehicle information like position (geographic coordinates), depth, altitude, attitude, and diagnostic observations, as well as measurements of environmental conditions such as water temperature, salinity, pH, and dissolved O<sub>2</sub> (Castro et al. 2019). The most important data transmitted from the ROV is the live

video feed because it is critical to effective vehicle operation as well as the research or industrial goals of the ROV dive. Because the live video feed is the principal means by which the operator's pilot and navigate the ROV over seafloor terrain and around obstacles, the primary high-resolution ROV camera angle must be near horizontal, resulting in oblique imagery of the seafloor. This necessarily makes utilization of the resulting video observations to map seafloor features more challenging, particularly in cases where geospatial analysis of imaged seafloor features is an important component of research.

The ROV data used in this thesis were collected with the ROV Deep Discoverer (Figure 2.1), which is operated by the National Oceanic Atmospheric Administration (NOAA) Office of Ocean Exploration (OE) and deployed from NOAA research vessel *Okeanos Explorer* (Figure 2.2). Deep Discoverer can dive up to depths of 6,000 meters, allowing oceanographers and marine scientists the opportunity to access approximately 96% of the global ocean floor and overlying water column. Deep Discoverer has two manipulator arms and can collect in-situ biological and geological seafloor samples during the duration of a dive and numerous environmental sensors designed to characterize the surrounding marine environment. The Deep Discoverer is equipped with extensive LED lighting and a high-resolution (2,200,000 pixels) forward-looking video camera system for recording seafloor and water column observations (NOAA OER).

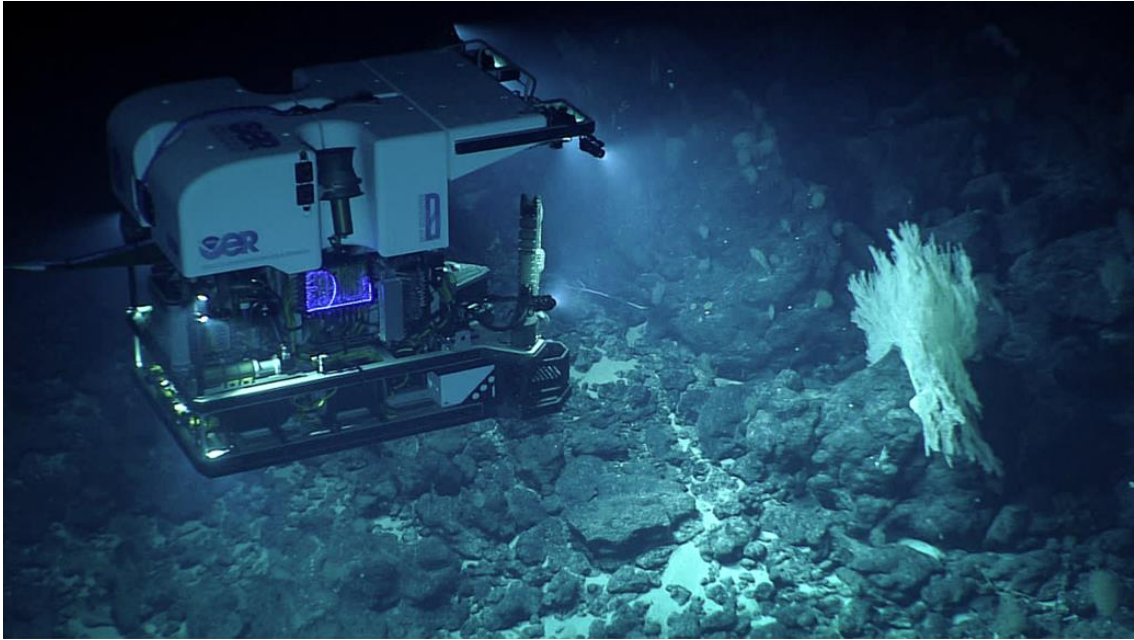


Figure 2.1 ROV Deep Discoverer (NOAA OE)



Figure 2.2 NOAA Ship *Okeanos Explorer* (NOAA OE)

NOAA ship *Okeanos Explorer* is currently the only United States federal research vessel designed and committed solely for deep ocean exploration and research [National Oceanic and Atmospheric Administration – Office of Ocean Exploration and Research, 2015].

## **2.2 Small Unmanned Aerial Systems**

Small unmanned aerial Systems (sUAS) are low-cost, efficient, user-friendly light-weight aircraft that are used for the collection of high-resolution aerial video and imagery data suitable for precision geospatial mapping, and high-resolution image analyses (Nagarajan et al. 2019). The initial development and deployment of sUAS were largely driven by governmental funding focused on surveillance and defense applications (Allen and Walsh, 2008). However, as sUAS systems have become more user-friendly and affordable in recent years their adoption by the scientific research and environmental management communities as well as by armature hobbyists has become widespread (Laliberte and Rango, 2008; Rodin, 2019). Through the use of sUAS, scientific studies requiring aerial imagery, that at one point were impossible, are now being routinely performed because sUAS can be used to collect observations where factors such as the cost, resolution, and operational inflexibility of standard remote sensing techniques (e.g., satellites, traditional aerial photo surveys with manned aircraft) are prohibitive (Whitehead et al. 2021).

The design of sUAS is divided into two categories. Rotary wing drones are constructed with blades that rotate around several rotors, like a helicopter. Fixed-wing drones are designed like traditional aircraft with fixed wings that lift the aircraft off the ground as forwarding airspeed increases (Kandrot and Holloway, 2020). One of the most widely used sUAS in the rotary wing class is the Da-Jiang Innovations (DJI) Phantom series, which includes the Phantom 4 Pro (Taddia et al. 2020), which is used for data collection in this thesis (Figure 2.3). Regardless of design class, sUAS are generally equipped with advanced cameras that enable imaging of the Earth surface across a wide range of bands of the electromagnetic spectrum, but most commonly, the visible light spectrum.

The sensor payloads packages carried on sUAS can include active (e.g., LiDAR, InSAR, SRT, Radar, and PSInSAR) and/or passive (i.e., hyperspectral Imaging, multispectral imaging, aerial Photography, FLIR, long-wave infrared, and near-infrared surveys) remote sensing instruments. (Malthus and Mumby, 2010; Patel et al., 2021; McBride and Byrnes, 1997; 2020; Yu and Action, 2004; Lin et al., 2019). The primary payload from most sUAS is a camera that records reflectance over bands of the electromagnetic spectrum. Most commonly these cameras record video of the visible light spectrum since this real-time data is most useful for pilots flying the sUAS. The sUAS video data used for this thesis were captured with an RGB camera system, which measures the reflectance of light in the visible portion of the electromagnetic spectrum (wavelengths between 0.4 and 0.75) (Kandrot and Holloway, 2020). An RGB camera system produces a “true color image” which is composed of individual pixels each containing reflectance information for the elements observed within the image (Kandrot and Holloway, 2020).

For situations requiring aerial imagery of an area of interest, sUAS offers many advantages over traditional imaging platforms such as satellites and manned aircraft (Sturdivant et al, 2017). Because they can fly at much lower altitudes, sUAS can provide high-resolution imagery of a study area with a less extensive coverage area (Morgan and Hodgson, 2020). Flight at lower altitudes also allows sUAS to operate below the cloud ceiling resulting in less obscured land surface imagery that is available from satellite platforms when clouds are present. Additionally, sUAS surveys can be flown at a lower cost than comparable manned aircraft surveys and can be conducted more frequently given the much lower threshold for mission planning and execution. Finally, sUAS are widely available and thus surveys can be conducted

when satellites are not in a position to image an area of interest and when manned aircraft, or required pilots, are unavailable (Kandrot and Holloway, 2020).

A fundamental challenge of creating maps with sUAS video imagery data is the fact that much of it is collected with cameras oriented obliquely to the Earth's surface, often with a near horizontal camera angle. An outward-looking oblique camera angle is often necessary for effective vehicle piloting and navigation but presents numerous challenges related to the use of the resulting imagery data. The most notable video data processing challenge created by the configuration of the sUAS camera when capturing oblique images is accurately plotting the location of the imaged portion of the Earth's surface in a geospatial context.



Figure 2.3 DJI Phantom 4 (<https://www.dji.com/phantom-4-pro>)

The DJI Phantom 4 is paired with a 1-inch 20-megapixel sensor with the capability of producing 4k/60fps video data and burst mode still imagery at 14 fps. The addition of a titanium alloy and magnesium alloy construction housing increases the durability of the sUAS airframe and diminishes weight, making the construction and design of the Phantom 4 Pro comparable in weight to the Phantom 4. (<https://www.dji.com/phantom-4-pro>)

### **2.3 Oblique Imagery & Orthorectification**

Aerial photography for mapping purposes has traditionally been collected with cameras positioned in a straight downward orientation normal ( $0^\circ$ ) to the land/water surface, or as close to this normal as allowed by platform motion (Nesbit and Hugenholtz, 2019) (Figure 2.4). Given this geometry, the process of “stitching” the individual images together to create mosaic images of the Earth’s surface has been relatively straightforward. Conventional oblique imagery is photography or videography that is collected at an angle relative to the surface of the Earth, which is typically between  $40^\circ$  and  $50^\circ$  (Nesbit and Hugengoltz, 2019). Due to the unique tilt of oblique imagery both image scale and pixel coverage on the ground can vary tremendously (Höhle, 2008). This complexity of image scale and pixel coverage variability results in geometric distortion of the image and is exacerbated as camera angles increase towards horizontal (Wiedemann and More, 2012). One of the primary challenges of mapping with oblique imagery is orthorectification, is the process of precisely determining the geographical position of each pixel in an image based on the Cartesian (latitude, longitude, altitude) position and orientation of the camera at the point in time the image was collected. The orthorectification process also includes the correction of geometric distortion in images such as height distortion and tilt displacement which commonly result from the movement of the vehicle that is being used to collect the aerial imagery data (Zhou et al. 2005). Indeed, a fundamental challenge to using ROV and sUAS imagery data to create mosaic maps of the Earth’s surface is the fact that much of it is collected with cameras oriented obliquely ( $> 0^\circ$ ) to the Earth’s surface (Figure 2.4) and often near horizontal. This is the case because an outward-looking oblique camera angle is often necessary for the effective piloting and navigation of unmanned vehicles.

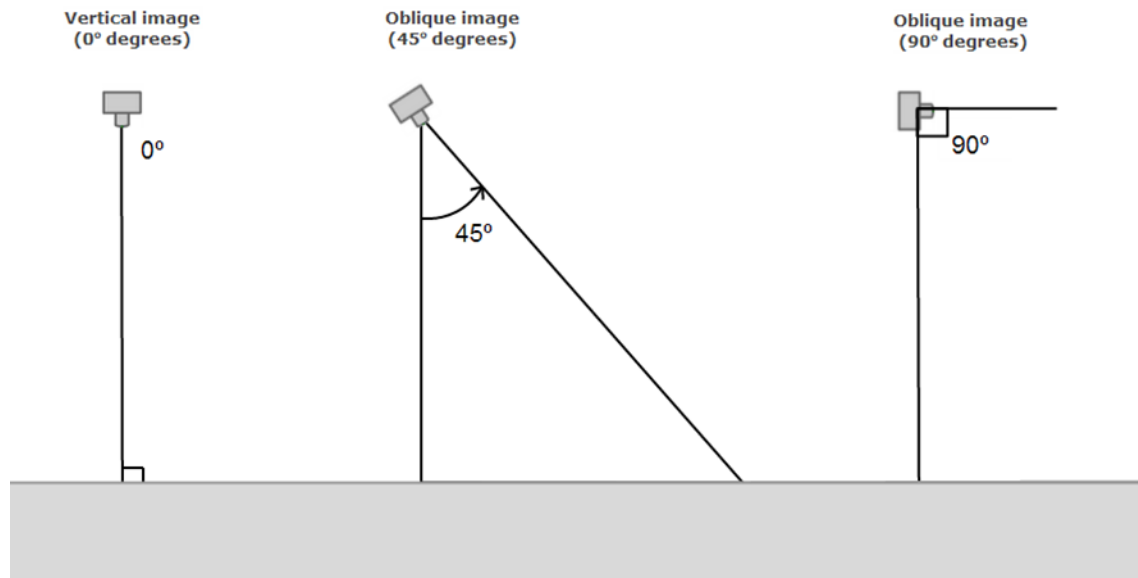


Figure 2.4 Vertical and Oblique Imagery

Images that are taken by the camera pointing perpendicular to the ground are vertical. Images that are taken when the camera axis is not pointing perpendicular to the ground are oblique (PIX4D Documentation) (<https://support.pix4d.com/hc/en-us/articles/202559859-Vertical-vs-oblique-imagery>).

Automated orthorectification of vertical and near vertical aerial and seafloor imagery to produce georeferenced mosaics is common and numerous image acquisition techniques, processing approaches, software, and aerial imaging matching tools (i.e., Agisoft Metashape, Pix4d) have been created for this purpose (Figure 2.5). However, the orthorectification of more oblique imagery can result in greater geometric image distortion and reduce the accuracy of resulting maps, sometimes requiring manual intervention to assure proper georeferencing and consistent mosaicking of adjacent images (Wiedemann & Moré. 2012). To effectively orthorectify high-resolution aerial imagery collected at an oblique angle, a variety of processing and geometric correction steps must be followed. These main processing steps include alignment with the use of aerial triangulation (AT), the creation of a Digital Elevation Model (DEM), and the final creation of a digital orthomosaic image. Automated processing and mapping of oblique



aerial images has recently become a popular topic in the GIS and remote sensing research community (Verykokou and Ioannidis, 2018). Datasets containing oblique images have been studied in a variety of ways to evaluate whether an automated process for georeferencing and mapping those data is possible and to quantify the accuracy of the results (Nesbit and Hugenholtz, 2019; Zhou and Liu, 2015; Verykokou and Ioannidis, 2018; Petrie, 2009; Wiedemann and More, 2012). Although some demonstrated successes in this field of study have been published, the creation of orthomosaic images from oblique imagery using an automated digital photogrammetric workflow remains a challenge (Aicardi et al, 2016; Zhou and Liu 2015; Ludwig et al, 2020).

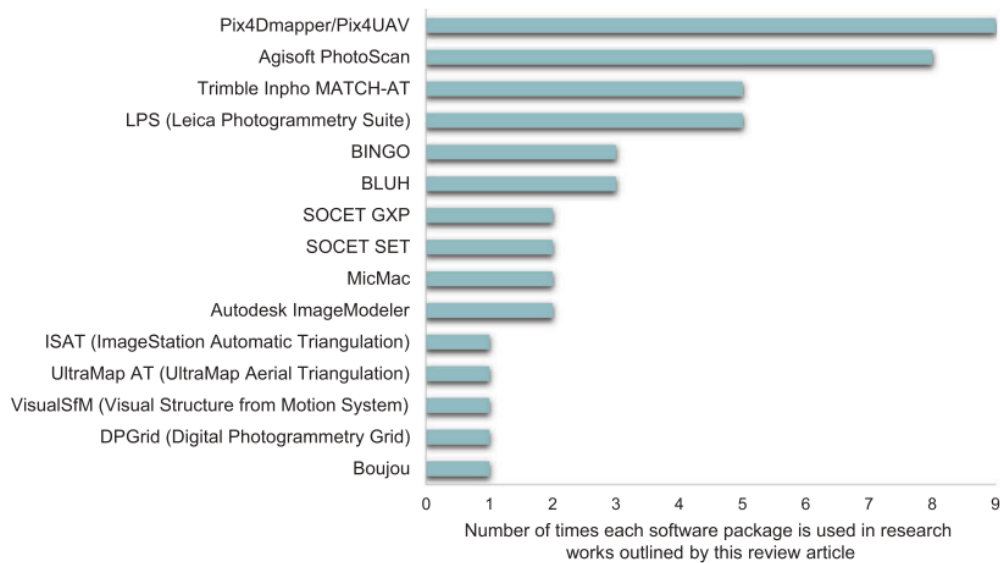


Figure 2.5 Comparison of Orthorectification Software

The above Figure shows the number of times specific aerial triangulation and orthorectification software packages were used in a research article that studied georeferencing procedures for oblique imagery (Verykokou and Ioannidis. 2018).

Advances in image processing techniques implemented in recently developed software (e.g., PIX4Dmapper and Agisoft Metashape) have improved the design and application of the complex geometric corrections necessary to accurately georeference oblique imagery, optimizing resulting orthomosaic image maps. Figure 2.5 indicates that both Pix4Dmapper/Pix4UAV and Agisoft Photoscan were frequently used for research involving the orthorectification of oblique imagery. Elkhachy (2021) evaluated the capacity of Agisoft Metashape and Pix4dmapper software to accurately generate orthorectified oblique images and found that Pix4dmapper generated more systematic errors and outliers than the Agisoft Metashape software. These results indicate that Agisoft Metashape software has a superior capacity to produce accurate orthorectified oblique images with the data used by Elkhachy (2021) . Accordingly, this thesis will use, in part, the processing approaches developed in Agisoft Metashape software in the image processing workflows presented in the Methods section.

## **2.4 Remotely Operated Vehicle Video Annotation and Viewshed Mapping**

Viewshed mapping is an approach to georeferencing the position of environmental features observed in oblique ROV video imagery (Ruby, 2017) that is fundamental to the analysis and results presented in this thesis. Additionally, it is a precursor to orthorectification of oblique ROV video imagery as presented in this thesis. Viewshed mapping was initially developed through NOAA funding as a means of improving the usability of the vast repository of ROV video data held by the NOAA National Centers for Environmental Information. It specifically addresses one of the primary challenges associated with ROV video data usability, which is finding a way to quickly and accurately convey to potential users, such as oceanographic researchers, what was observed in ROV dive video (Ruby, 2017). This is

imperative since watching hours of ROV video to determine its usefulness for a particular research application is not practical (Ruby, 2017).

An initial approach to solving the problem of ROV video data usability is video annotation in which disciplinary experts (e.g., marine biologists and geological oceanographers) record observed features in the ROV video (e.g., marine organisms and seafloor substrate) in real-time or after data collection. These expert annotations are associated with the time they were recorded and thus serve as a searchable index of observed features in ROV videos that can direct users to a specific time in the video when the listed feature was observed (Ruby 2017). NOAA was an early implementer of expert ROV video annotation and has been responsible for numerous developments in the application of the approach. In the early 2000s, NOAA initiated a program of deep-water expeditions aboard the ship *Okeanos Explorer* to explore a wide range of benthic environments using deep sea video collected with a high-resolution underwater camera that was mounted on the ROV Deep Discoverer (Medley, 2018). NOAA scientists and participating researchers aboard the *Okeanos Explorer*, as well as researchers on shore participating remotely via satellite telepresence, observed the video collected during each ROV dive and provided expert annotation of observed geological and biological features (Ruby, 2017). The observed annotations were classified using the widely adopted Coastal and Marine Ecological Classification Standard (CMECS) and logged using *Seatube V2* software.

CMECS is a classification system that was published by the Federal Geographic Data Committee in 2012 (Federal Geographic Data Committee, 2012; U.S. Geological Survey, 2012). CMECS is a catalog of environmental and ecological terms that provides a standardized scheme for the classification of coastal and marine observations. (NOAA Integrated Ocean & Coastal Mapping, 2022). This thesis specifically focuses on substrate annotations, which use the Coastal

Marine Ecological Classification Standard (CMECS) to classify the substrate material that is observed along the seafloor in the ROV video data for each dive.

Ocean Networks Canada (ONC) created the web-based Seatube V2 interface to provide the public with the ability to view both underwater ROV dive videos and associated time-stamped and georeferenced CMECS annotations that were made during the ROV dives. This video portal is an advanced tool constructed to provide external scientists and outside citizens with a platform for querying, discovering, and analyzing video data that was collected aboard the *Okeanos Explorer* and other ROV platforms. Seatube V2 allows users to download ROV video files as well as coincident dive annotation files and ROV navigation files in .csv format. The ROV navigation files contain attributes such as ROV position (coordinates, depth, and altitude) and ROV attitude (heading, pitch, and roll) sampled with a frequency of 1 Hz or greater.

Although the annotation methods described above significantly improved the usability of ROV video data, they could not visually display the location of the annotated features and the spatial relationship between them. To address this shortcoming, Ruby (2017) developed two different types of geospatial analysis methods to map data collected during these dives, a buffered representation, and a viewshed wedge. The buffered approach specifically maps the annotated CMECS class to a circle around the ROV position at the point in time when the annotation was made. The radius of the circle approximates the maximum viewable range of the ROV camera. This approach effectively represents an area on the seafloor that encompasses all theoretically possible locations for the annotated feature class. To refine this Ruby (2017) developed a “viewshed” approach, which reduces the circle of the buffer to a wedge-shaped polygon, approximating the footprint of the ROV camera field of view, oriented in the direction the camera is pointing (Ruby, 2017). In this thesis, the viewshed mapping methods developed by

Ruby (2017) are expanded, refined and automated, making the approach fully open source, and compatible with the current annotation software that is being used by NOAA OE for ROV dives (i.e., *Seatube V2* and *Seascribe*).

The viewshed approach introduced by Ruby (2017), and further developed in this thesis, is based on estimating the seafloor area that lies within the field of view of an ROV video frame recorded at the point in time an annotation is made. The natural extension of such an approach is to project (orthorectify) the actual video frame image on the seafloor rather than using a wedge-shaped polygon representing the extent of seafloor coverage for the frame). The ability to accurately estimate the area of a benthic surface encompassed in an ROV video frame has previously been a challenge in seafloor surveys (Dias et al, 2015). Initial efforts to do this were presented by Lundsten (2010), who used still image frames from a video collected using a forward-facing camera on an ROV to visually analyze physical changes to whale fall carcasses over time. This thesis advances this approach, applying orthorectification techniques developed for highly oblique sUAS video data to ROV video data to yield viable seafloor orthomosaic image maps from ROV video data.

## CHAPTER III

### METHODS

#### 3.1 ROV Data Processing Framework

The first objective of this thesis is the development of an automated data processing framework for the generation of georeferenced maps from ROV video imagery and its derivative data. Specifically, a suite of integrated data processing and GIS workflows were created as a methodological tool to digitally map the spatial distribution of seafloor substrate classes observed in oblique video imagery collected during ROV dives to enable a quantitative geospatial analysis of those data. Although this thesis focuses on seafloor substrate and an example case, it should be noted that the presented methodology could be used to map the spatial location of any CMECS annotation class recorded during a ROV dive, not just substrate.

All ROV data used in this thesis were acquired during deep water seafloor exploration dives conducted between 2018 and 2019 by the ROV Deep Discoverer aboard NOAA research vessel *Okeanos Explorer*. These dives occurred during cruises EX1803 (the Gulf of Mexico in April-May of 2018), EX1806 (Atlantic Ocean in June-July of 2018), and EX1903L2 (the Atlantic Ocean in June of 2019) (Figure 3.1). The ROV dives typically lasted between 4 and 7 hours with a majority of the dive spent exploring the seafloor with the ROV camera. Expert substrate annotations and video data from each dive were downloaded from Ocean Networks Canada's (ONC) open-source software web interface. Each annotation log contains dive annotation navigation parameters (latitude, longitude, and heading) and ecological annotations

(substrate, oxygen, sea water temperature, and salinity). All ROV data files were downloaded from NOAA OER Digital Atlas. These data files contain navigational information of the ROV (altitude, latitude, and longitude) sampled at a frequency of 1Hz.

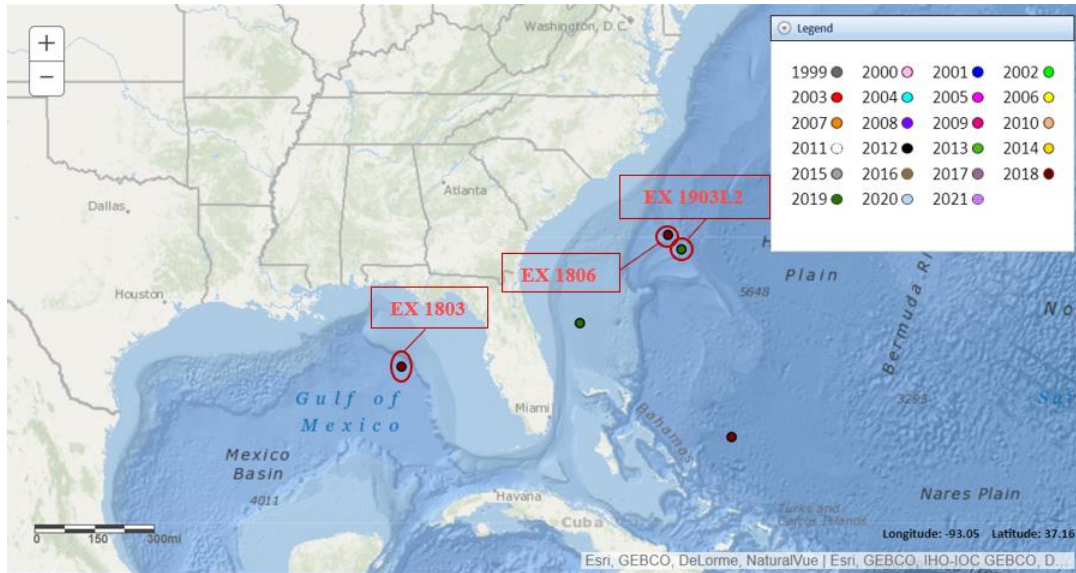


Figure 3.1 NOAA ROV *Okeanos Explorer* Study Sites

EX1803 – Gulf of Mexico, EX1806 – Atlantic Ocean, and EX1903L2 – Atlantic Ocean ROV dive sites (National Oceanic and Atmospheric Administration –Office of Ocean Exploration and Research).

### 3.1.1 Extraction of Annotation & Navigation Data

For each of the three *Okeanos Explorer* expeditions, a series of individual dives were conducted. A total of 51 dive annotation files were downloaded from *Seatube*.

- EX1803 – 15 dives
- EX1806 – 17 dives
- EX1903L2 – 19 dives

With the use of ONC's *Seatube V2*, annotations recorded during each dive were downloaded in .csv format. A variety of metadata and data are viewable in each annotation .csv file (i.e., dive id, dive name, cruise name, start date, end date, annotation id, substrate, depth, heading, latitude and longitude, oxygen, sea water temperature, and salinity) along with the contact information for the researcher who created the annotation during the ROV dive. For the automated mapping of each dive, the latitude, longitude, and heading were required to accurately indicate the position of the ROV when each dive annotation was recorded during a dive. These data were integrated with a separate file containing ROV vehicle navigation and attitude data sampled at 1 Hz to improve the accuracy of the ROV path and represent the full extent of the ROV dive beyond just the points where annotations were made.

### **3.1.2 Revision of Substrate Annotation to Fine Resolution CMECS Classes**

The expert seafloor substrate annotations for Deep Discoverer ROV dives conducted on *Okeanos Explorer* expeditions in 2019 (EX 1903L2) were directly entered into *Seatube* by participating scientists using CMECS-compliant terminology (Figure 3.4) and were processed at fine resolutions, thus allowing habitat heterogeneity and complexities that are experienced at finer scales to be classified (Kington, 2018). However, for dives conducted in 2018 (EX 1803 and 1806), the annotations were not entered using a formal classification system. For example – if a substrate annotation recorded within a dive annotation file for EX 1806 was annotated as “Primary Unconsolidated Secondary Unconsolidated” a simple spelling error or misuse of space within the substrate annotation would not correctly classify the particular substrate that was being viewed at that location during the dive. This is visible throughout a variety of different dive annotation files collected from EX 1803 and EX 1806. This means substrate annotation terminology recorded in *Seatube* can be inconsistent and repetitive and it is necessary to



standardize the annotations by converting them into CMECS-compliant format. Accordingly, a Python script was created to convert and assign seafloor substrate annotations recorded in an *ad hoc* legacy format into a new format compliant with the current CMECS standard. Thus, creating a more efficient and accurate classification schema and process for the annotated substrate categories found in the dive annotation records. The substrate annotations are based on four different categories referring to the primary (>50%) and secondary substrates: Hard/Hard, Hard/Soft, Soft/Hard, and Soft/Soft (Bassett et al. 2017) (Figure 3.4). The described Python script converts those annotations into CMECS-compliant substrate units following the scheme seen below in Figure 3.4. The full Python code for this revision process is presented in Appendix A and the full standard operating procedure document describing its application is presented in Appendix B.

• Fine Unconsolidated Mineral Substrate	> Fine: Fine
• Fine and Coarse Unconsolidated Mineral Substrate	> Fine:Coarse, Coarse:Fine
• Coarse Unconsolidated Mineral Substrate*	> Coarse:Coarse
• Rock & Fine Unconsolidated Mineral Substrate	> Fine:Rock, Rock:Fine
• Rock & Coarse Unconsolidated Mineral Substrate	> Rock:Coarse, Coarse:Rock
• Rock Substrate	> Rock:Rock
• Unobserved or Unknown Substrate	> Unknown

\* consider modifiers to 'Coarse' to represent *shell hash* and *coral rubble*

Figure 3.2 CMECS compliant substrate scheme (Basset et al, 2017)

### 3.1.3 Generation of ROV Viewshed Maps

The first step in the workflow for creating substrate maps of each ROV dive was the extraction of ROV navigational data that is stored in both the 1Hz datasets that were previously downloaded from the NOAA OER Digital Atlas, along with annotation data downloaded from

*Seatube V2*. The full Python code for this revision process is presented in Appendix A and the full standard operating procedure document describing its application is presented in Appendix B. Once extracted from the 1Hz dataset, the coordinates of the ROV for each second of a given dive were connected to create a line representing the complete dive path of the vehicle. The full Python code for this revision process is presented in Appendix A and the full standard operating procedure document describing its application is presented in Appendix B.

The next step in the workflow for creating substrate maps of each ROV dive is the creation of viewshed polygons. Viewshed polygons are georeferenced wedge-shaped polygons that approximate the area of seafloor imaged in each frame of the ROV video data. The wedges have an angle and a radius based on the view angle and focal length of the ROV camera on Deep Discoverer, approximations to the mean range of the images based on light attenuation in the observed environments, and an assumption of a flat seafloor (Figure 3.3). This viewshed application restricts the mapped area to a wedge-shaped polygon that expands 5 meters in the direction of the ROV's heading showing the area that was most likely viewed by the Deep Discoverer's main, forward-facing camera (Ruby, 2017) (Figure 3.3). See Ruby (2017) for a detailed review of the considerations used to determine representative viewshed geometry for the primary camera on the Deep Discoverer.

Once the location of each substrate annotation along the ROV dive path was created with the "create points layer from table" function, the "wedge buffer" function was used to create a set of viewshed polygons that represent the seafloor area observed by scientists, when they made the substrate annotations (Figure 3.4). Next, adjacent viewshed polygons with matching substrate classes are merged and plotted with uniform color to represent the seafloor spatial extent of each identified CMECS substrate class. An example map is presented in Figure 3.5. Maps produced

for every ROV dive included in this thesis are in Appendix C. The complete workflow created to produce ROV viewshed maps is presented in Figure 3.6.

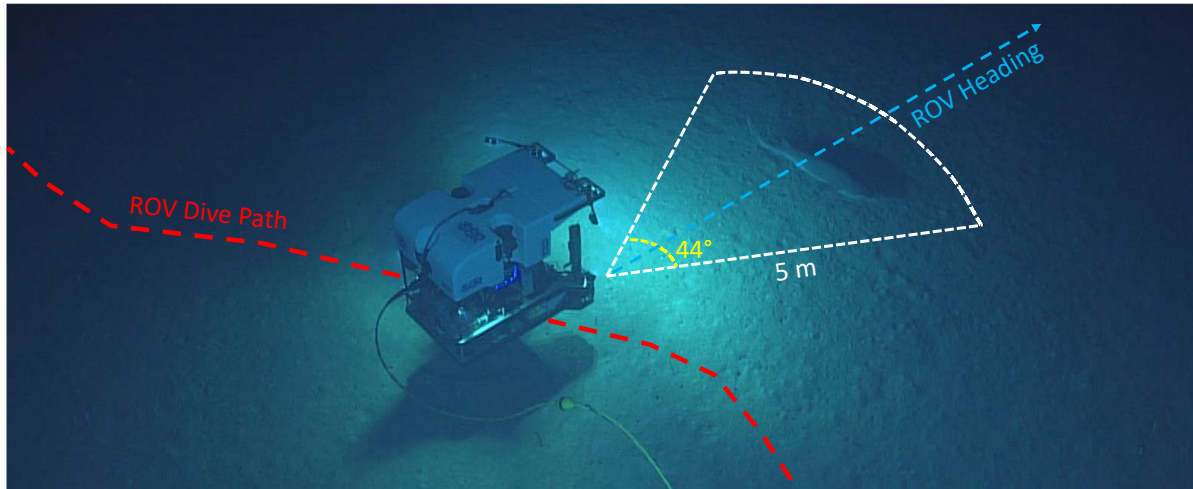


Figure 3.3 Deep Discoverer Viewshed Concept

The above image of the ROV Deep Discoverer shows a conceptual rendering of a dive track (red) and seafloor viewshed wedge polygon (white) approximating the area of the seafloor viewed in one frame of the ROV video.

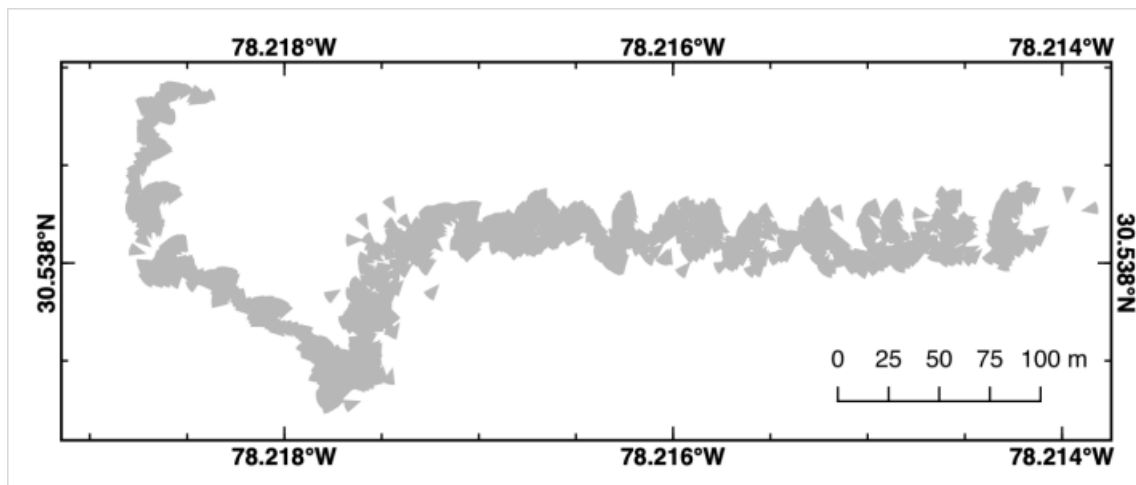


Figure 3.4 Viewshed Mapping Approach

The above map illustrates an example of mapped ROV video frame “viewsheds” along the path of EX 1903 L2 Dive 05. Collectively, the gray area represents the seafloor area imaged during the dive.

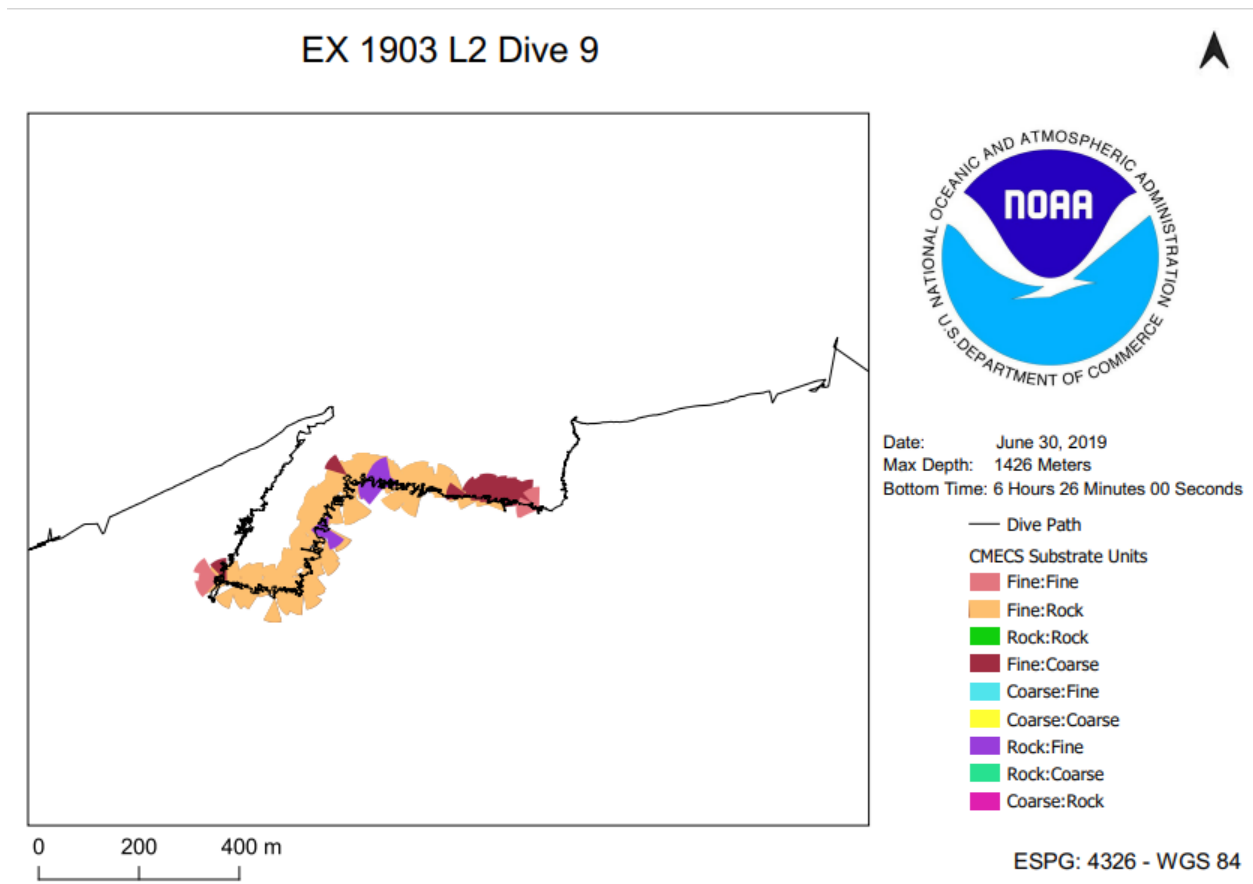


Figure 3.5 Example of the final ROV substrate map of EX 1903 L2 Dive 09

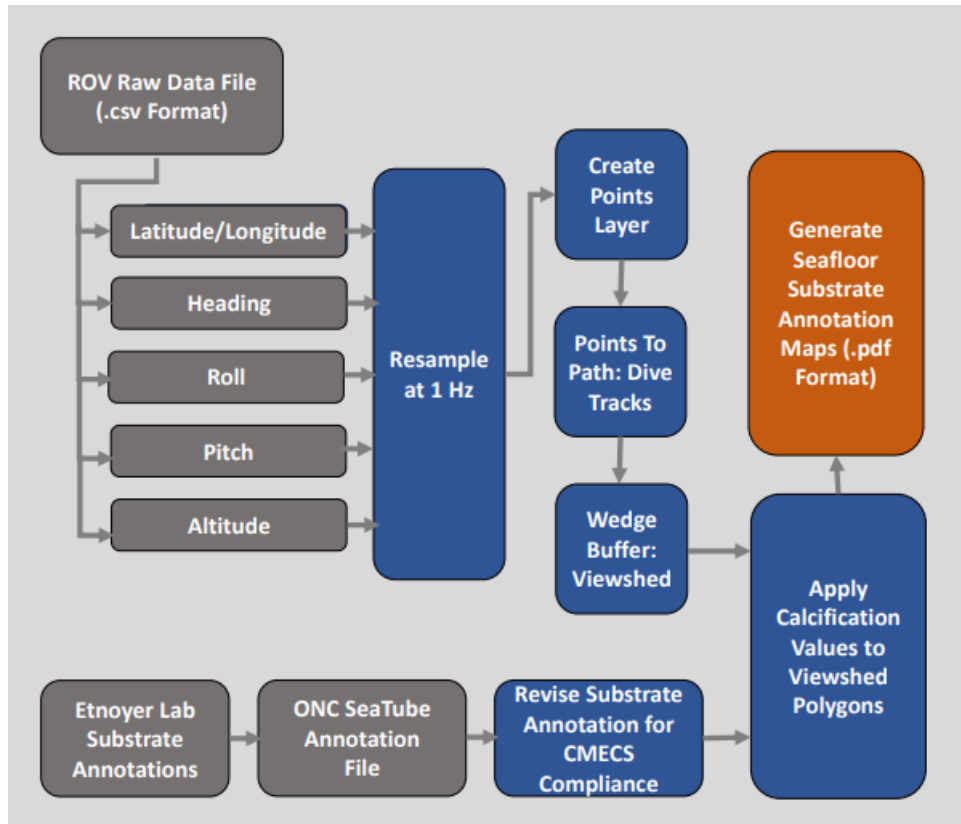


Figure 3.6 General Automated ROV Viewshed Mapping Workflow

### 3.2 Evaluation of ROV Viewshed Maps

The second objective of this thesis is to quantitatively evaluate the spatial agreement between produced ROV maps and those produced with conventional sonar image mapping approaches and platforms. The efficacy of the presented ROV digital mapping approach was evaluated by comparing resultant substrate viewshed maps to coincident seafloor backscatter data. This approach was selected because it was not possible to evaluate the absolute spatial accuracy of resultant ROV maps due to the infeasibility of collecting the equivalent of GPS ground truth points on the seafloor at the location of the ROV dive. Specifically, the degree to which ROV mapped seafloor substrate classification polygons were spatially coincident with

seafloor substrate classes, as indicated by independent sonar acoustic backscatter maps of the seafloor, was assessed. To do this, 10 of the 51 ROV final substrate maps were chosen based on the variety of substrate type that was identified and classified along the seafloor during each dive. These 10 dives were then overlayed onto acoustic backscatter intensity data of the same dive site, collected with a multibeam sonar on the ship *Okeanos Explorer*.

Acoustic backscatter intensity is a measurement of the amount of acoustic energy reflected by the seafloor. For a multibeam sonar system, the amount of energy in the projected seismic pulse (amplitude of generated sound wave) is known, as is the amount of energy in the returning pulse (amplitude of reflected sound wave) after it has been reflected by the seafloor. The ratio of these values reflects the degree to which the seafloor “scatters” acoustic energy. This is a function of seafloor hardness (substrate composition e.g. rock, sand, mud) as well as seafloor roughness (substrate mean grain size and surface texture). Accordingly, mapped seafloor backscatter intensity is a common method used to classifying the general properties and spatial distribution of seafloor substrate (i.e., rock, sand, mud, coral, seagrass) (Lurton, 2010; United States Geological Survey, 2014). In general, harder substrates like a rock will reflect more acoustic energy (sound) and thus have a relatively higher backscatter intensity than softer substrates like mud. Although there is a certain amount of uncertainty inherent in relating the acoustic reflectivity of the seafloor to a specific substrate class, seafloor backscatter intensity measured with a ship-based multibeam sonar represents the best way of making such observations without physical sampling and is the widely accepted conventional approach for mapping seafloor substrate type (Lurton, 2010). Thus, it is the most appropriate independent data for comparison to the viewshed substrate maps generated from integrated ROV video and annotation data. To evaluate the agreement between the viewshed mapped CMECS substrate

class and measured seafloor backscatter intensity, a subset of backscatter values from within each viewshed substrate polygon was created. The frequency distribution of these backscatter values was then plotted for each coincident CMECS substrate class polygon on a boxplot. The resulting boxplots were evaluated to determine if the observed backscatter intensity properties (e.g. mean) are consistent with the assigned CMECS classification in terms of relative magnitude. For example, the mean backscatter intensity of a hard and rough substrate like rock will be greater than a soft smooth substrate like fine sediment (Lurton, 2010). If such a relationship is demonstrated in the boxplot, this suggests that the assigned CMECS class is at least relatively consistent the observed seafloor properties.

### **3.3 sUAS Data Processing Framework**

The third objective of this thesis is the development of an automated data processing framework for the generation of georeferenced maps from sUAS video imagery. Given the approach to mapping ROV data presented above, the same or a similar mapping approach was applied to video data collected from other unmanned vehicle platforms, particularly sUAS. A key challenge to this approach was the fact that a standardized expert annotation process does not exist for sUAS video data as it does for ROV Deep Discoverer data. Therefore, I investigated the potential for directly mapping (orthorectified) oblique sUAS video frames rather than their representative viewsheds.

Two separate datasets were used for the sUAS portion of this thesis to evaluate the potential that sUAS have for the orthorectification of oblique imagery, and how this type of photogrammetric mapping approach can potentially be applied to both recreational and professional sUAS operations. All sUAS data in this thesis were collected using a DJI Phantom 4 Pro drone. Data collection occurred in two different locations, North Farm at Mississippi State

University (Figure 3.7), and the Mississippi Gulf Coast (Figure 3.8). A series of flights were conducted in both study areas; however, two broadly representative flights for each study area were selected for video frame extraction and orthorectification. These flights were conducted without the use of grid flight patterns which are commonly employed in surveys conducted with UAS for commercial and research applications. Data analyzed from these sUAS flights were limited to an RGB camera sensor attached to the sUAS to test the accuracy of oblique mapping when true color RGB images are orthorectified. All flight navigation parameters (latitude, longitude, GPS time, altitude, fly time, roll, pitch, and yaw) are downloaded from the sUAS and converted via a web interface (Phantom Help) for the generation of flight logs in .csv format.



## North Farm sUAS Flights Mississippi State University

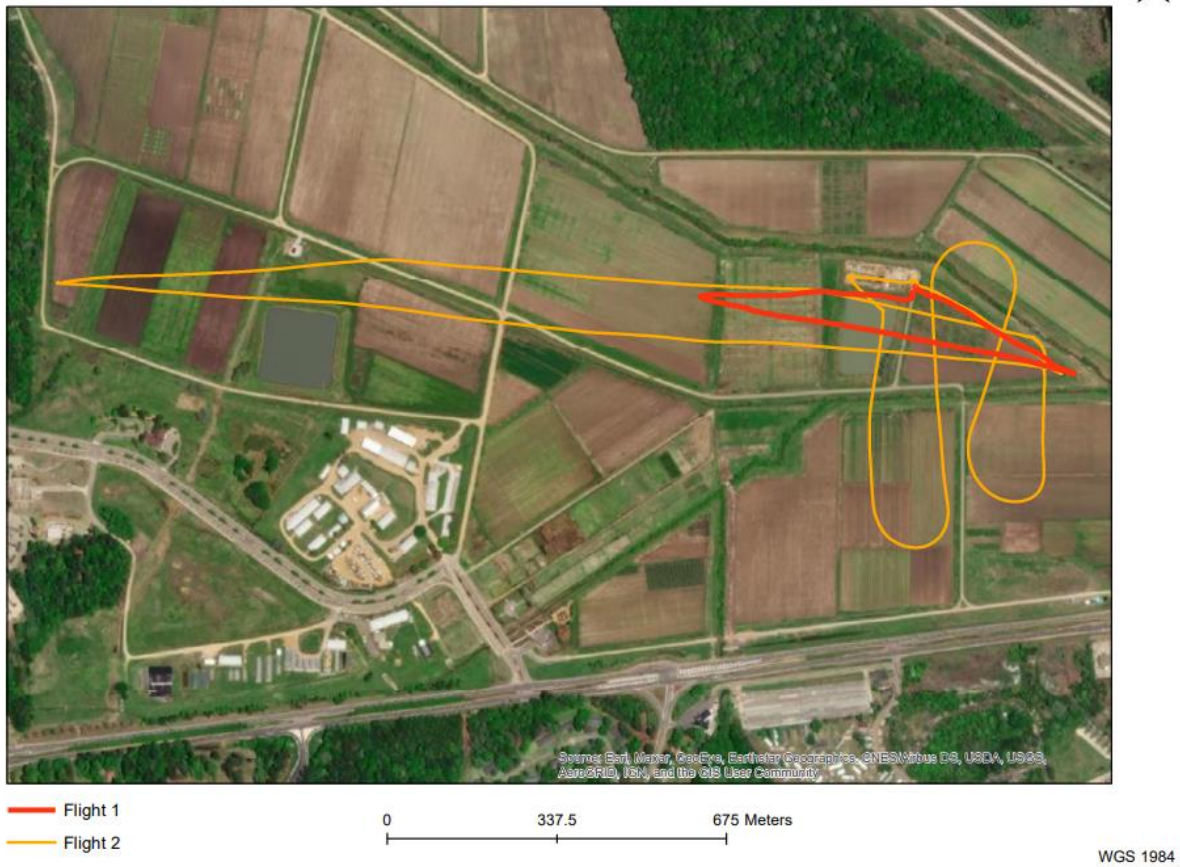


Figure 3.7 Mississippi State University North Farm Study Site with flight track lines.



Figure 3.8 Mississippi Gulf Coast Study Site (Waveland, MS)

Both sUAS flights studied in the Mississippi Gulf Coast Study Site were extracted from the flight track lines portrayed in the above map (Figure 3.15).

### 3.3.1 Video Frame extraction & timestamp scripts

Several different timestamp scripts were written in Python to effectively extract keyframes and timestamps from the sUAS and ROV video files. However, the creation of three separate video frame extraction and timestamp scripts was necessary for effective data preparation due to certain varying limitations found within all three unmanned vehicle navigation logs. Although all the Python scripts are designed to effectively produce the same output, each one of them contains different parameters and functions to work around these different limitations that were prevalent in the three navigation logs. Before executing the extraction and timestamp commands used in each Python script, FFmpeg, a command line tool used for the

extraction of both audio and video formats was used to extract the still frames from each video file.

Three Python scripts were created to process and condition the raw sUAS (and in one case ROV) vehicle data in a manner necessary to allow orthorectification of the oblique video imagery data. The goal for each of the three scripts is to effectively synchronize latitude, longitude, altitude, and depth with the video frames that are extracted from each of the video datasets that are collected with unmanned vehicle systems.

Initially, the sUAS position and attitude data associated with each video frame had to be determined. This was accomplished by querying the vehicle position and attitude data file for the specific point in time each video frame was recorded. The first initial section of code used FFmpeg to effectively extract key frames from the DJI MOV file. Once this was complete the Python script then generated timestamps from the DJI MOV file using FFprobe. FFprobe can extract information from a MOV file and transform it into a human, machine-readable format. The Python code then moves on to the task of finding the start of the video file. FFprobe is used again to perform this task. The Python code was then designed to open the flight log .CSV file to effectively synchronize (i.e., gpsTime, latIndex, lonIndex, altIndex, flyTime, rollIndex, pitchIndex, and yawIndex).

The second Python script was created to effectively produce the same results as the original keyframe and timestamp script described previously. Similarly, the sUAS position and attitude navigational data associated with each video frame had to be known. This again was accomplished by querying the vehicle position and attitude data file for the specific point in time each video frame was recorded. An example of this Python script is viewable in Appendix E at the end of the document.

The third and final frame extraction and timestamp script was specifically created for the extraction of frames recorded during the ROV Deepwater dive videos that were previously downloaded from *Seatube V2*. The general idea and script construction of the ROV frame extraction and timestamp script are similar to that of the previous two. As described previously, this extraction is done with the FFmpeg function. The user can manually set the number of frames that are extracted per second. When the script successfully extracted one frame every two seconds, a list of all the extracted frames was generated.

### **3.3.2 Agisoft Metashape Orthorectification**

Orthorectification methods and workflows used to effectively orthorectify and georeference each of the extracted video frames that were collected with the use of the ROV and sUAS platforms were performed in Agisoft Metashape. Agisoft Metashape is a software program that executes photogrammetric modeling and processing of high-resolution digital imagery. This advanced software has the ability to produce high-resolution products that can thus be used in a variety of geospatial analyses. Several photogrammetry processing steps were performed on both the sUAS and ROV Datasets within Agisoft Metashape to effectively produce accurate orthomosaic images of the study areas. The first and initial step that was required to begin the orthorectification process for both the ROV and sUAS datasets was alignment. The alignment process in Agisoft Metashape involves aerial triangulation (AT) and bundle block adjustment (BBA). During this stage, feature points found within the extracted timestamped images that were generated in the previous video frame and time stamp extraction Python script are identified. When these feature points are identified within each of the images, they are then matched across the images into a series of tie points. Another major data manipulation process

executed in this stage was defining the position of the camera for each image and automatically adjusting the camera orientation calibration parameters (i.e., internal (IO) and external (EO)).

Once the first data processing step was complete, both the sUAS and ROV datasets were then visualized in the form of a sparse point cloud and a series of camera positions. Although the generation of the sparse point cloud is not necessary for the specific processing workflow described here, it can be used for to rapidly generate sparse point cloud-based surface reconstructions, which may be useful for an initial evaluation of collected data quality. Moreover, it can be used for further analyses in other exterior programs. For example, a sparse point cloud model can be used in a 3D editor software program as a reference model for further generation of a depth map.

The second data processing step that was carried out on both the sUAS and ROV datasets was the generation of a dense point cloud layer and a Digital Elevation Model (DEM) through Structure from Motion (SfM) range imaging techniques. These dense point cloud layers were constructed based solely on the estimated camera positions and still images that were collected by the sUAS and ROV platforms. For both the sUAS and ROV datasets, the DEMs for all four of the final orthomosaic images were constructed based on the resulting dense point cloud layer data. When a DEM is constructed from a dense point cloud layer a variety of terrain ground features and surface objects can be visible (i.e., light poles, piers, houses, trees, and seawalls) thus creating both digital surface models (DSM), and digital terrain models (DTM). A variety of these features are visible in the final orthomosaic images presented in the Results section of this thesis.

The third and final data processing step that was carried out was the production of the final orthomosaic image. The final orthomosaic images that were produced for both the sUAS

and ROV datasets were generated by accurately projecting the extracted timestamp video frames by their EO and IO camera orientation data onto the surface of the DEM that was previously generated by the dense point cloud layer. The collective ROV video data processing workflow is presented in Figure 3.9.

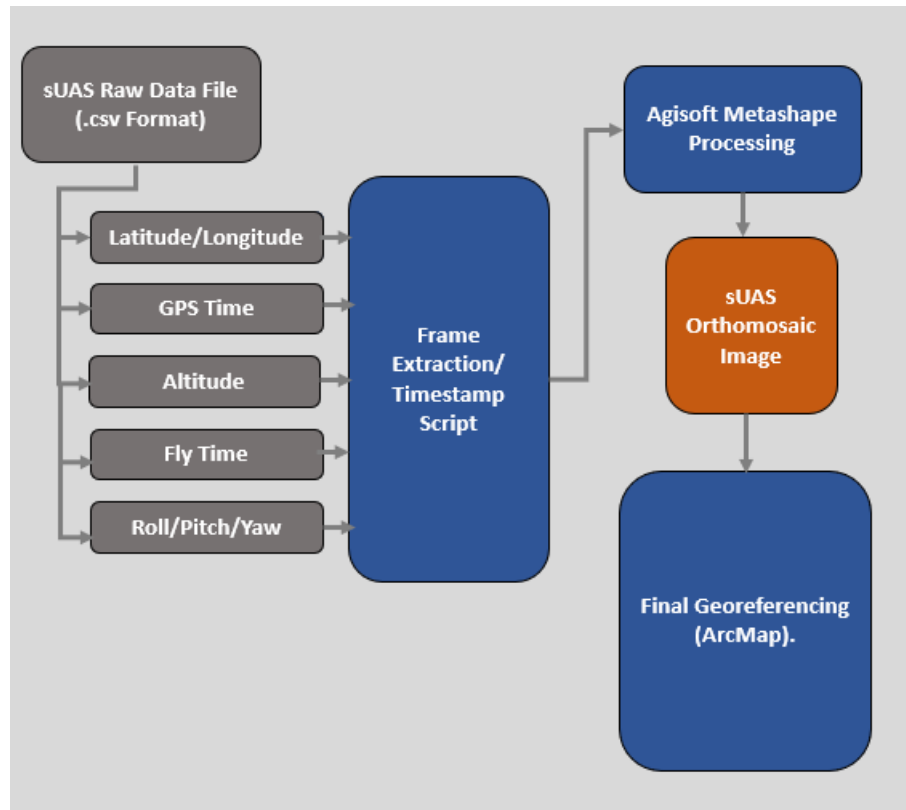


Figure 3.9 General sUAS Workflow

### 3.3.3 Evaluation of Spatial Accuracy

The fourth objective of this thesis was to quantitatively evaluate the spatial accuracy of resulting sUAS maps and determine their comparability to that of maps produced with conventional remote sensing platforms. The accuracy of generated sUAS maps was determined

through analysis of the Root Mean Square Error (RMSE) of the horizontal position of features in the resultant maps relative to their ground-truth position as determined by direct GPS measurement and conventional remote sensing imagery. Two different techniques were used to analyze the RMSE between the oblique images that were collected using the sUAS at the Mississippi State University North Farm study site.

The first approach evaluated the RMSE of generated sUAS maps relative to GPS measured ground control points. For each of the two flights that were conducted at North Farm, a set of ground control points (GCP's) were collected using a Trimble Geo7x GPS unit. For sUAS North Farm flight one (Figure 4.9) 39 ground control points were collected. For the second sUAS North Farm flight (Figure 4.11) 44 GCP's were collected. An analysis was then conducted to quantify the RMSE between the sUAS map created from the orthorectified oblique images and the ground control points that were collected at North Farm using the Trimble Geo7x. The goal of this analysis was to determine the spatial accuracy of the generated sUAS images.

The second approach evaluated the RMSE of generated oblique imagery sUAS maps relative to a three-centimeter resolution aerial image of the study area collected with a more conventional nadir oriented camera (2021 MSU OrthoImagery Project, Mississippi State University; J. Cartwright Per. Comm.) This was done by creating a series of points in ESRI's ArcMap software for the sUAS and conventional aerial imagery that mark matching conspicuous features identified in both. An RSME value was determined by measuring the spatial offset of matching features between the maps. This assessment evaluates the degree of agreement between the sUAS oblique imagery derived maps and conventional nadir aerial image of the study area. For sUAS North Farm Flight one and sUAS North Farm Flight two (Figure 4.14 and Figure 4.15) 34 ground features were identified within both the three-centimeter resolution map and the

orthorectified sUAS maps. The goal of this analysis was to determine the comparability of generated sUAS maps with maps generated with conventional (nadir aerial imagery) remote sensing platforms.

The analysis that was conducted on the Mississippi Gulf Coast study site (Figure 4.16 and Figure 4.17) was done by comparing the RMSE between ground features found within both the sUAS orthorectified images and a 0.5 meter Google Earth satellite image of the study area produced by Esri. A total of 15 ground features were identified and compared for both Mississippi Gulf Coast sUAS orthorectified maps. The goal of this analysis was to determine the comparability of generated sUAS maps with maps generated with conventional remote sensing platforms specifically in a coastal environment.



## CHAPTER IV

### RESULTS

#### 4.1 Automated ROV Substrate Maps

The ROV automated mapping methods that were presented in the preceding chapter were applied to a total of 51 mappable ROV dives. Each of the resulting final ROV substrate maps effectively displays the ROV dive path and CMECS classified seafloor substrate features that are observed in the video data that was acquired during ROV dives performed by the ROV Deep Discoverer aboard the *Okeanos Explorer*. Each map contains cartographic features to accurately represent the study area surveyed by the ROV Deep Discoverer (i.e., Title, Legend, Scale bar, north arrow, Date, Max Depth, and Bottom time). All final ROV substrate maps that were created from the automated mapping process are presented in Appendix C. Standard operation procedure (SOP) documentation was created for the entire automated ROV automated mapping process. A total of five detailed SOPs and three main Python automated mapping scripts were created to effectively execute automate mapping workflow.

- SOP #1 – This SOP describes how to download the ROV video annotation and vehicle data from *Seatube*. These data are necessary to generate substrate maps for an ROV dive.
- SOP #2 – The second SOP describes how to download the 1Hz ROV vehicle data from NOAA’s OER Digital Atlas. Like the ROV annotation data, ROV navigation data recorded with a frequency of 1 Hz are necessary to generate substrate maps for an ROV dive.

- SOP #3- The third SOP describes the process of executing the first Python script used in the automated mapping workflow. The Python script *CMECS classification script.py* converts seafloor substrate annotations recorded in legacy format into a new format compliant with the current CMECS standard.
- SOP #4 – The fourth SOP describes the second Python script that is executed in the automated mapping workflow. The *Mapping Script.py* is used to generate digital shapefiles of the ROV dive path (line feature) as well as the classified seafloor viewsheds (polygon features).
- SOP #5 – The final SOP documented describes the use of the third and final Python script *Map production script.py*, which is executed to create the final ROV substrate map in .pdf format.

All SOP documents that were created from the automated mapping process are presented in Appendix B of this document.

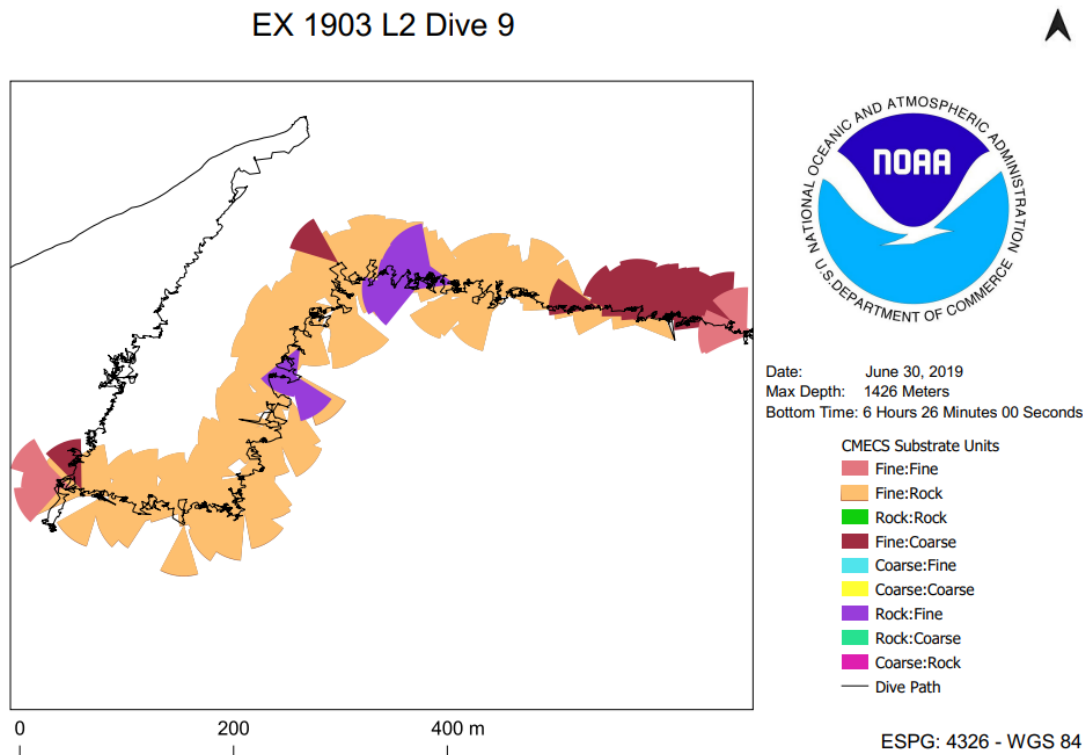


Figure 4.1 Seafloor substrate map generated from observations recorded during EX 1903 L2 Dive 09

The above Figure portrays an example ROV seafloor substrate map representing substrate features that were observed during EX 1903 L2 Dive 09 by the ROV Deep Discoverer. Each map contains (a title, scalebar, legend, north arrow, date, max depth, and bottom time).

## 4.2 ROV Substrate Map Evaluation

For the included backscatter maps such as the one depicted in Figure 4.2, brighter tones indicate a stronger backscatter intensity, suggesting the presence of hard geologic features like an exposed rock within the dive site or study area. Conversely, darker tones indicate weaker backscatter intensity, suggesting the presence of softer seafloor features such as sand and mud. Maps of seafloor substrate as determined from ROV video data are plotted over backscatter data

collect by conventional ship-based sonar to facilitate comparison between both. Maps of this nature for the remaining nine ROV dives are presented in Appendix D.

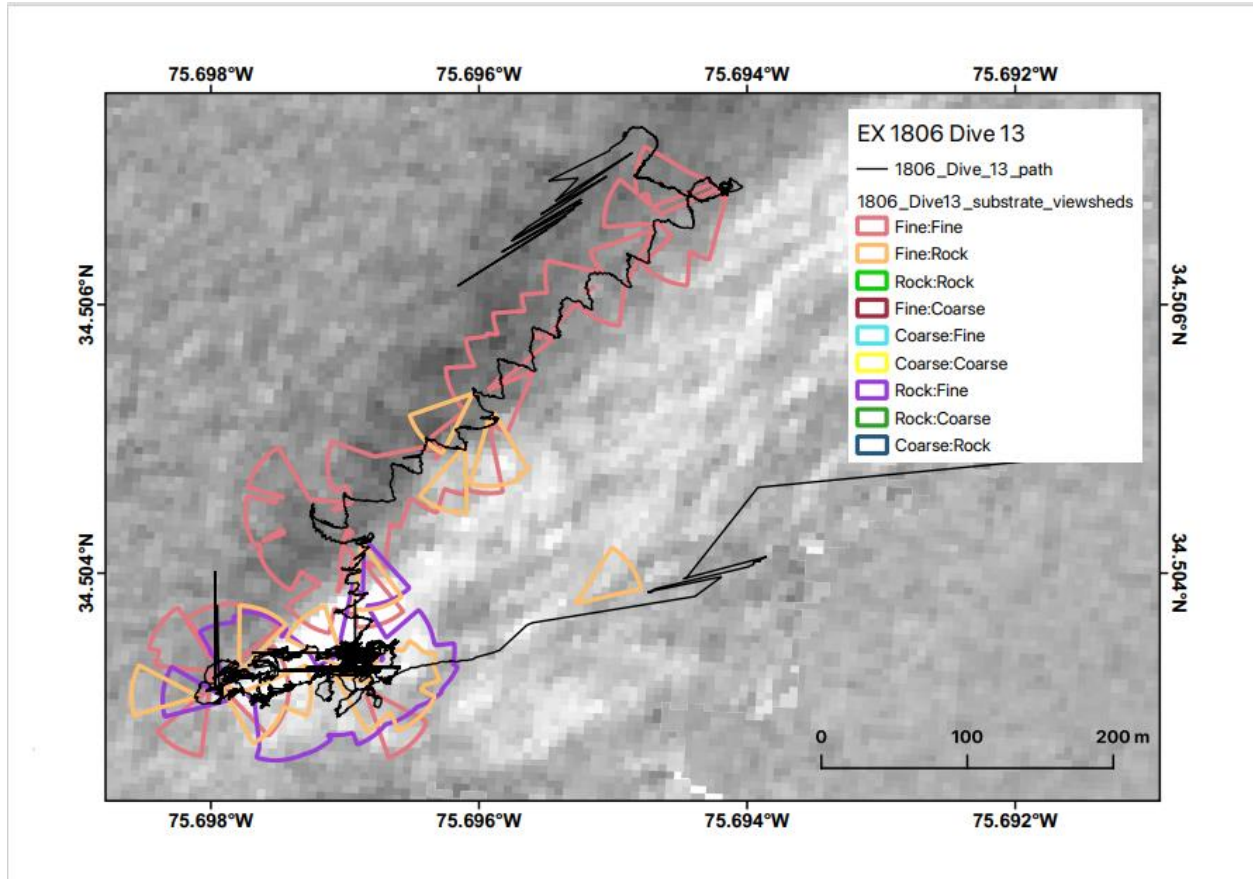


Figure 4.2 Backscatter Comparison of EX 1806 Dive 13

The above map shows seafloor substrate classification for EX 1806 Dive 13 overlaid onto acoustic backscatter data collected aboard the *Okeanos Explorer* in 2018. As described in the previous section darker tones within backscatter data indicate weaker intensity, thus indicating a high volume and presence of soft seafloor sediment. In the image above the CMECS classification “Fine:Fine” matches dark backscatter intensity features, thus indicating that the presence of soft fine material is valid and frequent throughout EX 1806 Dive 13.

To evaluate the agreement between the viewshed mapped substrate and backscatter mapped substrate, a subset of backscatter values from within each viewshed substrate polygon

was created. The frequency distribution of these backscatter values was then plotted for each coincident CMECS substrate class polygon on a boxplot. These boxplots demonstrate the degree of agreement between the distribution of backscatter intensity and derived substrate polygons, allowing for evaluation of the spatial agreement between produced ROV maps and those produced with conventional sonar image mapping approaches and platforms. Ideally, harder (high reflectivity) substrate, such as rock, will have a higher median backscatter value than softer (lower reflectivity) substrate such as fine sediment (Lurton, 2010). An example boxplot of backscatter intensity distribution binned by determined CMECS substrate class from ROV dive 13 on expedition EX 1806 (Figure 4.2) is shown in Figure 4.3. Boxplots of this nature for the remaining nine dives are presented in Appendix D.

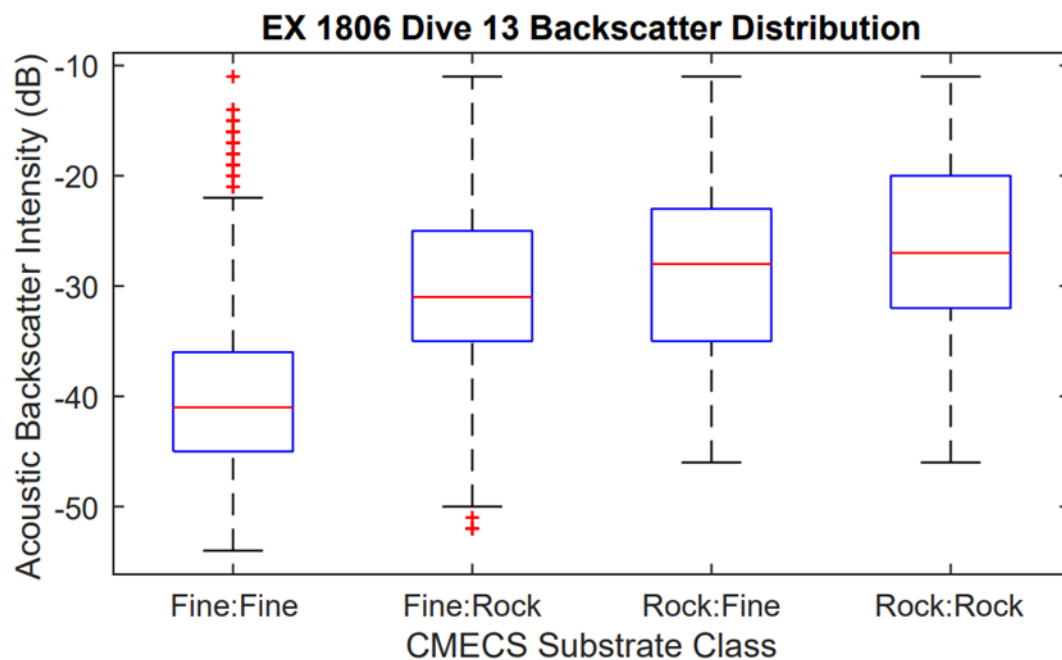


Figure 4.3 Example Boxplot of EX 1806 Dive 13 Backscatter Distribution

The above boxplot indicates the frequency distribution of backscatter intensity values spatially coincident with the labeled CMECS classes for EX 1806 Dive 13.

### **4.3 sUAS Orthorectification**

The sUAS video frame and timestamp extraction Python scripts, and Agisoft Metashape Orthorectification practices described within the previous methods section were applied to four sUAS flights that surveyed two different study sites (i.e., Mississippi State University North Farm, and Mississippi Gulf Coast). Two flights for each of the study sites were selected to test the potential efficacy of orthorectified oblique video images that were generated from the inflight videos. Neither single-grid nor double-grid flight patterns were followed while collecting the aerial videography. A variety of free roam, and open flight patterns were flown without the use of ground control points to gain a better understanding of how the presented image mapping approach would handle the variety of geometric distortions found within standard oblique sUAS aerial imagery. All mapped flights are presented in Figures 4.9, 4.11, and 4.13 – 4.17.

### **4.4 sUAS Map Evaluation**

The efficacy of the presented sUAS digital mapping approach was determined by evaluating the spatial accuracy of resulting sUAS maps as well as their comparability to maps produced with conventional remote sensing platforms. After video data from each sUAS flight was orthorectified, the resulting orthomosaic aerial maps were then loaded into ESRI ArcMap and overlaid on a conventional aerial image of the flight area. For the North Farm survey the aerial base map used to overlay the sUAS orthomosaic is a high resolution (3 cm resolution) aerial image collected by the Northern Gulf Institute (NGI) at Mississippi State University. For the Gulf Coast survey the satellite base map used to overlay the sUAS orthomosaic is a 0.5 meter Google Earth satellite image. To evaluate the accuracy of sUAS mosaic map, distance between visible discrete features in the image and matching GPS measured ground control points was determined as a measure of spatial error (Figures 4.9 and 4.11). Then these measures of error for

individual points were used to calculate the RMSE (Figure 4.4), which is a measure of the total spatial error between the maps. If there is strong agreement between the generated sUAS image and ground control points or aerial image, the associated RMSE values will be small.

Conversely, if there is poor agreement between the generated sUAS image and ground control points or the aerial image, the associated RMSE values will be large.

$$\text{RMS error} = \sqrt{\frac{e_1^2 + e_2^2 + e_3^2 + \dots + e_n^2}{n}}$$

Figure 4.4 Root Mean Square Error (RMSE) equation (ESRI, 2022)

RMSE measures errors between the destination control points and the transformed locations of the source control points. This transformation is obtained using least squares, therefore more links can be specified than necessary. However, for an accurate RMSE calculation the specification of links is three to calculate an RMSE result. The more links that are provided, the more accurate the final RMSE value will be (ESRI, 2022).

Given the geometry of sUAS near horizontal video imagery, it seemed likely that imaged features farther away from the sUAS camera would be subject to greater geometric distortion and thus poorer spatial accuracy. To assess this, the relationship between calculated error for each ground control point and distance of that point from the sUAS trackline was plotted to investigate if there was any systematic increase in map error associated with increasing distance from the sUAS (Figures 4.10, 4.12). The resulting plots do not indicate a systematic increase in RMSE with distance of observation away from the sUAS camera.

Similarly, the relationship between error of the GPS ground control points collected in the field and the high-resolution aerial image are visible in Figure 4.13. Analysis of RMSE between both sUAS orthorectified North Farm maps and the high-resolution aerial image are portrayed in Figure 4.14 (Flight 01), and Figure 4.15 (Flight 02). The orthorectified maps of the Mississippi Gulf Coast flights one and two are compared to the satellite image that was used to analyze RMSE, in Figure 4.16 (Flight 01), and Figure 4.17 (Flight 02).

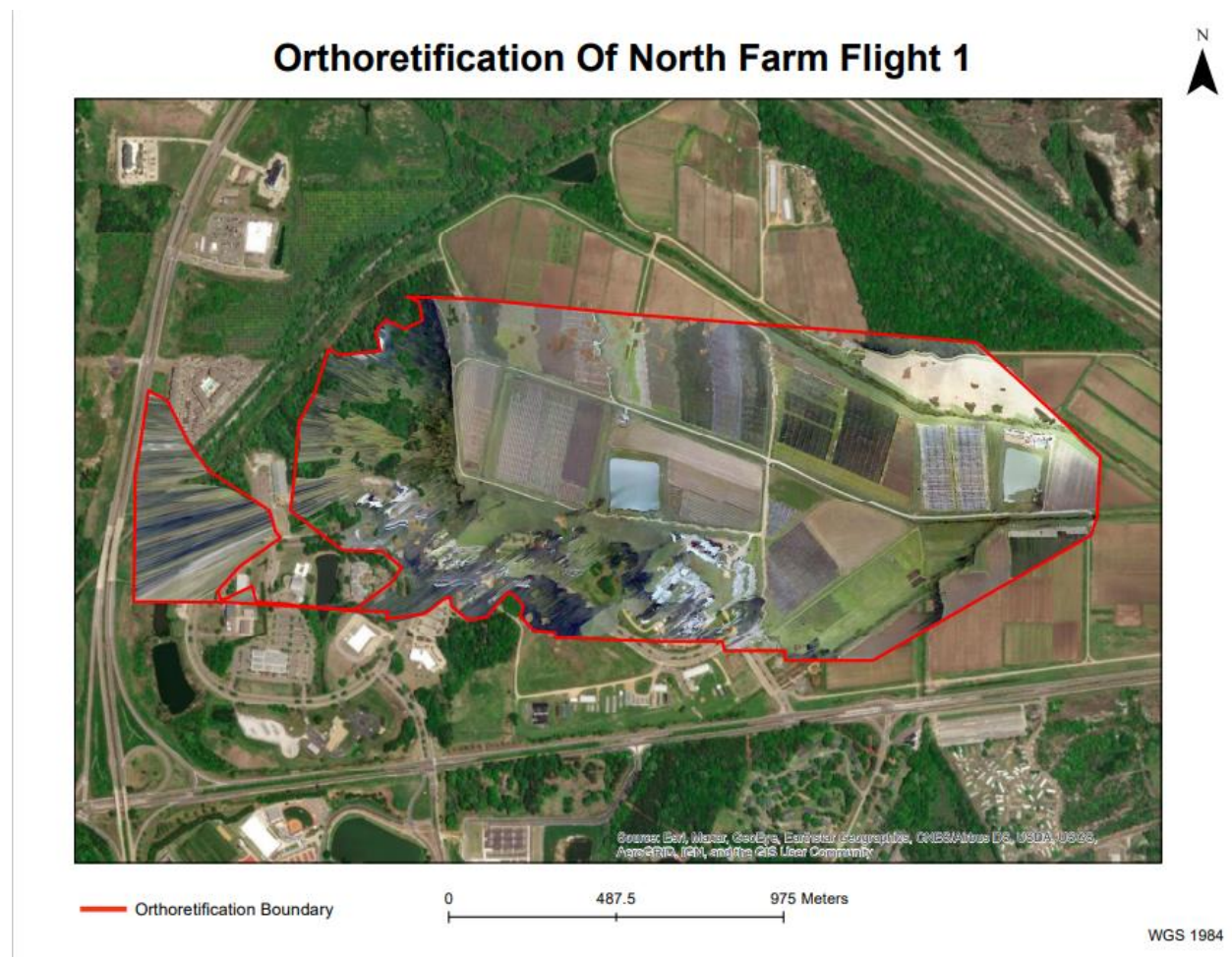


Figure 4.5 Orthorectification of North Farm sUAS Flight 01

The above map shows the orthomosaic aerial image that was orthorectified using the Agisoft Metashape Software. The orthomosaic image is outlined in red and is overlaid onto a satellite image base map of the Mississippi State University North Farm study site.



## Orthorectification Of North Farm Flight 2

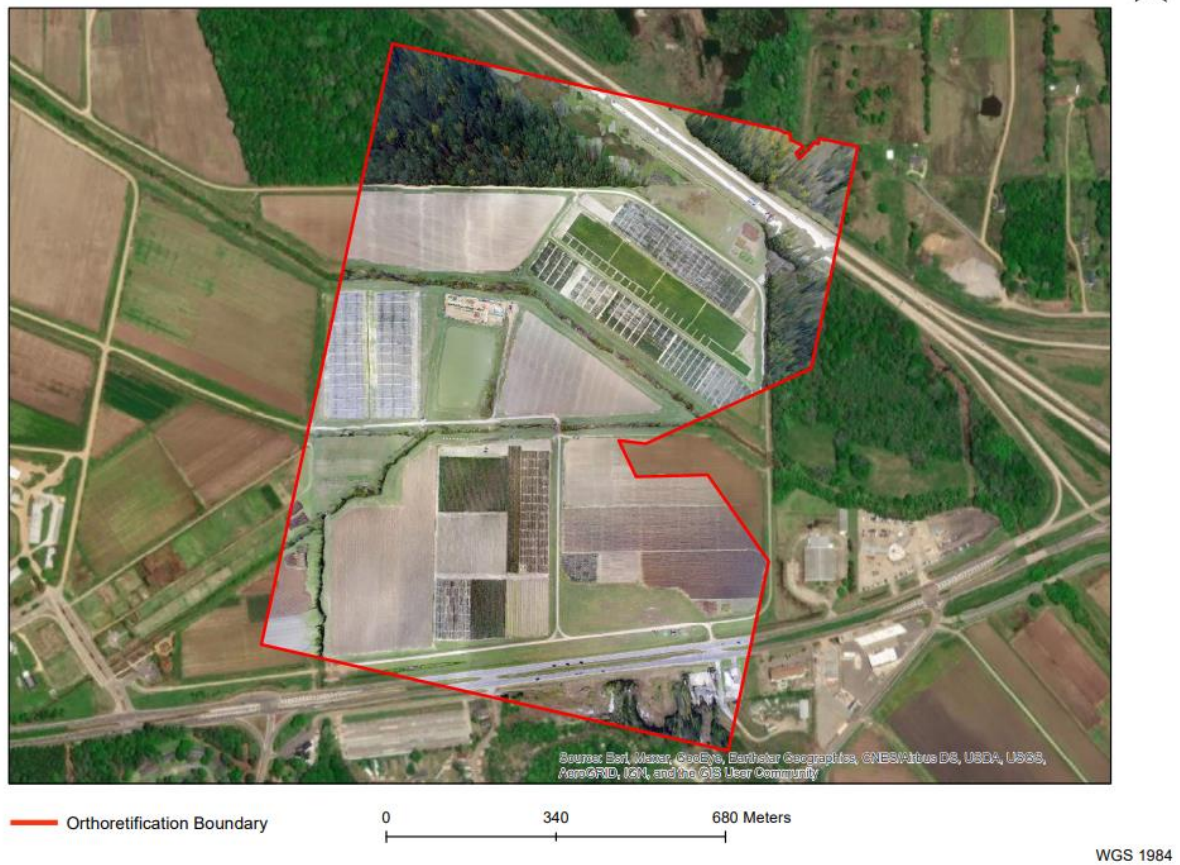


Figure 4.6 Orthorectification of North Farm sUAS Flight 02

The above map shows flight 02 which was flown at the Mississippi State University North Farm study site. Like the previous map showing flight 01, the orthomosaic image produced in Agisoft Metashape is outlined in red and is overlaid onto a satellite image base map of the Mississippi State University North Farm study site.



Figure 4.7 Orthorectification of Mississippi Gulf Coast sUAS flight 01

The above figure accurately displayed the orthomosaic of the Mississippi Gulf Coast sUAS flight 01 that was conducted in Waveland, MS. The orthomosaic of the flight area is outlined in red and was generated using Agisoft Metashape. The orthorectified image is overlaid onto a base map of the Mississippi Gulf Coast study site.



Figure 4.8 Orthorectification of Mississippi Gulf Coast sUAS Flight 02

The above map portrays the orthomosaic of the Mississippi Gulf Coast Flight 02 that was conducted in Waveland, MS. The orthorectified aerial image is overlaid onto a satellite image base map of the Mississippi Gulf Coast study site.

Table 4.1 Comparison of RMSE Values for each sUAS Orthorectified Orthomosaic

<b>RMSE Analysis</b>	<b>Flight Number</b>	<b>Ground Control Points (GCP's)</b>	<b>RMSE Value</b>
Mississippi State North Farm GCP RMSE	Flight 01	39 Points	6.61 meters
Mississippi State North Farm GCP RMSE	Flight 02	44 Points	0.84 meters
Mississippi State North Farm GCP and 3cm High Resolution Aerial Image	N/A	78 points	2.16 meters
Mississippi State North Farm 3 (cm) aerial image RMSE	Flight 01	45 points	5.21 meters
Mississippi State North Farm 3 (cm) aerial image RMSE	Flight 02	33 points	1.28 meters
Mississippi Gulf Coast	Flight 01	15 points	1.68 meters
Mississippi Gulf Coast	Flight 02	15 points	16.24 meters

The above table shows RMSE values that were calculated using aerial triangulation and georeferencing techniques performed in Agisoft Metashape and ArcMap.



## Orthorectification of North Farm Flight 01



### Error (m)

- 0.156169 - 2.590329
- 2.590330 - 4.909273
- 4.909274 - 8.489008
- 8.489009 - 15.035677
- 15.035678 - 19.159680

— Flight Path

0 195 390 780 Meters

WGS 1984  
Jacob Freeman

Figure 4.9 Orthorectification of North Farm Flight 01 Error Map

The map shown in the above figure represents the amount of error measured in meters that was calculated by comparing the ground features identified in the sUAS orthorectified map of North Farm Flight one and ground control points in the North Farm Study Area.

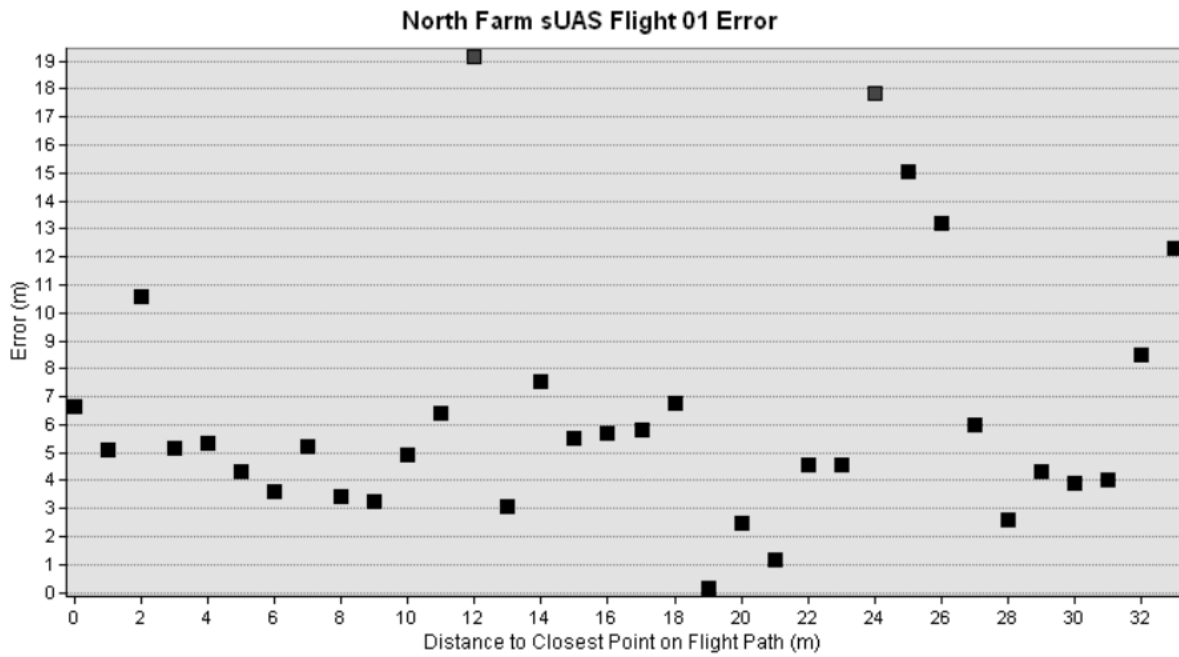


Figure 4.10 North Farm sUAS Flight 01 Error Graph

The above scatter plot shows the distribution of the ground features observed in flight one based on the error measured in meters and their distance to the closest point on the sUAS Flight Path.

## Orthorectification of North Farm Flight 02

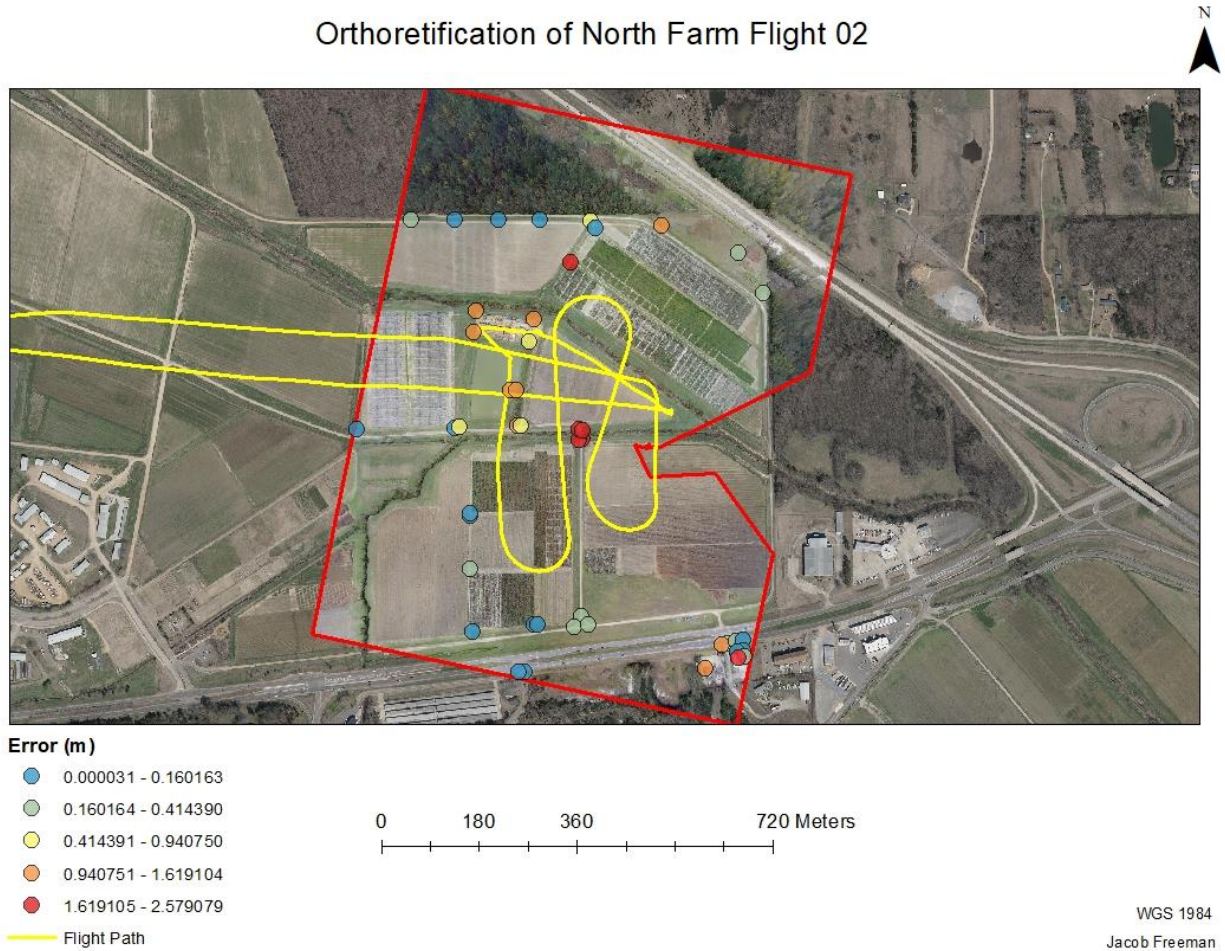


Figure 4.11 Orthorectification of North Farm Flight 02 Error Map

The map shown in the above figure represents the amount of error measured in meters that was calculated by comparing the ground features identified in the sUAS orthorectified map of North Farm Flight two and ground control points in the North Farm Study Area.

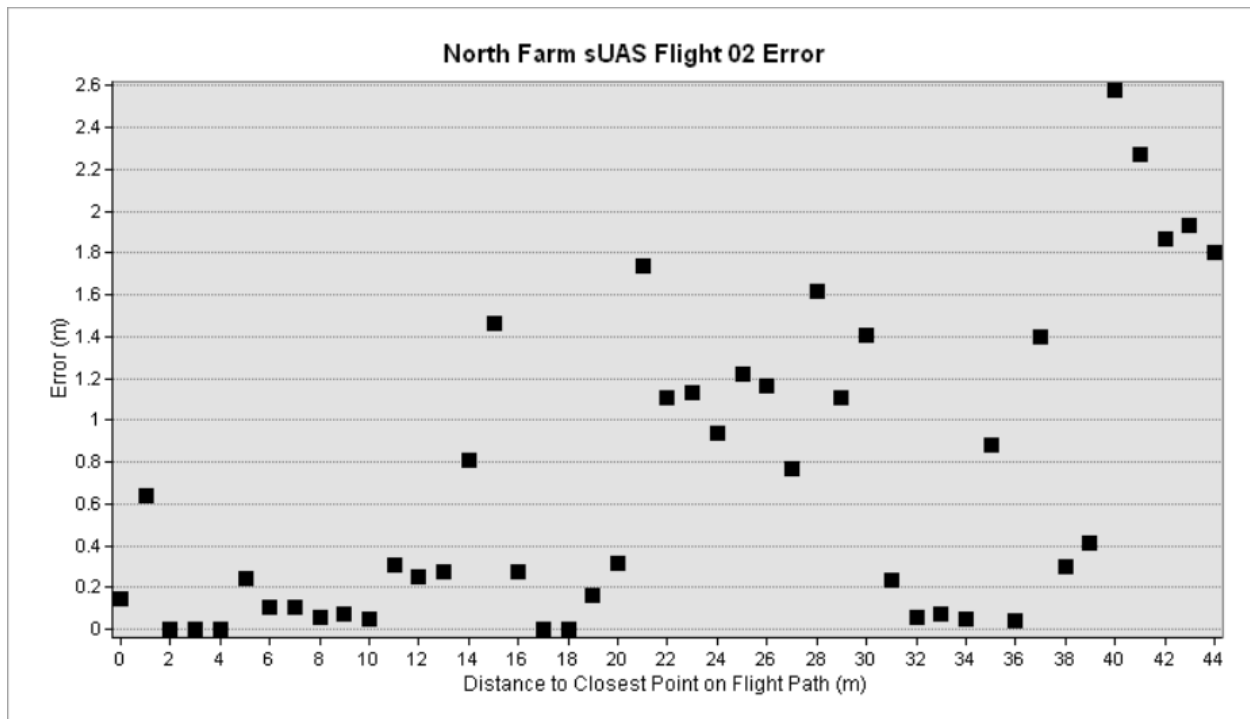


Figure 4.12 North Farm sUAS Flight 02 Error Graph

The above scatter plot shows the distribution of the ground features observed in flight two based on the error measured in meters and their distance to the closest point on the sUAS Flight Path.



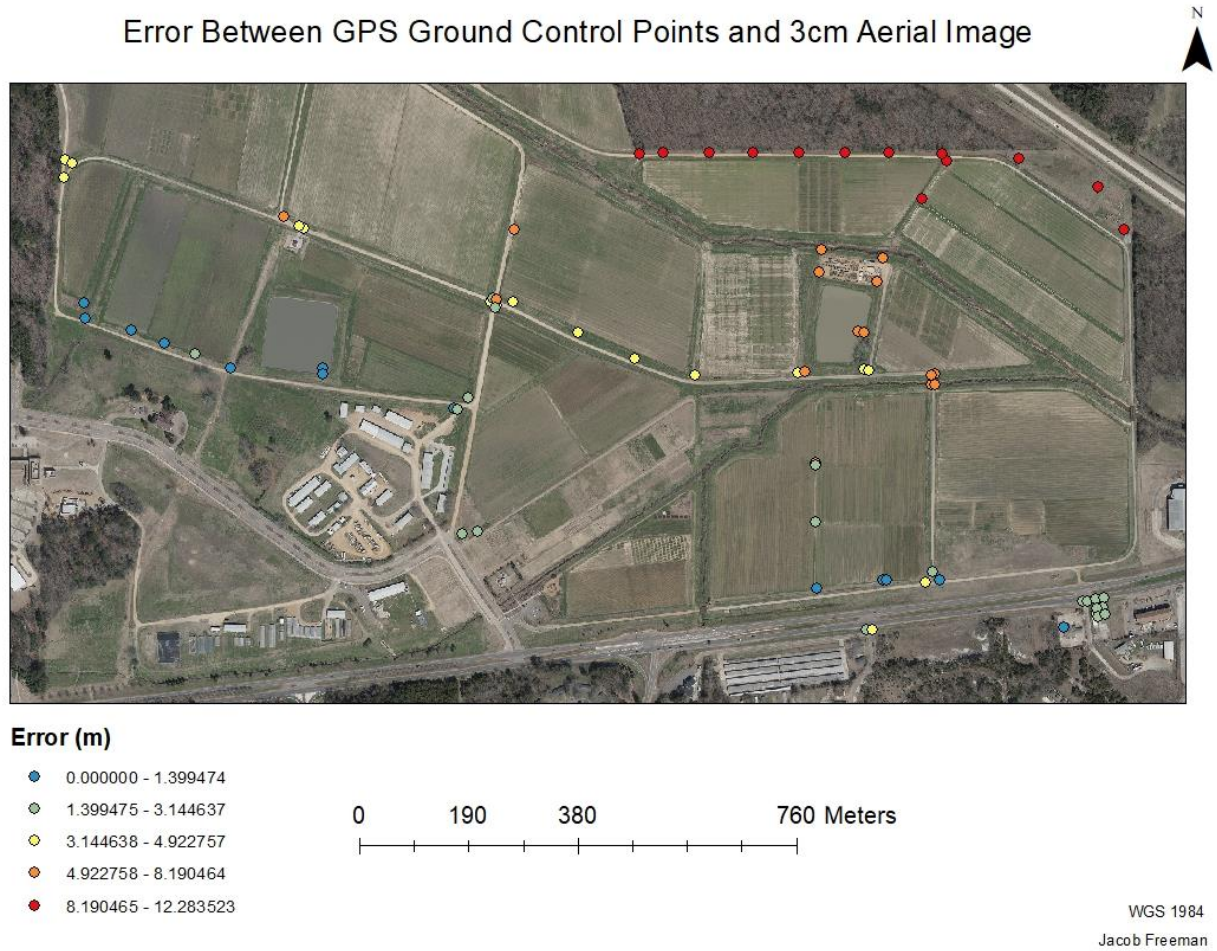


Figure 4.13 Error Between GPS Ground Control Points and 3cm Aerial Image

The map shown in the above figure represents the amount of error measured in meters that was calculated by comparing GCP's collected in the field and the high resolution 3cm aerial image of the North Farm Study Area.

## Error Between North Farm sUAS Flight 01 and 3cm Aerial Image

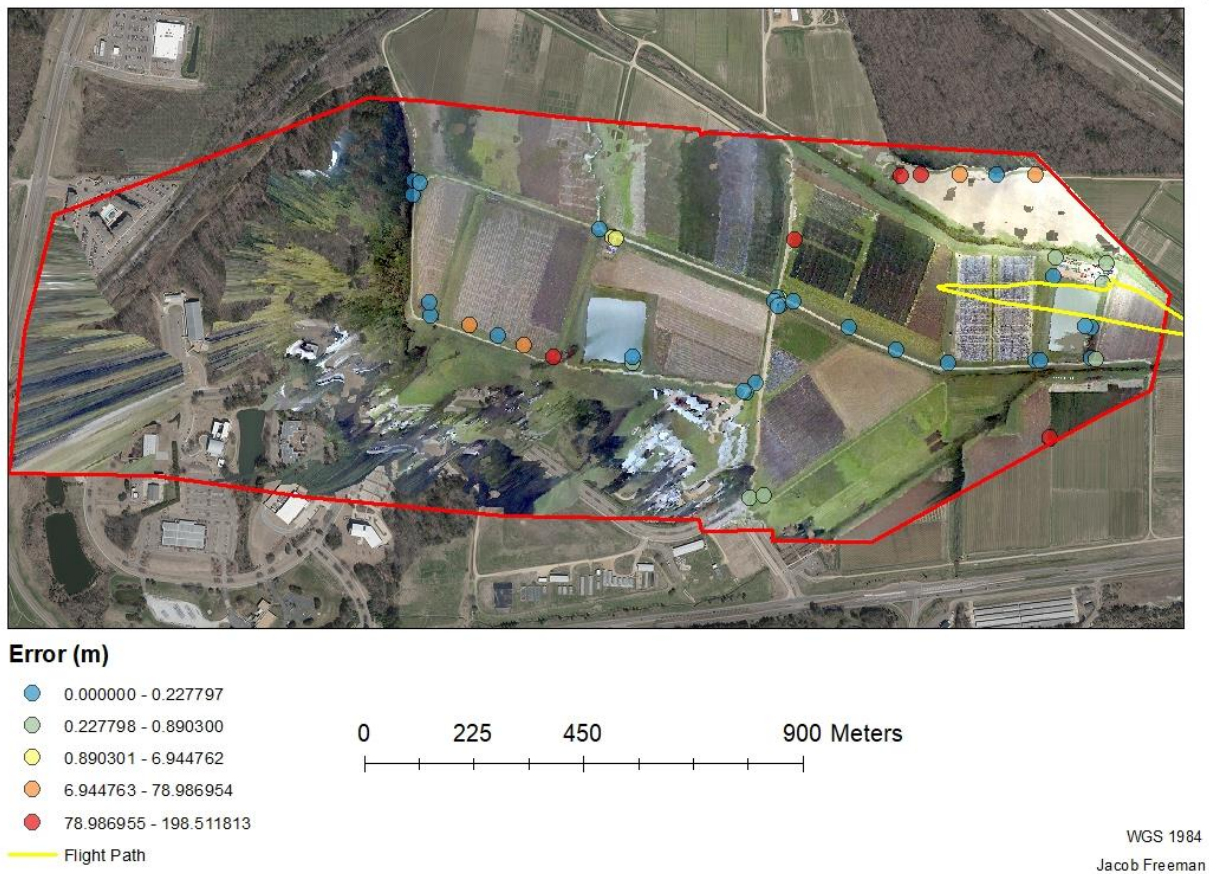


Figure 4.14 Error Between North Farm sUAS Flight 01 and 3cm Aerial Image

The map shown in the above figure represents the amount of error measured in meters that was calculated by comparing the sUAS orthorectified map image of flight 01 and the high resolution aerial image.



## Error Between North Farm sUAS Flight 02 and 3cm Aerial Image

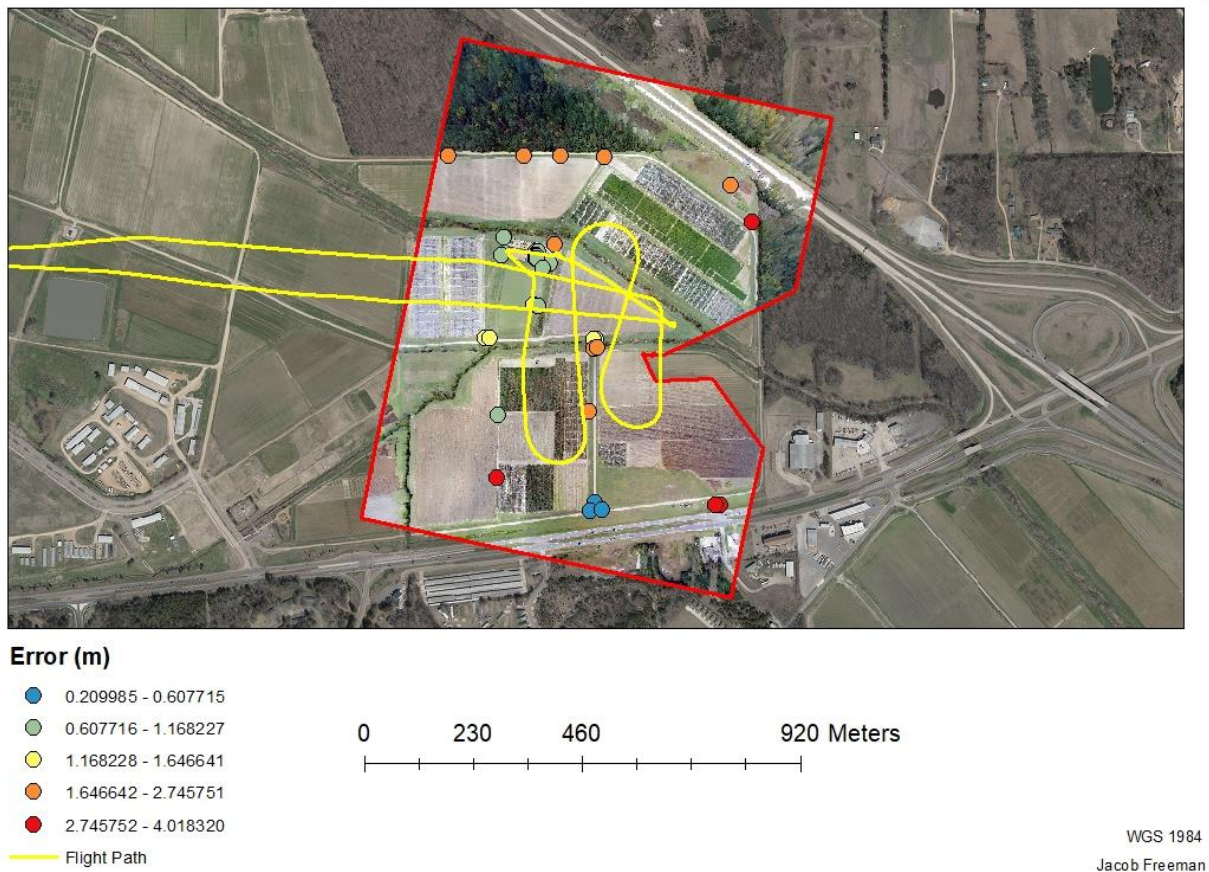


Figure 4.15 Error Between sUAS North Farm Flight 02 and 3cm Aerial Image

The map shown in the above figure represents the amount of error measured in meters that was calculated by comparing the sUAS orthorectified map image of flight 02 and the high resolution aerial image.

## Error Between sUAS Coast Flight 01 and Satellite Image

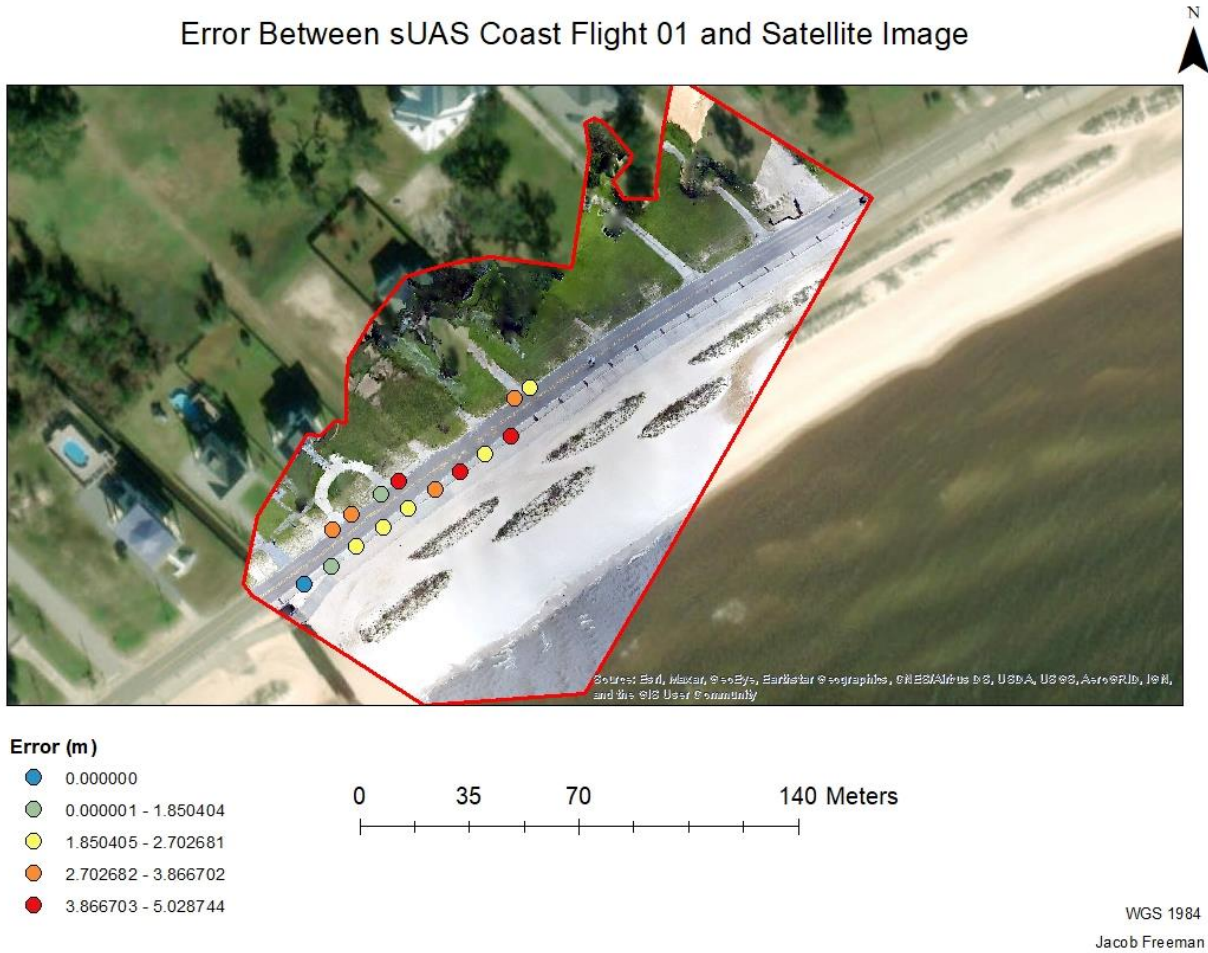


Figure 4.16 Error Between sUAS Coast Flight 01 and Satellite Image

The map shown in the above figure represents the amount of error between sUAS Mississippi Gulf Coast flight 01 orthorectified map and the satellite image. The flight path for this sUAS flight is visible in Figure 3.8.

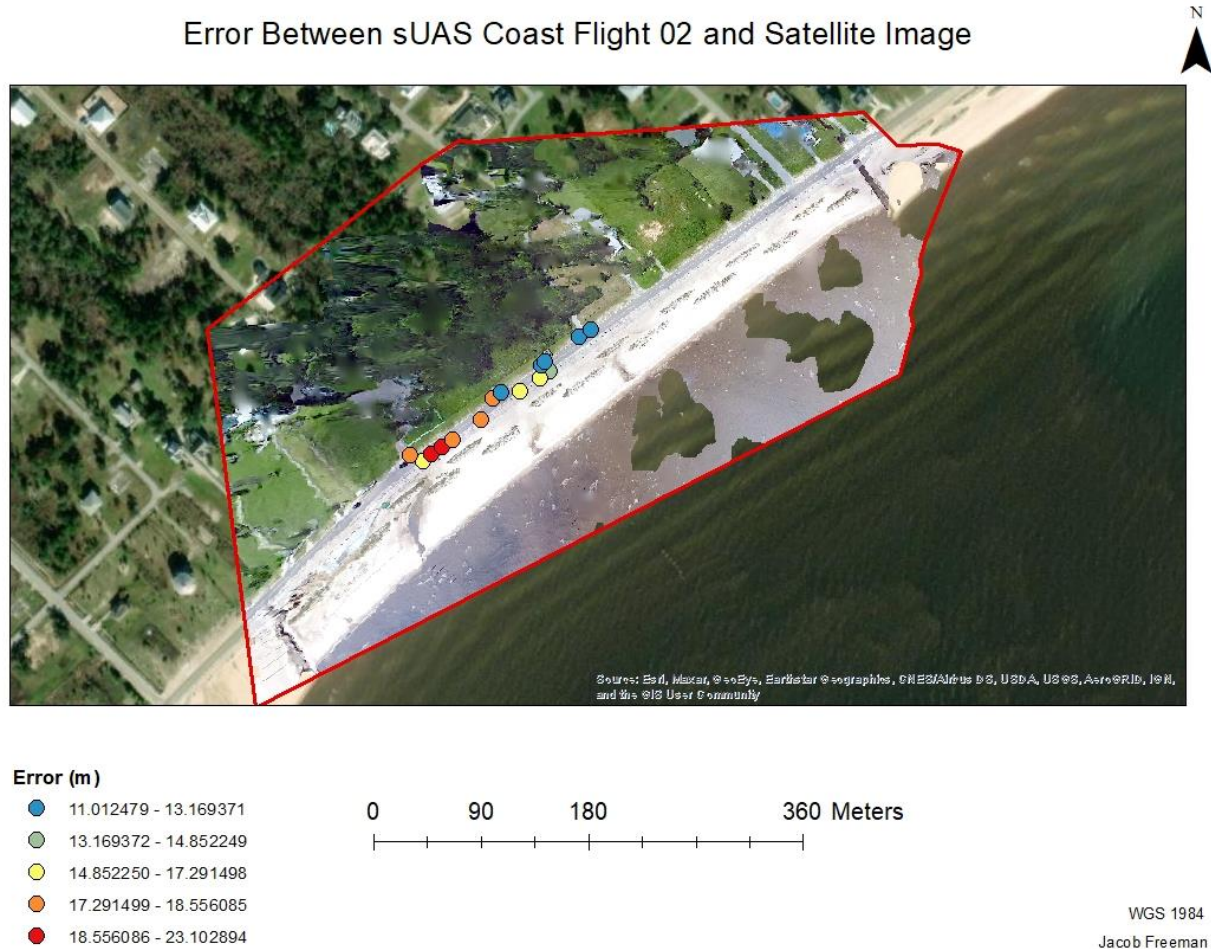


Figure 4.17 Error Between sUAS Coast Flight 02 and Satellite Image

The map shown in the above figure represents the amount of error between sUAS Mississippi Gulf Coast flight 02 orthorectified map and the satellite image. The flight path for this sUAS flight is visible in Figure 3.8.

## 4.5 ROV Orthorectification

Once both the Mississippi State University North farm and Mississippi Gulf Coast sUAS datasets were accurately orthorectified, the same Agisoft Metashape methodology was applied to the ROV Deep Discoverer video data to investigate the possibility of potentially orthorectified ROV oblique imagery and videography that is collected along the seafloor. The ROV video



frame extraction and timestamp script that was previously introduced in the methods section was applied to two ROV dive video datasets downloaded from *Seatube V2* (i.e., EX 1903 L2 Dive 06, and EX 1903 L2 Dive 8).

The orthorectification of the extracted timestamped video frames from the ROV video data was executed in Agisoft Metashape where a final orthomosaic image of the seafloor was produced. However, due to the lack of an existing seafloor map that contains visible seafloor geologic features, it is not possible to use RMSE techniques to evaluate the accuracy of resultant maps. Specifically, low resolution and zoom distortion limitations were prevalent within the ROV video datasets. Visual comparisons for seafloor substrate of the ROV orthomosaic image are obtainable through identifying and comparing similar substrate benthic features found throughout both the Final ROV substrate Maps found in Appendix C and the Acoustic Backscatter Comparison Maps found in Appendix D.

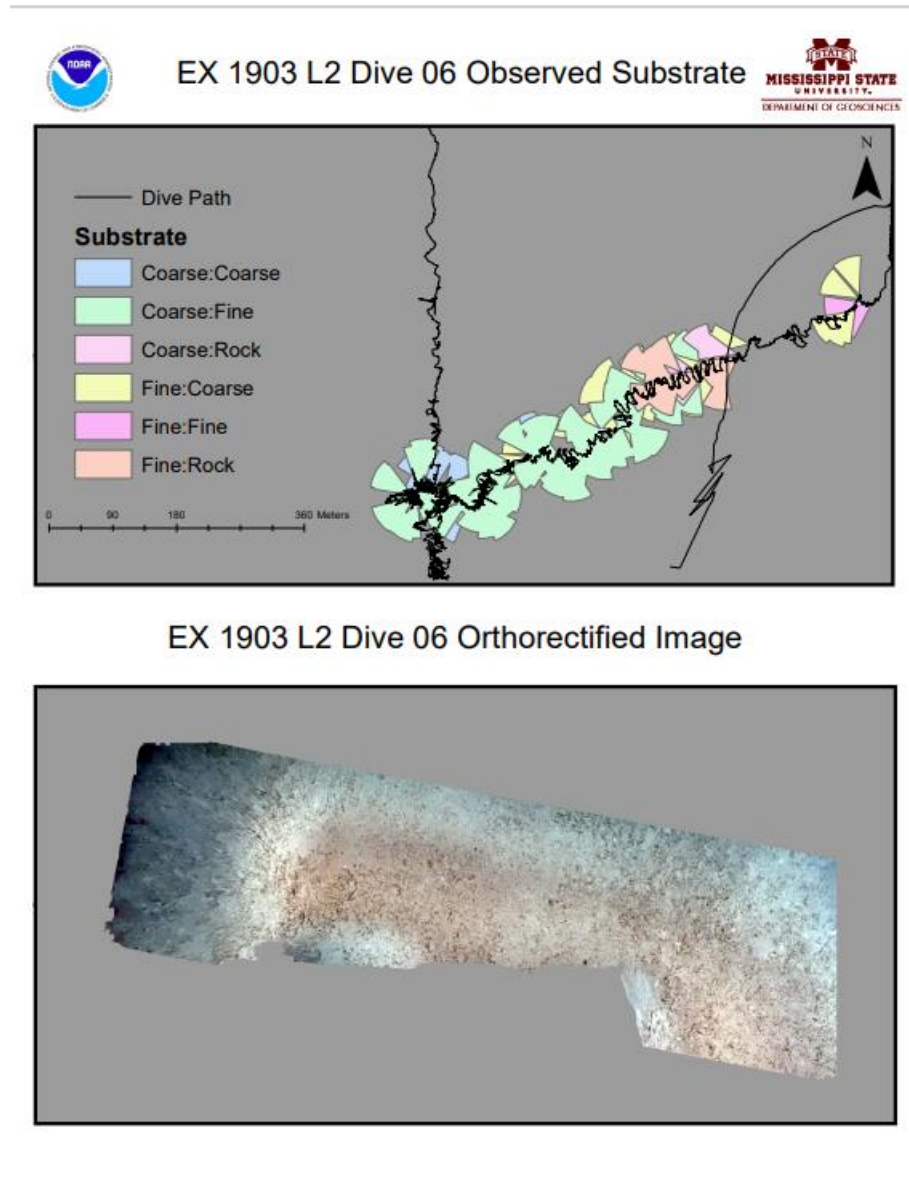
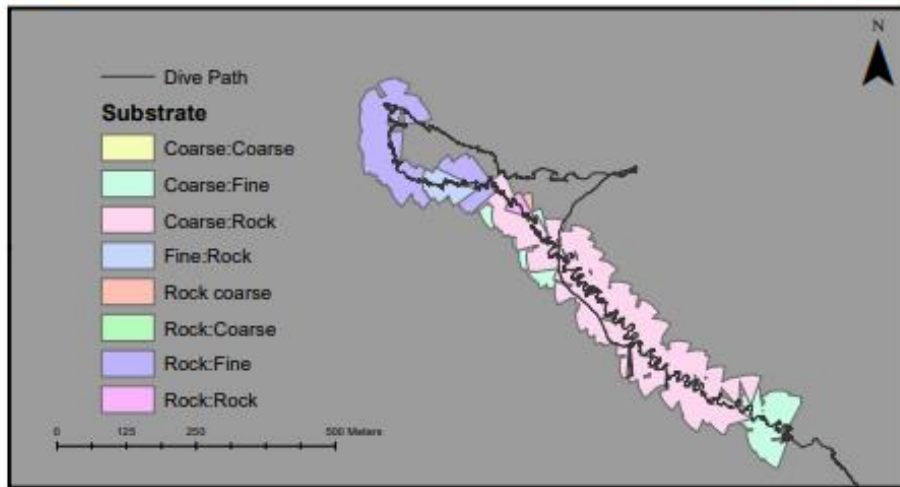


Figure 4.18 Orthorectified ROV Orthomosaic Comparison to Observed Substrate EX 1903 L2 Dive 06

The above map depicts both observed substrates throughout EX 1903 L2 Dive 06 and the orthomosaic image of EX 1903 L2 Dive 06 that was produced in Agisoft Metashape.



## EX 1903 L2 Dive 08 Observed Substrate



## EX 1903 L2 Dive 08 Orthorectified Image

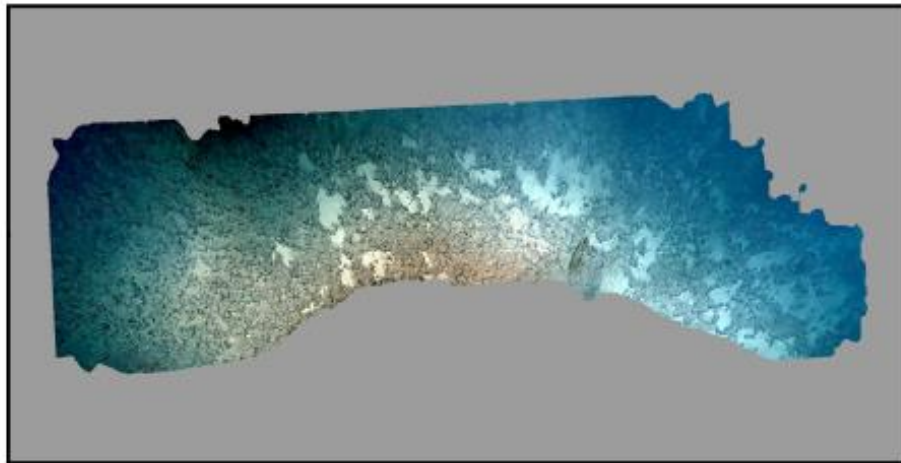


Figure 4.19 Orthorectified ROV Orthomosaic Comparison to Observed Substrate EX 1903 L2 Dive 08

As shown in the above figure shows both the Observed Substrate Map for EX 1903 L2 Dive 08 and the ROV orthomosaic image for EX 1903 L2 Dive 08 that was produced in Agisoft Metashape.



## CHAPTER V

### DISCUSSION

#### 5.1 ROV Automated Mapping Implementation

Implementation of the newly developed open source ROV video data processing tools and GIS workflows presented in this thesis resulted in the efficient generation of digital maps that represent the spatial distribution of seafloor substrate classes observed in oblique video imagery collected on ROV dives in a manner suitable for quantitative geospatial analysis (Appendix C). The efficacy of the presented digital mapping approach was evaluated by comparing resultant substrate maps to coincident seafloor backscatter data, which is the closest dataset to a “ground truth” available for the study area and a widely accepted conventional approach for mapping seafloor substrate type (Lurton, 2010). Thus, it is the most appropriate independent data for comparison to the viewshed substrate maps generated from integrated ROV video and annotation data. Figure 4.1 is an example of the substrate map resulting from this thesis overlain on backscatter data. Notably the mapped substrate classes Fine:Fine are generally spatially coincident with areas of low backscatter (darker tone) intensity, which is consistent with mud substrate, and the mapped substrate classes Fine:Rock and Rock:Fine are generally spatially coincident with areas of high backscatter intensity (lighter tone), which is consistent with rock substrate (Lurton, 2010). Visual interpretation of these results suggests that the substrate maps produced through the methods presented in this thesis are consistent with coincident independent measurements of substrate made with multibeam sonar. However, more detailed quantitative

assessment was undertaken to evaluate this conclusion. Figure 4.3 and Appendix D contain boxplots that indicates the frequency distribution of backscatter intensity values spatially coincident with the labeled CMECS polygons. Figure 4.3 demonstrates an expected relationship based on sonar theory (Lurton 201) in which the median backscatter intensity value is lowest for the softest and smoothest substrate (Fine:Fine), progressively increases for substrate classes that are progressively harder and rougher (Fine:Rock, Rock: Fine), and is greatest for the hardest and roughest substrate class (Rock:Rock). Of the 10 ROV dives with assigned CMECS substrate classes that were compared to acoustic backscatter intensity, six of exhibited agreement between substrate class and relative backscatter intensity (e.g., EX 1803 Dive 15 (Figure D.12), EX 1806 Dive 04 (Figure D.13), EX 1806 Dive 08 (Figure D.14), EX 1806 Dive 13 (Figure D.51), EX 1903 L2 Dive 09 (Figure D.18).. For the remaining four ROV dive datasets (e.g., EX 1923 L2 Dive 05 (Figure D.16), EX 1923 L2 Dive 06 (Figure D.17), EX 1803 Dive 08 (Figure D.11), and EX 1903 L2 Dive 19 (Figure D.19), the substrate comparison of the ROV substrate viewshed map and the acoustic backscatter data was not as consistent. This lack of agreement is likely due to survey environmental conditions that violated assumptions inherent in the use of acoustic backscatter as ground truth for substrate maps, including assumptions of a flat seafloor and uniform viewshed size (Malik, 2019).

One of the most important ways the presented results enable ocean research and management is by allowing scientists, managers, and other users to rapidly understand the information contained in ROV video data. For individuals not directly involved in the scientific study and the collection of the video data, the amount of time and effort that is required to review and understand what tens to hundreds of hours of video is analyzing to determine its potential value to their study and application is prohibitive. NOAA National Centers for Environmental

Information (NCEI) host the largest database of ROV video data in the world. The useability of those data could be substantially enhanced if interested scientists and others could review maps, such as the ones developed for this thesis, that show the presence or absence of biological, chemical, physical, and geological features of interest in the video data. Additionally, beyond simply understanding the presence or absence of features of interest, scientists can use the maps of ROV video data to understand the spatial association of those features across each dive site. This can yield immediate insight into processes and relationships that would not necessarily be evident from watching the dive videos alone.

The automated ROV video data processing tools and GIS workflows presented in this thesis are not exclusively limited to the mapping of seafloor substrate. *Seatube* annotation files contain expert observations of biological organisms, and other geological properties of the seafloor observed on ROV dives. Any annotation class of interest can be plotted on maps with the same viewshed framework presented in this thesis for substrate annotations. Additionally, this approach could be expanded to map any recorded feature of interest on an ROV dive.

## **5.2 Suggested ROV Viewshed Mapping Improvements**

Based upon the final map products presented in this thesis, there are several improvements that could strengthen the viewshed approach to automated mapping of ROV video observations. First, a designated Python script designed to extract both the ROV annotation dataset from the *Seatube* web interface and the 1Hz ROV dataset from NOAA OER digital atlas web interface in an automated fashion would increase the utility and efficiency of the ROV automated mapping script. Directly accessing the required ROV navigation and video annotation data from their hosting websites would eliminate the current requirement to manually download each dataset from online sources, increasing the efficiency of the ROV automated mapping

approach. However, directly accessing the data from the hosting website comes with a variety of disadvantages too. For example, if a Python script was constructed to extract both the 1Hz ROV dataset and the ROV annotation dataset from the original website sources (i.e., *Seatube* and OER Digital Atlas) it would only be functional as long as the source code for the respective websites were not updated. Any future updates to those websites could render the Python data extraction code nonfunctional and therefore the code base would have to be tested and updated often to ensure continuous functionality and forward compatibility.

As noted, viewsheds are an approximation of the area of the seafloor encompassed in a single ROV video frame image. The dimensions and angle of the viewsheds used herein were assigned constant values that are representative of the visible range of the ROV camera over flat seafloor terrain when the camera is not zoomed or tilted. In the case where the ROV camera zoom is increased, the viewshed will be smaller. Likewise, when the camera is tilted up the viewshed range will be longer and when it is tilted down the viewshed range will be shorter. Additionally, in the case of non-flat seafloor terrain, the shape of the viewshed will vary based on the morphology of the seafloor in the viewshed. In the future, the viewshed approach could be improved by dynamically altering the shape, size, and range of each viewshed based on real-time values for camera tilt and zoom as well as seafloor morphology. Because camera tilt and zoom values are transmitted digital signals, recording them as part of the ROV operation data should be an easy task. Integrating seafloor morphology will be more challenging; however, the *Okeanos Explorer* maps the bathymetry of each dive site with a multibeam sonar, as standard practice, which yields a three-dimensional digital elevation model of the seafloor. Accordingly, calculating the shape of the viewshed based upon seafloor morphology is possible but may

require significant data sourcing and complex geometric projection, which may prove to be prohibitive in terms of processing time and computational workload.

### **5.3 ROV Orthorectification of Oblique Imagery**

This thesis represents the first time ROV video data has been orthorectified to produce an orthomosaic image of seafloor environments. This approach was tested on sample dives (EX 1903 L2 Dive 06, and EX 1903 L2 Dive 08) to evaluate the potential for the production of an accurate orthorectified mosaic image of seafloor habitat. For both dives it was presumed that certain limitations (i.e., zoom, video resolution, and camera orientation) found within the ROV video data would create a high degree of geometric distortion within the final seafloor orthomosaic, thus limiting its usefulness. Much of this anticipated distortion was corrected through manual adjustments to be made for the sample orthomosaic, as presented in the methods section. The presence of three-dimensional seafloor objects and organisms (i.e., coral, sea sponges, squid, and fish) when matched with the highly variable zoom of the ROV camera lenses produced highly distorted areas within the final orthomosaic of the seafloor. This distortion relates to ROV camera zoom mentioned previously and the way in which it reduces the size of the video frame viewshed. As the ROV camera records video throughout the dive, a variety of seafloor features are zoomed into view by the ROVs forward camera, if the focal length of the ROVs camera was specified each time it zooms a seafloor feature into view, there would be less distortion evident in the final seafloor mosaic. Alternatively, video frames collected during periods when the camera was zoomed in could be removed from the image dataset used to create the orthostatic image. Despite distortion associated with zoom, the generated seafloor orthomosaics for EX 1903 L2 Dive 06 and EX 1903 L2 Dive 08 presented in Figures 4.18 and 4.19 demonstrate the potential utility of this approach for benthic research. Indeed, the resulting

orthomosaic images could serve many of the same purposes the substrate viewshed maps serve, such as geospatially representing the location and spatial variability of seafloor features observed in ROV dives. Further, the application of computer vision techniques to orthomosaic images could identify imaged features like biological organisms without the need for expert annotation. Such efforts are beyond the scope of this thesis, but the results of this thesis demonstrate the initial feasibility and potential of such approaches.

Although Agisoft Metashape generally applies photogrammetric processing techniques to digital images collected with a nadir-viewing (downward) camera acquired above the surface, a variety of different ROV navigational and camera parameters (i.e., high-resolution video data, fixed zoom, and camera orientation) could potentially be adjusted to obtain a correct geometrically oriented orthomosaic of the seafloor. For example, if a double grid transect was conducted by the Deep Discoverer to survey a dive site video conducted with both a forward-facing camera (oblique) or a nadir-viewing camera (downward) the resulting orthomosaics would be far more accurate and clear.



Figure 5.1 ROV Image of seafloor EX 1903 L2 Dive 06

The above image shows an image of the seafloor captured by the forward-facing camera of the Deep Discoverer.



Figure 5.2 ROV orthorectified orthomosaic image of the seafloor from EX 1903 L2 Dive 06

The above figure is an orthomosaic image section of EX 1903 L2 Dive 06. Some minor distortion within the image is visible, this is due to the forward-facing camera on the Deep Discoverer zooming in on seafloor features found within the video.

The results of this thesis demonstrate that seafloor features observed within the ROV video (Figure 5.1) are visually identifiable in an orthomosaic of the full dive (Figure 5.2) such as the presence of soft sediment. However, some seafloor features such as *Lophelia* coral identifiable in Figure 5.1 are not as clear within the seafloor orthomosaic in Figure 5.2. Both images can be compared to two different ROV seafloor map products. The acoustic backscatter intensity map For EX 1903 L2 Dive 06 found in Appendix D and the ROV substrate viewshed map found in Appendix C can be examined for further comparison of the observed benthic features.





Figure 5.3 ROV image of benthic features throughout EX 1903 L2 Dive 08

The above image emphasizes the observed seafloor rock features that were examined throughout EX 1903 L2 Dive 8.

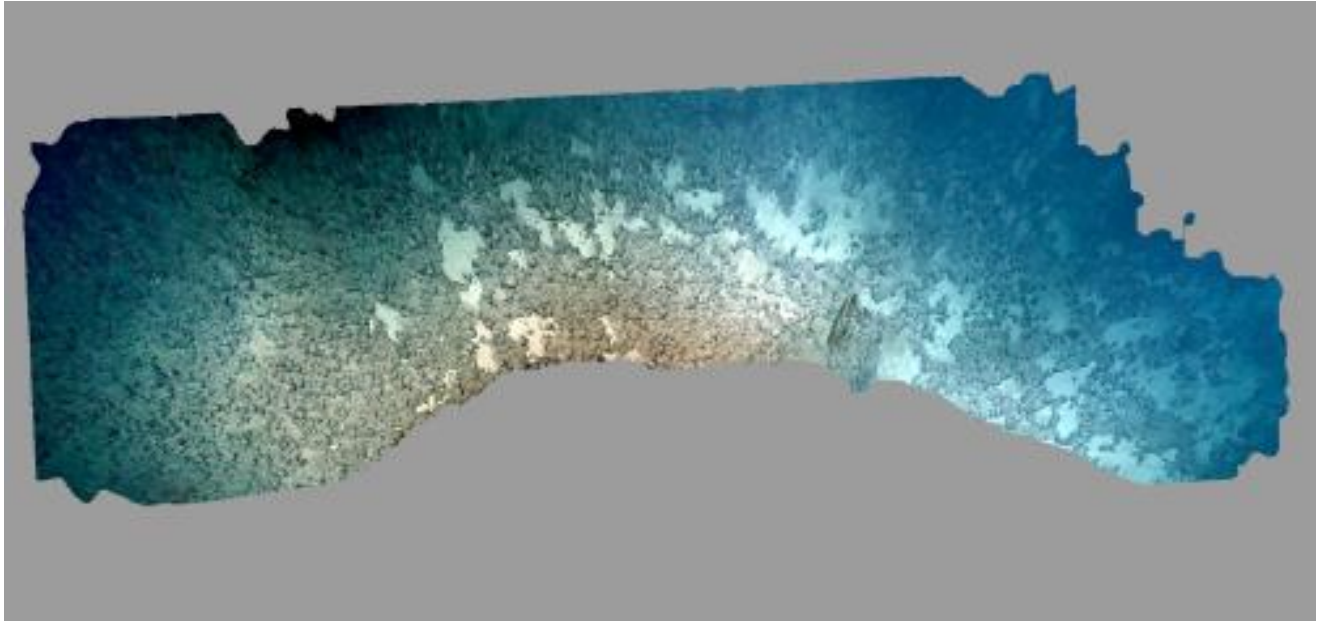


Figure 5.4 Orthomosaic image section of EX 1903 L2 Dive 08.

For example, in Figure 5.3 a large quantity of coarse rock features is visible on the seafloor. In the above orthorectified orthomosaic image an abundance of coarse rock features is visible.

Figures 5.3 and 5.4 again demonstrate that the seafloor features observed within the ROV video (Figure 5.3) are visually identifiable in an orthomosaic of the full dive (Figure 5.4), in this case, coarse rock. Both images are also consistent with the ROV substrate map for EX 1903 L2 Dive 08 (Figure C.29) found in Appendix C, which indicates an abundance of the CMECS class Coarse:Rock. As noted previously, orthomosaics produced from ROV video data can observe both biotic and abiotic features found within benthic habitats. In the figures below the ability to observe marine organisms with orthomosaics that are found within benthic habitats is demonstrated.



Figure 5.5 ROV image of marine organisms during EX 1903 L2 Dive 16

The image above shows a benthic environment that is inhabited by a variety of marine organisms (e.g., crabs).

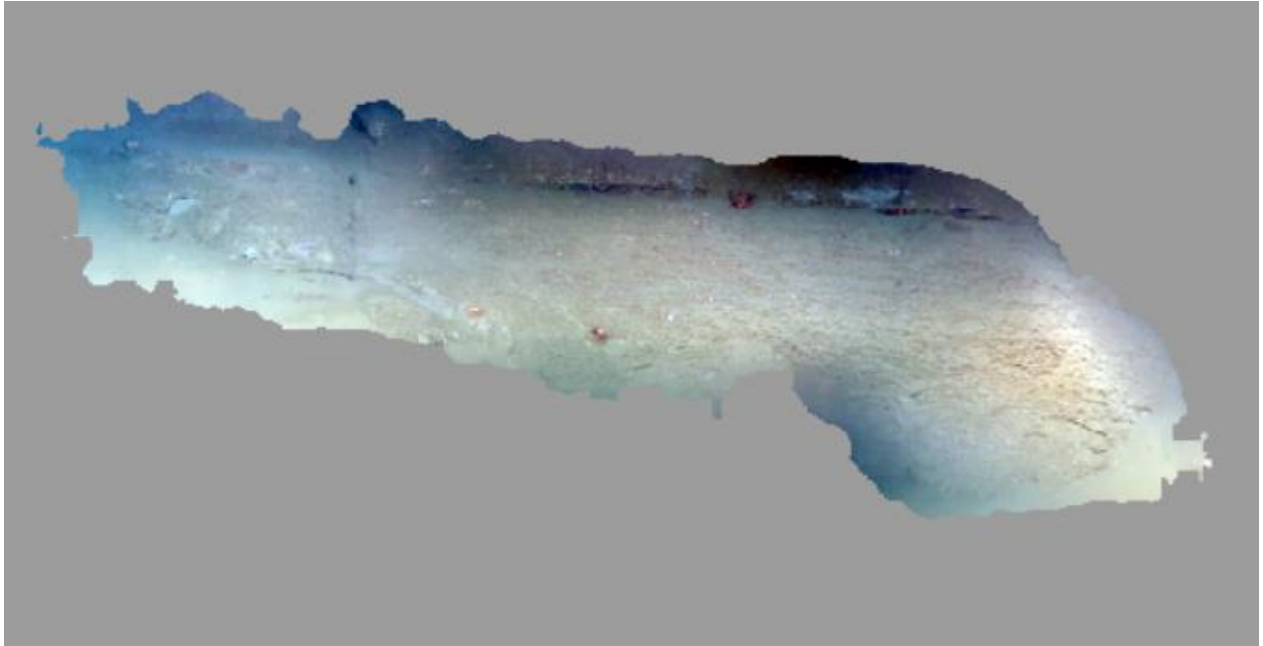


Figure 5.6 Orthomosaic image section of EX 1903 L2 Dive 16

Visible biotic and abiotic marine features (e.g., crabs) are visible throughout the above-orthorectified image of EX 1903 L2 Dive 16.

Similar to the previous images found in the above figures, both images in Figure 5.5 and 5.6 can be visually compared to the ROV substrate maps found in Appendix C and the Acoustic backscatter comparison maps found in Appendix D. When compared, both orthomosaic images represent the distribution of seafloor substrate by correctly classifying the substrate type and size that was observed during the ROV dive.

#### 5.4 sUAS Orthorectification Implementation

A desirable RMSE value for aerial photography and videography collected with a sUAS is a final RMSE value of up to 1 pixel; however, larger distortions and thus higher RSME values are produced when more conventional “low-cost” cameras are used for aerial imagery collection. Therefore, an RMSE value of 1.5 to 2 pixels is generally viewed as acceptable for the aerial triangulation of sUAS nadiral collected imagery (Calvario et al, 2017).

Although substantial previous research focused on the orthorectification of sUAS imagery has been conducted, a small number of these studies consider near-horizontal oblique imagery collected by sUAS and the potential of using it to create accurate maps. The objective of the sUAS orthorectification portion of this thesis was to use a suite of tools and GIS workflows designed to digitally map oblique video imagery (Orthomosaic) collected on sUAS flights in order to enable a quantitative geospatial analysis of those data. The efficacy of this approach was evaluated by assessing quantitative spatial agreement between identifiable features (i.e., pier, culvert, light pole, house) in orthorectified images and coincident ground control points or surface satellite imagery. Four sUAS flights were selected for orthorectification and georeferencing and therefore were the only sUAS images available for RMSE calculation.

Creating orthomosaic images from oblique sUAS imagery offers some specific advantages over traditional surveys conducted with cameras in a nadiral (downward) orientation. For example, imagery collected with a sUAS flown at a low altitude with an oblique viewing camera orientation can effectively capture a variety of ground (surface) features that are not visible through the orthorectification of nadiral imagery (Hodgson and Morgon, 2020). This allows researchers to survey and monitor coastal and marine environments with an accuracy not previously possible (Hodgson and Morgon, 2020). With the potential to orthorectify a variety of camera orientation angles, the capability to create a more in-depth and accurate visual orthomosaic of the study area is achievable. The resulting orthomosaic images of the Mississippi State North Farm Study area seen in Figure 4.9, Figure 4.11, Figure 4.13, Figure 4.14, and Figure 4.15 depict the potential that oblique sUAS image orthorectification to effectively access and survey an agricultural study site. Figure 4.16, and 4.17 of the results section represent the final orthomosaic images and spatial analysis of the Mississippi Gulf Coast study site. The results

indicate that recreational and conventional data collection methods allow geospatial analyses to be conducted in a marine and coastal environment with the use of oblique imagery.

## **5.5 Suggested sUAS Orthorectification Mapping Improvements**

A variety of potential improvements to the presented methods for orthorectified sUAS imagery were revealed through the research presented here. Foremost, the sUAS flights used for this thesis were all conducted in “free flight” mode in which the pilot did not follow a specific flight path pattern such as a grid. In many ways, this is advantageous because it demonstrates that the presented approach is valid for existing sUAS data from flights in which a flight path pattern was not used. This is particularly true because amateur operators are likely to fly in such a mode and this thesis makes clear the resulting data can still be plotted in an orthomosaic using the approaches presented herein. However, published results indicate that the quality of resulting orthomosaics can be optimized by flying the sUAS survey in a grid flight pattern (Taddia et al, 2020). This method is demonstrated in a variety of published studies and would potentially yield more accurate results in future studies (Nesbit and Hugenholtz, 2018; Taddia et al, 2020). In Figure 4.9, the first sUAS Mississippi State North Farm flight is shown. This flight was flown in a “Down and Back” flight pattern. The final georeferenced orthomosaic of this flight had the highest RMSE value of 6.61 when the field collected GCP’s were used to quantify the error. As seen in Figure 4.11, flight 02 was flown in an “s” shaped pattern with crossing lines oriented at 90° approximating a grid, resulting in a final orthomosaic image with the lowest of RMSE value (0.84 meters) of all sUAS flights. The three-centimeter aerial imagery of North Farm at Mississippi State and the collected GCP’s were used to conduct the RMSE portrayed in Figure 4.13. The results indicated that the 3-cm aerial image created using conventional mapping techniques yielded an RMSE value of 2.16 meters. This result indicates that the sUAS

orthomosaic map collected on North Farm flight 1 has a lower accuracy (RMSE = 6.61m) than a map generated with data collected with a conventional aerial remote sensing platform (RMSE = 2.16m). Conversely, the sUAS orthomosaic map collected on North Farm flight 2 has a higher accuracy (RMSE = 0.84m) than a map generated with data collected with a conventional aerial remote sensing platform (RMSE = 2.16m). North Farm sUAS flight 01 yielded the highest RMSE value at 5.21 meters when referenced to the 3cm aerial image. (Figure 4.14), whereas flight 02 (Figure 4.15) had an RMSE value of 1.28 when compared to the 3cm aerial image of Mississippi State North Farm. Moreover, the results indicate that a double or single-grid flight pattern may be a more desirable method rather than conducting a straight uniform flight pattern when using oblique imagery for mapping, which is consistent with published guidelines (Taddia et al, 2020). Figure 4.11 and Figure 4.15 both represent flight 02 of the Mississippi State North Farm study site, yielding the lowest RMSE values calculated throughout this research (0.84 and 1.28 meters). Figures 4.16 and 4.17 of the results section are RMSE maps created by comparing ground features between both sUAS coast flights, and the satellite image used to generate the amount of error between them. Flight 01 (Figure 4.16) yielded a RMSE of 1.68 meters. Flight 02 (Figure 4.17) had a RMSE of 16.24 meters. Both sUAS coast flights were conducted in a “free roam” flight pattern and did not follow a grid flight path.

## **5.6 sUAS Applications of Orthorectified Oblique Imagery**

### **5.6.1 Coastal Mapping**

Geospatially representing sUAS orthomosaic images collected in a marine and coastal environment that are recorded using aerial videography techniques is possible throughout the workflows and methods found within this thesis. The ability to accurately visualize environmental parameters that are recorded during an sUAS flight over a coastal environment

and analyze them in a geospatial context allows researchers to gain a deeper understanding of what is being visually observed within the sUAS video dataset, and how these recorded environmental observations can be applied to their specific scientific field or research study. The ability to conduct sUAS flights with a DJI Phantom 4 multispectral camera system and an infrared camera will allow land/water delineation analysis to be conducted in future studies. Land/ water delineation will allow coastal researchers to accurately monitor the rate of coastal erosion, as well as the ability to conduct analyses pertaining to coastal vegetation, hydrology, and geology. Digital Shoreline Analysis System (DSAS) will further enable the ability of a shoreline to be mapped and monitored within successive coastal orthomosaic images. In future applications, coastal land/water delineation could be used to effectively optimize DSAS and the rate of change that is prevalent throughout historical shoreline locations (Woods Hole Coastal and Marine Science Center, 2018). Note that accurate assessment of shoreline change requires repeatable accuracies on the order of centimeters. This level of accuracy was not achieved in the examples of this thesis. However, it may be achievable with careful data collection, a systematic gridded survey pattern, and the use of ground control points and static benchmarks for alignment of repeat survey orthomosaic maps. When paired with advanced photogrammetric software such as Agisoft Metashape the orthorectification of oblique images collected in a coastal environment has the potential to benefit a variety of research institutions and federally funded organizations with the ability to perform coastal mapping analyses at a viewing angle that is not generally mapped. By incorporating conventional sUAS technology that is available to the public, amateur mapping analyses will provide end users the ability to effectively produce high-resolution orthomosaic images, thus providing the citizen science community with a valuable tool for coastal mapping.



### **5.6.2 Marine Spatial Planning**

The process and ideology of marine spatial planning focuses on a wide range of environmental studies and organizations from all over the world. Coastal and marine habitats found all over the world are declining at an alarming rate and the ability to monitor these declinations has been insufficient (Foley et al, 2010). Recent geospatial technology has been observed for its potential to use complex spatial mosaics as a form to monitor coastal resources (Collie et al, 2012). Maps that are produced from flying sUAS over areas that are suffering from shoreline change can give us a look at what needs to be done to prevent further erosion, and degradation as well as the devastation caused by strong hurricanes and tropical depressions. When conducting research regarding shoreline change along Jupiter inlet located on the east coast of Florida, Nagarajan states in his findings how information that is acquired when conducting these sUAS flights, eventually will allow government agencies and academic institutions the ability to conduct major research projects regarding the protection coastal areas by developing and enforcing new policies and guidelines (Tsokos et al. 2018, as cited in Nagarajan et al. 2019), this is a prime example of how this monitoring process can be used for multiple different studies and applications. The public must understand the changes that coastal areas are experiencing. A set of methods such as those found in this thesis has the potential to equip these marine spatial planning organizations with a unique and effective tool that can effectively geospatially analyze and monitor how human activities are currently affecting the use of marine and coastal space. Thus, meeting the demand for a cost-effective spatial planning tool with the ability to monitor the environment and help deliver accurate spatial analyses on both a scientific and social spectrum.

### **5.6.3 Storm Damage Assessment**

In the past storm damage assessment within a coastal environment has been performed through manned aircraft flights which at times can be high-priced and inefficient. Coastal areas, as well as the water bodies that surround them, are highly dynamic. Over the past several years marine and coastal areas have suffered significant and, in some cases catastrophic erosion due to rising sea levels and significant storm events (Padua et al. 2017). Human-caused changes such as urban development and population concentrations along coasts are both major causes of shoreline erosion found all over the world (Nagarajan et al. 2019). Natural Hazards such as hurricanes and strong extratropical storms can be more occurring and devastating to coastal areas than any other environment. When a major hurricane or tropical system approaches land, an immense amount of damage can occur to coastal communities. This is due to the violent characteristics that occur within the hurricane phenomenon (i.e., storm surge, inland flooding, heavy rains, and high winds). Abnormal tides increase storm surge depths as coastal estuaries flood into communities found along the coast. Heavy rains cause immense amounts of flooding in low-lying areas, as devastating hurricane-force winds batter the internal urban structure of Marine and coastal environments. Of these exposed coastal areas, the Northern Gulf of Mexico is particularly threatened by major hurricanes. Coastal areas found all over the world have a long extensive history as being a major economic driving factor, these areas are not only responsible for generating seafood, trade, and tourism, but they are also home to some of the richest environmental coastal zones on the planet on both an economic and environmental aspects (Clark, 2016). Not only are coastal areas rich in ecological habitats (i.e., intertidal areas, wetlands, salt marshes, barrier islands, and coral reefs) they also provide bordering coastal communities with access to immense amounts of coastal resources and goods and services (i.e.,

food, fossil fuels, transport and recreation, and trading). Upon analyzing both the visual and statistical accuracy of the final orthomosaic results of the Mississippi Gulf Coast sUAS study site, the methods found herein have the potential to perform accurate, repeatable, and detailed storm damage assessments rapidly immediately following storm events. Due to sUAS being affordable in terms of operating cost, these unmanned platforms enable the possibility of recurrent surveys to be conducted in coastal and marine environments to monitor annual coastal changes as well as the assessment of individual storm impacts within a specific geographic location (Clark, 2016). By visually examining the Mississippi Gulf Coast Final orthomosaic maps for flight 01 (Figure 4.4) and flight 02 (Figure 4.5) found in the previous results section and comparing the RMSE values for both images found in Table 4.2 the potential for an accurate storm assessment is promising. Other methods such as DSAS and land/water delineation analysis would also be effective in monitoring individual storm assessments with the use of historic shoreline change. Thus, allowing small communities and organizations found within them to accurately assess the severity of storm damage using a cost-effective approach.

## CHAPTER VI

### CONCLUSION

The goal of this thesis is to evaluate the capacity of sUAS and ROV platforms to accurately map coastal and marine environments with oblique video imagery. The presented results demonstrate that the developed tools can be successfully used to map seafloor substrate observations derived from ROV video data and to create orthomosaic maps derived from both sUAS and ROV oblique imagery. Additionally, the efficacy of the presented digital mapping approaches was evaluated through the comparison of resultant substrate maps to coincident seafloor backscatter data, and by assessing quantitative agreement between orthorectified images and ground control points as well as coincident surface aerial imagery. That evaluation largely supported the validity of the maps resulting from the application of the developed tools demonstrating that sUAS and ROV can map environments with accuracy comparable to conventional mapping platforms if surveys are designed and conducted to minimize error with components like grided survey patterns (e.g. North Farm Flight 2 RMSE = 0.84m) and the use of a robust number of ground control points. Notably, sUAS and ROV can achieve this comparable level of accuracy while presenting a number of advantages relative to conventional mapping platforms in terms of cost, survey frequency, survey timing, and map resolution.

The presented tools and automated GIS processing framework as well as the resulting map products hold the potential to further geospatial analysis of marine and coastal environments by improving the usability of archived data and increasing the efficiency, affordability,

resolution, and frequency of mapping with unmanned vehicles. Although published literature exists regarding the potential of aerial imagery orthorectification for unmanned vehicles, it is primarily focused on images collected normally to the Earth's surface. The presented research falls within a small percentage of studies that have analyzed the potential that oblique video imagery has to be orthorectified using both ROV and sUAS unmanned platforms. Implementing and optimizing this type of photogrammetric processing schema will provide both the geospatial and environmental geoscience research community with a tool capable of fully automating the mapping process of unmanned vehicle platforms and yielding maps with accuracies comparable to conventional mapping platforms and approaches.

Finally, standard operating procedures (SOPs) for the ROV data processing and map generation have been created and are housed with the associated processing codebase through GitHub ([https://github.com/askarke/ROV\\_Video\\_Mapping\\_CMECS](https://github.com/askarke/ROV_Video_Mapping_CMECS)) in order to further enable the adoption and application of the methods and results presented herein. It is expected that these SOPs and code can be immediately applicable to data collected with the NOAA's ROV Deep Discoverer and archived within NOAA NCEI as well as archived sUAS video data.

## **6.1 ROV Automated Mapping**

The findings within this thesis demonstrate the effectiveness of the presented automated mapping system in accurately representing deep-sea benthic habitats in a geospatial context. Specifically, all 51 deep water dives were successfully automatically mapped and represented in a geospatial and cartographic format. The ability to accurately geospatially represent deep-sea benthic habitats and environmental annotations found within them enhances the ROV data and provides scientists within the oceanographic community with an effective geospatial visualization tool, thus allowing the ROV video and the recorded annotations to be rapidly

mapped – lessening the demand to analyze tens to hundreds of hours of ROV video data (Ruby, 2017). The final map product generated by the ROV automated mapping program uses organized cartographic features to accurately convey geospatial relationships found within a variety of substrate observations while facilitating ocean science and the ability to conduct oceanographic research among a variety of platforms and users.

## **6.2 ROV oblique Image Orthorectification**

The concept of potentially being able to accurately orthorectify oblique images collected from the ROV Deep Discoverers forward-facing camera was anticipated based on the sUAS orthorectification methods and results found in this thesis. After examining the navigation parameters found in the ROV 1Hz dataset it was suspected that the same task that Agisoft Metashape was performing on the sUAS data could also be performed on the ROV data, given that each dataset for the unmanned platforms contains similar vehicle navigational and attitude parameters. Although the resulting orthomosaic images of the seafloor yielded areas of distortion due to zoom and camera tilt orientation the resulting mosaics demonstrate substantial potential for this approach to enable detailed spatial analysis for seafloor imagery. Suggested improvements include collecting the data in a nadiral (downward) viewing direction and surveying the seafloor in a double or single-grid transect should be analyzed in future studies. The ROV video frame extraction and timestamp script created in this thesis can be applied to any video data collected from an ROV. For example, if the ROV Deep Discoverer was to perform a series of dives following a single or double grid dive path along the seafloor using its downward facing (nadiral) camera the video data collected could then be interpolated into the ROV video frame extraction and timestamp script to produce an accurately oriented orthomosaic of the seafloor. Nevertheless, limitation and distortion factors found within oblique imagery must be

absent or manipulated with a nadiral dataset to obtain the best initial orthorectified output. However, the findings in this thesis present the potential and ability to successfully create orthorectified images of the seafloor that were extracted from ROV oblique videography. Thus, allowing other ROV mapping applications such as the viewshed approach to be effectively tested for accuracy, and the validation of bathymetric models to be obtained.

### **6.3 sUAS Oblique Image Orthorectification**

The sUAS results in this thesis demonstrate the ability to accurately orthorectify oblique imagery collected using an sUAS. Given that each of the four selected sUAS flights yielded accurate orthomosaics, as indicated by comparison with satellite imagery and reasonable RMSE values, the mapping of oblique aerial imagery has shown its potential as a useful geospatial tool. The sUAS oblique orthorectification approach can be improved by capturing still oblique images that are collected while the sUAS follows a grid flight pattern (Taddia et al, 2020). However, the results of this thesis demonstrate that the creation of useful orthomosaic maps is achievable even under nonideal sampling situations such as free flight vehicle paths that are not in gridded patterns. Future studies related to sUAS oblique aerial image orthorectification accuracy assessment and validation could approach the following potential improvements for more accurate orthomosaics:

- Perform survey flights in a single or double grid flight transect- This can effectively be conducted during the collection of data when in the field. Varying grid flight patterns would be flown over the study site to collect aerial imagery of the study site on a larger scale (Taddia et al, 2020).
- Incorporate ground control points- Although this is not completely necessary since a high-resolution camera has the ability to record accurate ground features

that can be used for GCPs, it should be tested in future research to compare the RMSE values of the images processed without uniform GCPs.

- Extract exact camera orientation parameters from the sUAS for accurate orthorectification in Agisoft Metashape- Consistent camera orientation parameters extracted from the sUAS for all survey areas would produce better initial georeferenced orthomosaic images in Agisoft Metashape and would not require as much post processing georeferencing to be done in ArcMap.

#### **6.4 Future Work**

The resulting methodology will yield user-friendly open-source digital tools that will ultimately enhance the ability of scientists, environmental managers, and the public to generate maps, conduct geospatial analysis, and derive quantitative results from oblique imagery collected with unmanned vehicles. As mentioned previously, limitations found within both the ROV and sUAS datasets will additionally require revision to make the methods found within this thesis completely operational. The variety and functionality of these unmanned platforms can effectively produce geospatial analyses in marine and coastal environments at an efficient cost and time frame. However, a future and more valid verification method would be possible if the final ROV orthomosaic images were oriented correctly and could thus serve as a basemap for the ROV substrate map and the cartographic features that represent the observed seafloor substrate. For the future implementation of orthorectified sUAS aerial imagery for the classification of land water delineation, a multispectral camera is suggested to obtain NIR imagery for the classification of a land and water boundary. Moreover, future applications that both ROV and sUAS mapping programs will only become more prevalent in the future as geospatial technology increases. A variety of scientific studies and analyses that could potentially benefit from



automated and oblique mapping programs have been prevalent throughout this study. Through future research and data collection, both benthic habitats and shallow coastal environments could greatly benefit from the use of ROV and sUAS technologies.

## REFERENCES

- Acoustic backscatter data. Acoustic backscatter data | U.S. Geological Survey. (n.d.). Retrieved April 14, 2022, from [https://www.usgs.gov/media/images/acoustic-backscatter-data#:~:text=Detailed%20Description,intensity%20\(possibly%20softer%20seafloor%3F\)](https://www.usgs.gov/media/images/acoustic-backscatter-data#:~:text=Detailed%20Description,intensity%20(possibly%20softer%20seafloor%3F))
- Aguirre-Castro, O. A., Inzunza-González, E., García-Guerrero, E. E., Tlelo-Cuautle, E., López-Bonilla, O. R., Olguín-Tiznado, J. E., & Cárdenas-Valdez, J. R. (2019). Design and construction of a roV for underwater exploration. *Sensors (Switzerland)*, 19(24), 1–25. <https://doi.org/10.3390/s19245387>
- Allen, J., & Walsh, B. (2008). Enhanced oil spill surveillance, detection and monitoring through the applied technology of unmanned air systems. *International Oil Spill Conference - IOSC 2008, Proceedings*, 113–120. <https://doi.org/10.7901/2169-3358-2008-1-113>
- Arcgis.com. (n.d.). Retrieved April 28, 2022, from <https://www.arcgis.com/home/item.html?id=10df2279f9684e4a9f6a7f08febac2a9>
- ArcMap. Calculate Transformation Errors (Editing)-ArcMap | Documentation. (n.d.). Retrieved April 28, 2022, from <https://desktop.arcgis.com/en/arcmap/latest/tools/editing-toolbox/calculate-transformation-errors.htm>
- Bassett, RD, M Finkbeiner, PJ Etnoyer. (2017), Application of the Coastal and Marine Ecological Classification Standard (CMECS) to Deep-Sea Benthic Surveys in the Northeast Pacific: Lessons from Field Tests in 2015. NOAA Tech Memo NOS NCCOS 228, NOAA National Ocean Service, Charleston, SC 29412. 49 pp.
- Calvario, G., Sierra, B., Alarcón, T. E., Hernandez, C., & Dalmau, O. (2017). A multi-disciplinary approach to remote sensing through low-cost UAVs. *Sensors (Switzerland)*, 17(6). <https://doi.org/10.3390/s17061411>
- Chandler, J., Ashmore, P., Paola, C., Gooch, M., & Varkaris, F. (2002). Monitoring river-channel change using terrestrial oblique digital imagery and automated digital photogrammetry. *Annals of the Association of American Geographers*, 92(4), 631–644. <https://doi.org/10.1111/1467-8306.00308>
- Casella, E., Rovere, A., Pedroncini, A., Mucerino, L., Casella, M., Cusati, L. A., Vacchi, M., Ferrari, M., & Firpo, M. (2014). Study of wave runup using numerical models and low-altitude aerial photogrammetry: A tool for coastal management. *Estuarine, Coastal and Shelf Science*, 149, 160–167. <https://doi.org/10.1016/j.ecss.2014.08.012>

- Clague, D., Paduan, J., Caress, D., Chadwick, W., Le Saout, M., Dreyer, B., & Portner, R. (2017). High-Resolution AUV Mapping and Targeted ROV Observations of Three Historical Lava Flows at Axial Seamount. *Oceanography*, 30(4).  
<https://doi.org/10.5670/oceanog.2017.426>
- Clark, A. (2017). Small unmanned aerial systems comparative analysis for the application to coastal erosion monitoring. *GeoResJ*, 13, 175–185.  
<https://doi.org/10.1016/j.grj.2017.05.001>
- Collie, J. S., Vic Adamowicz, W. L., Beck, M. W., Craig, B., Essington, T. E., Fluharty, D., Rice, J., & Sanchirico, J. N. (2013). Marine spatial planning in practice. *Estuarine, Coastal and Shelf Science*, 117, 1–11. <https://doi.org/10.1016/j.ecss.2012.11.010>
- Comittee, F. G. D. (2012). Coastal and marine ecological classification standard, June 2012. National Oceanic and Atmospheric Administration, 343.  
[http://webqa.csc.noaa.gov/benthic/cmecs/Version\\_III\\_Official\\_Review\\_Draft.doc](http://webqa.csc.noaa.gov/benthic/cmecs/Version_III_Official_Review_Draft.doc)
- Costanza, R., & Farley, J. (2007). Ecological economics of coastal disasters: Introduction to the special issue. *Ecological Economics*, 63(2–3), 249–253.  
<https://doi.org/10.1016/j.ecolecon.2007.03.002>
- Dias, F. C., Gomes-Pereira, J., Tojeira, I., Souto, M., Afonso, A., Calado, A., Madureira, P., & Campos, A. (2015). Area estimation of deep-sea surfaces from oblique still images. *PLoS ONE*, 10(7), 1–14. <https://doi.org/10.1371/journal.pone.0133290>
- Digital Shoreline Analysis System (DSAS) active. Digital Shoreline Analysis System (DSAS) | U.S. Geological Survey. (n.d.). Retrieved April 28, 2022, from <https://www.usgs.gov/centers/whcms/science/digital-shoreline-analysis-system-dsas>
- Dolan, M. F. J., Grehan, A. J., Guinan, J. C., & Brown, C. (2008). Modelling the local distribution of cold-water corals in relation to bathymetric variables: Adding spatial context to deep-sea video data. *Deep-Sea Research Part I: Oceanographic Research Papers*, 55(11), 1564–1579. <https://doi.org/10.1016/j.dsr.2008.06.010>
- Dunford, R., K. Michel, M. Gagnage, H. Piégay, and M. L. Trémelo (2009), Potential and constraints of Unmanned Aerial Vehicle technology for the characterization of Mediterranean riparian forest, *Int. J. Remote Sens.*, 30(19), 4915–4935, doi:10.1080/01431160903023025.
- Elkhrachy, I. (2021). Accuracy Assessment of Low-Cost Unmanned Aerial Vehicle (UAV) Photogrammetry. *Alexandria Engineering Journal*, 60(6), 5579–5590.  
<https://doi.org/10.1016/j.aej.2021.04.011>

- ESTIMATION OF MEASUREMENT UNCERTAINTY OF SEAFLOOR ACOUSTIC BACKSCATTER BY BSc ( Hons .) Marine Sciences , Karachi University , Pakistan , 1998 MS Ocean Engineering ( Ocean Mapping ), University of New Hampshire , USA , 2005 DISSERTATION Submitted to the U. (2019). December.
- Etnoyer, PJ, M Malik, D Sowers, C Ruby, R Bassett, J Dijkstra, N pawlenko, S Gottfried, K Mello, M Finkbeiner, and A Sallis. 2018. Working with video to improve deep-sea habitat characterization. In Raineault, N.A., J Flanders, and A. Bowman, eds. 2018. New frontiers in ocean exploration: The E/V Nautilus, NOAA Ship Okeanos Explorer, and R/V Falkor 2017 field season. <https://tos.org/oceanography/issue/volume-31-issue-01-supplement>
- Fernandes, O., Murphy, R., Merrick, D., Adams, J., Hart, L., & Broder, J. (2019). Quantitative Data Analysis: Small Unmanned Aerial Systems at Hurricane Michael. 2019 IEEE International Symposium on Safety, Security, and Rescue Robotics, SSRR 2019, 116–117. <https://doi.org/10.1109/SSRR.2019.8848935>
- Foley, M. M., Halpern, B. S., Micheli, F., Armsby, M. H., Caldwell, M. R., Crain, C. M., Prahl, E., Rohr, N., Sivas, D., Beck, M. W., Carr, M. H., Crowder, L. B., Emmett Duffy, J., Hacker, S. D., McLeod, K. L., Palumbi, S. R., Peterson, C. H., Regan, H. M., Ruckelshaus, M. H., ... Steneck, R. S. (2010). Guiding ecological principles for marine spatial planning. *Marine Policy*, 34(5), 955–966. <https://doi.org/10.1016/j.marpol.2010.02.001>
- Hewitt, J. E., Thrush, S. F., Legendre, P., Funnell, G. A., Ellis, J., & Morrison, M. (2004). Mapping of marine soft-sediment communities: Integrated sampling for ecological interpretation. *Ecological Applications*, 14(4), 1203–1216. <https://doi.org/10.1890/03-5177>
- Hodgson, M. E., & Morgan, G. R. (2021). Modeling sensitivity of topographic change with sUAS imagery. *Geomorphology*, 375, 107563. <https://doi.org/10.1016/j.geomorph.2020.107563>
- Höhle, J. (2008). Photogrammetric measurements in oblique aerial images. *Photogrammetrie, Fernerkundung, Geoinformation*, 1(HEFT 1), 7–14.
- Hugenholtz, C. H., B. J. Moorman, K. Riddell, and K. Whitehead (2012), Small unmanned aircraft systems for remote sensing and Earth science research, *Eos (Washington. DC).*, 93(25), doi:10.1029/2012EO250005.
- Hugenholtz, C. H., K. Whitehead, O. W. Brown, T. E. Barchyn, B. J. Moorman, A. LeClair, K. Riddell, and T. Hamilton (2013), Geomorphological mapping with a small unmanned aircraft system (sUAS): Feature detection and accuracy assessment of a photogrammetrically-derived digital terrain model, *Geomorphology*, 194, 16–24, doi:10.1016/j.geomorph.2013.03.023.

- Ju, J., and D. P. Roy (2008), The availability of cloud-free Landsat ETM+ data over the conterminous United States and globally, *Remote Sens. Environ.*, 112(3), doi:10.1016/j.rse.2007.08.011.
- Kandrot, S., & Holloway, P. (2020). Applications of Drone Technology for Sustainable Development of the Coastal Zone : A Literature Review. *Sustainable Resilient Coasts*, July, 1–51.
- Kington, K. (2018). Applying the coastal and marine ecological classification standard (CMECS) to nearshore habitats in the northeastern Gulf of Mexico. *Geosciences* (Switzerland), 8(1). <https://doi.org/10.3390/geosciences8010022>
- Klemas, V. (2011), Remote sensing techniques for studying coastal ecosystems: An overview, *J. Coast. Res.*, 27(1), 2–17, doi:10.2112/JCOASTRES-D-10-00103.1.
- Laliberte, A. S., Winters, C., & Rango, A. (2008). a Procedure for Orthorectification of Sub-Decimeter Resolution Imagery Obtained With an Unmanned Aerial Vehicle ( Uav ) Uav Platform , Sensors and Image Acquisition. ASPRS 2008 Annual Conference.
- Lin, Y. C., Cheng, Y. T., Zhou, T., Ravi, R., Hasheminasab, S. M., Flatt, J. E., Troy, C., & Habib, A. (2019). Evaluation of UAV LiDAR for mapping coastal environments. *Remote Sensing*, 11(24), 1–33. <https://doi.org/10.3390/rs11242893>
- Lumiatti, G., Carley, J. T., Drummond, C. D., & Vos, K. (2019). Use of emerging remote sensing technologies for measuring long-term shoreline change and coastal management. In: *Australasian Coasts and Ports 2019 Conference: Future directions from 40 [degrees] S and beyond*, Hobart, 10-13 September 2019. Hobart: Engineers Australia, 2019: 797-803. Engineers Australia. <https://search.informit.org/doi/10.3316/informit.800049591264258>
- Lundsten, L., Schlining, K. L., Frasier, K., Johnson, S. B., Kuhnz, L. A., Harvey, J. B. J., Clague, G., & Vrijenhoek, R. C. (2010). Time-series analysis of six whale-fall communities in Monterey Canyon, California, USA. *Deep-Sea Research Part I: Oceanographic Research Papers*, 57(12), 1573–1584. <https://doi.org/10.1016/j.dsr.2010.09.003>
- Lurton, X. (2010), *An Introduction to Underwater Acoustics: Principles and Applications*.
- Macreadie, P. I., McLean, D. L., Thomson, P. G., Partridge, J. C., Jones, D. O. B., Gates, A. R., Benfield, M. C., Collin, S. P., Booth, D. J., Smith, L. L., Techera, E., Skropeta, D., Horton, T., Pattiaratchi, C., Bond, T., & Fowler, A. M. (2018). Eyes in the sea: Unlocking the mysteries of the ocean using industrial, remotely operated vehicles (ROVs). *Science of the Total Environment*, 634, 1077–1091. <https://doi.org/10.1016/j.scitotenv.2018.04.049>
- Manley, J. E. (2008). New tools for ocean exploration, equipping the NOAA ship okeanos explorer. *Oceans 2008*. <https://doi.org/10.1109/OCEANS.2008.5151876>

- Marsh, L., Copley, J. T., Huvenne, V. A. I., Tyler, P. A., & the Isis ROV Facility. (2013). Getting the bigger picture: Using precision Remotely Operated Vehicle (ROV) videography to acquire high-definition mosaic images of newly discovered hydrothermal vents in the Southern Ocean. *Deep-Sea Research Part II: Topical Studies in Oceanography*, 92, 124–135. <https://doi.org/10.1016/j.dsr2.2013.02.007>
- Mayer, L. A. (2006), Frontiers in seafloor mapping and visualization, *Mar. Geophys. Res.*, 27(1), 7–17, doi:10.1007/s11001-005-0267-x.
- McBride, R. A., & Byrnes, M. R. (1997). Regional variations in shore response along barrier island systems of the Mississippi River delta plain: Historical change and future prediction. *Journal of Coastal Research*, 13(3), 628–655.
- McLean, D. L., Parsons, M. J. G., Gates, A. R., Benfield, M. C., Bond, T., Booth, D. J., Bunce, M., Fowler, A. M., Harvey, E. S., Macreadie, P. I., Pattiaratchi, C. B., Rouse, S., Partridge, J. C., Thomson, P. G., Todd, V. L. G., & Jones, D. O. B. (2020). Enhancing the Scientific Value of Industry Remotely Operated Vehicles (ROVs) in Our Oceans. *Frontiers in Marine Science*, 7(April). <https://doi.org/10.3389/fmars.2020.00220>
- Morgan, G. R., Wang, C., & Morris, J. T. (2021). Rgb indices and canopy height modelling for mapping tidal marsh biomass from a small unmanned aerial system. *Remote Sensing*, 13(17), 1–18. <https://doi.org/10.3390/rs13173406>
- Nagarajan, S., Khamaru, S., & De Witt, P. (2019). UAS based 3D shoreline change detection of Jupiter Inlet Lighthouse ONA after Hurricane Irma. *International Journal of Remote Sensing*, 40(24), 9140–9158. <https://doi.org/10.1080/01431161.2019.1569792>
- Nesbit, P. R., & Hugenholtz, C. H. (n.d.). remote sensing Enhancing UAV-SfM 3D Model Accuracy in High-Relief Landscapes by Incorporating Oblique Images. <https://doi.org/10.3390/rs11030239>
- NOAA Office of Ocean Exploration and Research. (n.d.). Retrieved April 4, 2022, from <https://oceanexplorer.noaa.gov/>
- Orange, D. L., Yun, J., Maher, N., Barry, J., & Greene, G. (2002). Tracking California seafloor seeps with bathymetry, backscatter and ROVs. *Continental Shelf Research*, 22(16), 2273–2290. [https://doi.org/10.1016/S0278-4343\(02\)00054-7](https://doi.org/10.1016/S0278-4343(02)00054-7)
- Patel, K., Jain, R., Patel, A. N., & Kalubarme, M. H. (n.d.). Shoreline change monitoring for coastal zone management using multi-temporal Landsat data in Mahi River estuary, Gujarat State. <https://doi.org/10.1007/s12518-021-00353-8/Published>
- Petrie, G. (2009). Systematic Oblique Aerial Photography Using Multiple Digital Cameras. *Prezentacja!*, Feb 2009, 102–107.
- Phantom 4 Pro - DJI. DJI Official. (n.d.). Retrieved April 4, 2022, from <https://www.dji.com/phantom-4-pro>

- Prost, G.L. (2013). Remote Sensing for Geoscientists: Image Analysis and Integration, Third Edition (3rd ed.). CRC Press. <https://doi.org/10.1201/b15638>
- Raineault, N. (2018). New frontiers in ocean exploration: the E/V nautilus, NOAA ship Okeanos Explorer, and R/V falkor 2017 field season. *Oceanography*, 31(1), 1–126. <https://doi.org/10.5670/oceanog.2018.supplement.01>
- Ruby, C. (2017). Application of Coastal and Marine Ecological Classification Standard (CMECS) to remotely operated vehicle (ROV) video data for enhanced geospatial analysis of deep sea environments. M.S. thesis, Mississippi State University.
- Rupnik, E., Nex, F., & Remondino, F. (2013). Automatic Orientation of Large Blocks of Oblique Images. *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, XL-1/W1(June 2014), 299–304. <https://doi.org/10.5194/isprsarchives-xl-1-w1-299-2013>
- Schlining, K., von Thun, S., Kuhn, L., Schlining, B., Lundsten, L., Jacobsen Stout, N., Chaney, L., & Connor, J. (2013). Debris in the deep: Using a 22-year video annotation database to survey marine litter in Monterey Canyon, central California, USA. *Deep-Sea Research Part I: Oceanographic Research Papers*, 79, 96–105. <https://doi.org/10.1016/j.dsr.2013.05.006>
- Sturdivant, E. J., Lentz, E. E., Robert Thieler, E., Farris, A. S., Weber, K. M., Remsen, D. P., Miner, S., & Henderson, R. E. (n.d.). remote sensing UAS-SfM for Coastal Research: Geomorphic Feature Extraction and Land Cover Classification from High-Resolution Elevation and Optical Imagery. <https://doi.org/10.3390/rs9101020>
- Taddia, Y., Stecchi, F., & Pellegrinelli, A. (2020). Coastal mapping using dji phantom 4 RTK in post-processing kinematic mode. *Drones*, 4(2), 1–19. <https://doi.org/10.3390/drones4020009>
- Tsokos, A., Kotsi, E., Petrakis, S., & Vassilakis, E. (2018). Combining series of multi-source high spatial resolution remote sensing datasets for the detection of shoreline displacement rates and the effectiveness of coastal zone protection measures. *Journal of Coastal Conservation*, 22(2), 431–441. <https://doi.org/10.1007/s11852-018-0591-3>
- US Department of Commerce, N. O. and A. A. (2021, August 16). NOAA ship okeanos explorer: NOAA ocean exploration. NOAA Ship Okeanos Explorer: NOAA Ocean Exploration. Retrieved April 4, 2022, from <https://oceanexplorer.noaa.gov/okeanos/welcome.html>
- Vertical vs oblique imagery - support. (n.d.). Retrieved April 4, 2022, from <https://support.pix4d.com/hc/en-us/articles/202559859-Vertical-vs-oblique-imagery>
- Verykokou, S., & Ioannidis, C. (2018). Oblique aerial images: a review focusing on georeferencing procedures. *International Journal of Remote Sensing*, 39(11), 3452–3496. <https://doi.org/10.1080/01431161.2018.1444294>

- Whitehead, K., & Hugenholtz, C. H. (2014). Remote sensing of the environment with small, unmanned aircraft systems (Uass), part 1: A review of progress and challenges. *Journal of Unmanned Vehicle Systems*, 2(3), 69–85. <https://doi.org/10.1139/juvs-2014-0006>
- Wiedemann, A., & Moré, J. (2012). Orientation Strategies for Aerial Oblique Images. *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, XXXIX-B1(September), 185–189. <https://doi.org/10.5194/isprsarchives-xxxix-b1-185-2012>
- Wright, L. D., J. P. M. Syvitski, and C. R. Nichols (2019), Coastal systems in the Anthropocene, in *Coastal Research Library*, vol. 27.
- Yahyanejad, S., M. Quaritsch, and B. Rinner (2011), Incremental, orthorectified and loop-independent mosaicking of aerial images taken by micro UAVs, *ROSE 2011 - IEEE Int. Symp. Robot. Sensors Environ. Proc.*, 137–142, doi:10.1109/ROSE.2011.6058531.
- Zhou, G. (2009), Near real-time orthorectification and mosaic of small UAV video flow for time-critical event response, *IEEE Trans. Geosci. Remote Sens.*, 47(3), 739–747, doi:10.1109/TGRS.2008.2006505.
- Zhou, Q., & Liu, J. (2015). Automatic orthorectification and mosaicking of oblique images from a zoom lens aerial camera. *Optical Engineering*, 54(1), 013104. <https://doi.org/10.1117/1.oe.54.1.013104>



APPENDIX A

ROV AUTOMATED MAPPING PYTHON SCRIPTS

```

import csv, sys
import pdb

# This is where the input and output annotation files are entered. The outfile must be set in a new file folder within the current working folder.
# Nothing in this script should be edited except the "file" and "outfile".
file = ('C:\\Users\\Jakeb\\OneDrive\\Desktop\\Graduate Research- Desktop\\Dive_annotation_files\\EX 1806\\Dive14old.csv')
outfile = ('C:\\Users\\Jakeb\\OneDrive\\Desktop\\Graduate Research- Desktop\\Dive_annotation_files\\EX 1806\\new\\Dive#14_new.csv')

with open(file, 'r') as csvfile:
    reader = csv.DictReader(csvfile)

    with open(outfile, 'w') as csvoutfile:
        writer = csv.DictWriter(csvoutfile, lineterminator='\n', fieldnames = reader.fieldnames)
        writer.writeheader()

        for row in reader:
            try:
                # Grab substrate string.
                reclass = row['Substrate']
                #pdb.set_trace()
                # Find start index of "Primary" in string
                findstart = reclass.index('Primary')
                # Take from start of "Primary" until the end of the string
                reclass = reclass[findstart:len(reclass)]

                # Example: Primary Unconsolidated Secondary Unconsolidated

                # Split string into a list
                reclass_list = reclass.split(' ')

                #pdb.set_trace()

                # Find where "Primary" is in list
                pindex = reclass_list.index('Primary')
                # Grab next list value for primary classification
                primary = reclass_list[pindex + 1]

                # Find where "Secondary" is in list
                sindex = reclass_list.index('Secondary')
                # Grab next list value for secondary classification
                secondary = reclass_list[sindex + 1]

                # Get rid of anything after colon

                # Example: Unconsolidated:Sloping (5 to <30 Degrees)
                # [Unconsolidated, Sloping (5 to <30 Degrees)]
                # Take index of first and assign to secondary class variable

                secondary = secondary.split(':')[0]

                # Set up empty string
                reclass_string = ""

                # Build up primary portion of string
                primary = primary.lower()
                if primary == 'unconsolidated':
                    reclass_string = reclass_string + 'Fine:'

```

Figure A.1 *CMECS classification script.py*

The *CMECS classification script.py* converts seafloor substrate annotations into a new format that is compliant with the current CMECS standard.

```

        reclass_string = reclass_string + 'Fine:'
    elif primary == 'rock':
        reclass_string = reclass_string + 'Rock:'
    elif primary == 'reef' or primary == 'rubble' or primary == 'hash':
        reclass_string = reclass_string + 'Coarse:'
    else:
        print ("Primary: No idea what to do with this!")

    # Build up secondary portion of string
    secondary = secondary.lower()
    if secondary == 'unconsolidated':
        reclass_string = reclass_string + 'Fine'
    elif secondary == 'rock':
        reclass_string = reclass_string + 'Rock'
    elif secondary == 'reef' or secondary == 'rubble' or secondary == 'hash':
        reclass_string = reclass_string + 'Coarse'
    else:
        print ("Secondary: No idea what to do with this!")

    # Write new string back to row.
    row['Substrate'] = reclass_string
    writer.writerow(row)
except:
    print ("Error in parsing line.")
    print (reclass)

```

Figure A.1 *CMECS classification script.py* continued

```

# Geospatial analysis of deep-sea environments using ROV video data with the Coastal and Marine Ecological Classification Standard (CMECS).
# Creator: Jacob Freeman
# Institution: Mississippi State University
# College: Department of Geosciences

# This script was designed to run in PyQGIS.

# The first create points layer from table algorithm allows 1 hz rovd data to be mapped in the form of a point shapefile.
# INPUT of this algorithm will always remain a 1 hz data file from the ROV in csv format.
# XFIELD will remain 'LON_DD' as long as the user does not make manual corrections to the 1 hz csv file.
# YFIELD will remain 'LAT_DD' as long as the user does not make manual corrections to the 1 hz csv file.
# ZFIELD will remain None.
# TARGET_CRS will remain QgsCoordinateReferenceSystem('EPSG:4326').
# OUTPUT should be the working folder of the user, or a desired folder for the output shapefile to be stored.
# onehzpath will always remain ['OUTPUT'].
result = processing.run("qgis:createpointslayerfromtable",
{'INPUT': 'C:\\Users\\Jakeb\\OneDrive\\Desktop\\Graduate Research- Desktop\\1 hz dive files\\EX 1903L2 Dives\\EX1903L2_DIVE11_20190703\\EX1903L2_DIVE11_RovTrack1Hz.csv',
'XFIELD': 'LON_DD',
'YFIELD': 'LAT_DD',
'ZFIELD': None,
'TARGET_CRS': QgsCoordinateReferenceSystem('EPSG:4326'),
'OUTPUT': 'C:\\Users\\Jakeb\\OneDrive\\Desktop\\Graduate Research- Desktop\\NOAA Shapefiles\\path_dive_EX1903#11.shp'})
onehzpath = result['OUTPUT']

# This points to path algorithm creates a dive path from 1 hz dives points that were generated from the previous create points layer from table.
# INPUT of this algorithm will always remain as the output parameter that is set in the previous algorithm. (Example: onehzpath).
# ORDER_FIELD will remain date.
# GROUP_FIELD will remain None.
# DATE_FORMAT will remain ''.
# OUTPUT should be the working folder of the user, or a desired folder for the output shapefile to be stored.
# divepath will always remain ['OUTPUT'].
result = processing.run("qgis:pointstopath",
{'INPUT': onehzpath,
'ORDER_FIELD': 'DATE',
'GROUP_FIELD': None,
'DATE_FORMAT': '',
'OUTPUT': 'C:\\Users\\Jakeb\\OneDrive\\Desktop\\Graduate Research- Desktop\\NOAA Shapefiles\\Dive Path.shp'})
divepath = result['OUTPUT']

# The second create points layer from table allows the location of annotations to be mapped in the form of a point shapefile.
# INPUT of this algorithm will always remain a Dive annotation data file from the ROV in csv format, that is acquired from seabeam.
# XFIELD will remain 'Environment - Longitude' as long as the user does not make manual corrections to the annotation csv file.
# YFIELD will remain 'Environment - Latitude' as long as the user does not make manual corrections to the 1 annotation csv file.
# ZFIELD will remain None.
# TARGET_CRS will remain QgsCoordinateReferenceSystem('EPSG:4326').
# OUTPUT should be the working folder of the user, or a desired folder for the output shapefile to be stored.
# annotation will always remain ['OUTPUT'].
result = processing.run("qgis:createpointslayerfromtable",
{'INPUT': 'C:\\Users\\Jakeb\\OneDrive\\Desktop\\Graduate Research- Desktop\\Dive_annotation_files\\EX 1903 L2\\Dive11.csv',
'XFIELD': 'Environment - Longitude',
'YFIELD': 'Environment - Latitude',
'ZFIELD': None,
'TARGET_CRS': QgsCoordinateReferenceSystem('EPSG:4326'),
'OUTPUT': 'C:\\Users\\Jakeb\\OneDrive\\Desktop\\Graduate Research- Desktop\\Graduate Research\\NOAA Shapefiles\\annotation_points_EX1903_Dive#11.shp'})
annotation = result['OUTPUT']

```

Figure A.2 Mapping Script.py

*The Mapping Script.py* generates digital shapefiles of the ROV dive path (line feature) and classified seafloor viewsheds (polygon features).

```

# This algorithm will create wedge buffers showing the viewing direction of the ROV as well as substrate type.
# INPUT will remain annotation, this allows the wedges to be placed where observations were made along the seafloor.
# AZIMUTH will always remain QgsProperty.fromExpression("attribute('Environment_1')").
# WIDTH is currently set to 44. This parameter can be changed each time the code is ran if needed.
# OUTER_RADIUS the output of this parameter is generated best if the radius is set to 0.0005.
# INNER_RADIUS should remain 0.
# OUTPUT should be the working folder of the user, or a desired folder for the output shapefile to be stored.
# wedgebuffer will always remain ['OUTPUT'].
result = processing.run("native:wedgebuffer",
{'INPUT': annotation,
'AZIMUTH': QgsProperty.fromExpression("attribute('Environment_1')"),
'WIDTH': 44,
'OUTER_RADIUS': 0.0005,
'INNER_RADIUS': 0,
'OUTPUT': "C:\\Users\\Jakeb\\OneDrive\\Desktop\\Graduate Research- Desktop\\Graduate Research\\NOAA Shapefiles\\wedge_buffers_EX1903_dive#11.shp"})
wedgebuffer = result['OUTPUT']

# This algorithm pulls the output of the wedgebuffer and takes common substrate observations and dissolves boundaries between them.
# INPUT will always remain wedgebuffer.
# DISSOLVE_ALL will always remain None.
# FIELD will always remain 'Substrate'
# OUTPUT should be the working folder of the user, or a desired folder for the output shapefile to be stored.
result = processing.run('qgis:dissolve',
{'INPUT': wedgebuffer,
'DISSOLVE_ALL': None,
'FIELD': 'Substrate',
'OUTPUT': "C:\\Users\\Jakeb\\OneDrive\\Desktop\\Graduate Research- Desktop\\NOAA Shapefiles\\OHECS Substrate Units.shp"})
dissolvebuffer = result ['OUTPUT']

# This section of the program loads the shapefiles previously generated, into QGIS.
# qmlfile sets the exact color and symbology for the dissolve buffer. This should always remain the same. The substrate qml file should be stored in the same working folder as all other shapefiles and scripts used to generate the maps.
# fileInfo will always remain QgsFileInfo(dissolvebuffer).
# basename will always remain fileInfo.basename().
# rlayer will always remain QgsVectorLayer(dissolvebuffer, basename).
# The if statement and the parameters found within it should all remain the same.
qmlfile = "C:\\Users\\Jakeb\\OneDrive\\Desktop\\Graduate Research- Desktop\\NOAA qml files\\Substrate type by color.v2.qml"
fileinfo = QgsFileInfo(dissolvebuffer)
basename = fileInfo.basename()
rlayer = QgsVectorLayer(dissolvebuffer, basename)
if rlayer.isValid():
    print ('Layer loaded!')
QgsProject.instance().addMapLayer(rlayer)
processing.run("native:setlayerstyle", ('INPUT': dissolvebuffer, 'STYLE': qmlfile)) ##returns nothing.....
extent = rlayer.extent()
rlayer.triggerRepaint()

# This section of the code loads and classifys the dive path previously generated, into QGIS.
# qmlfile sets the exact color and symbology for the dive path. This should always remain the same. The dive path qml file should be stored in the same working folder as all other shapefiles and scripts used to generate the maps.
# fileInfo will always remain QgsFileInfo(divepath).
# basename will always remain fileInfo.basename().
# rlayer will always remain QgsVectorLayer(divepath, basename).
# The if statement and the parameters found within it should all remain the same.
qmlfile = "C:\\Users\\Jakeb\\OneDrive\\Desktop\\Graduate Research- Desktop\\NOAA qml files\\divepath.qml"
fileinfo = QgsFileInfo(divepath)
basename = fileInfo.basename()
rlayer = QgsVectorLayer(divepath, basename)
if rlayer.isValid():
    print ('Layer loaded!')
QgsProject.instance().addMapLayer(rlayer)
processing.run("native:setlayerstyle", ('INPUT': divepath, 'STYLE': qmlfile))
extent = rlayer.extent()
rlayer.triggerRepaint()

```

Figure A.2 *Mapping Script.py* continued

```

from qgis.PyQt import QtGui
# Jacob Freeman
# Mississippi State University
# NOAA Final Map Production Script

# This particular section of code allows you to enter the map layers you want to project in the layout.
# For this automated map, we want both the Dive Path, and CMECS Substrate Units to be mapped.
# Both the Dive Path and CMECS substrate layers should always be called in order for them to be mapped in the final layout.
layers = QgsProject.instance().mapLayersByName('Dive Path')
layer = layers[0]
layers2 = QgsProject.instance().mapLayersByName('CMECS Substrate Units')
layer2 = layers2[0]

project = QgsProject.instance()
manager = project.layoutManager()
# layoutName allows you to assign a final name to the map layout.
layoutName = 'Dive Map layout'
layouts_list = manager.printLayouts()
# This will remove and duplicate layouts
# These parameters must stay the same in order for the desired map to be properly created.
for layout in layouts_list:
    if layout.name() == layoutName:
        manager.removeLayout(layout)
layout = QgsPrintLayout(project)
layout.initializeDefaults()
layout.setName(layoutName)
manager.addLayout(layout)

# create map item in the layout
# This line of code sets the size of the rectangle for the map.
# These parameters must stay the same in order for the desired map to be properly created.
map = QgsLayoutItemMap(layout)
map.setRect(20, 20, 20, 20)

# set the map extent
ms = QgsMapSettings()
ms.setLayers([layer, layer2])
# set layers to be mapped
rect = QgsRectangle(ms.fullExtent())
rect.scale(1.1)
ms.setExtent(rect)
map.setExtent(rect)
map.setBackgroundColor(QColor(255, 255, 255, 0))
layout.addLayoutItem(map)

# Creates the frame around the map.
# If a frame is not desired for the final map, the true statement must be changed to false.
map.setFrameEnabled(True)
# Both of these map parameters position the map to be placed at a desired location within the layout.
map.attemptMove(QgsLayoutPoint(5, 25, QgsUnitTypes.LayoutMillimeters))
map.attemptResize(QgsLayoutSize(200, 169, QgsUnitTypes.LayoutMillimeters))
# This section of code creates the legend for the map.
# The only line of code that should be changed in this section is the title.
legend = QgsLayoutItemLegend(layout)
# If you wish to change the title for the legend manually enter the new title name.
# The title must have quotations around it.
legend.setTitle("") # Title of legend can be changed.
layerTree = QgsLayerTree()

```

Figure A.3 *Map production script.py*

The *Map production script.py* produces a final ROV substrate map in .pdf format

```

layerTree = QgsLayerTree()
layerTree.addLayer(layer)
layerTree.addLayer(layer2)
legend.model().setRootGroup(layerTree)
layout.addLayoutItem(legend)
legend.attemptMove(QgsLayoutPoint(226.954, 114.926, QgsUnitTypes.LayoutMillimeters))

# This section of code creates the scalebar for the final map.
scalebar = QgsLayoutItemScaleBar(layout)
# There are many different types of scalebars that can be used during the production of the final map.
# For this particular dive I have set the code to create a scalebar 'Line Ticks Up'.
scalebar.setStyle('Line Ticks Up') # Type of scalebar can be edited.
# Sets the type of units that the scalebar will use to measure distance.
# For this map I chose to use Meters.
scalebar.setUnits(QgsUnitTypes.DistanceMeters)
# Sets the number of segments in the scalebar.
scalebar.setNumberOfSegments(2)
scalebar.setNumberOfSegmentsLeft(0)
scalebar.setUnitsPerSegment(200)
# Links the current map with the scalebar.
scalebar.setLinkedMap(map)
# Sets the unit label for the scalebar.
scalebar.setUnitLabel('m')
# Sets the font style and size for the scalebar.
scalebar.setFont(QFont('Arial', 15)) # This is where scalebar font cant be edited.
scalebar.update()
layout.addLayoutItem(scalebar)
# Moves the scalebar to the desired location on the map layout.
# For this particular map, these parameters should remain the same.
scalebar.attemptMove(QgsLayoutPoint(4.7986, 195.063, QgsUnitTypes.LayoutMillimeters))

# This section Creates the title for the final map.
# For each new dive being mapped this is where the title will be manually entered.
title = QgsLayoutItemLabel(layout)
title.setText("EX 1806 Dive 6") # This is where a new map title will be entered.
title.setFont(QFont('Arial', 24))
title.adjustSizeToText()
layout.addLayoutItem(title)
title.attemptMove(QgsLayoutPoint(70, 5, QgsUnitTypes.LayoutMillimeters))

# Creates the subtitle, showing date that the dive was conducted.
# For each new dive being mapped this is where the subtitle will be manually entered.
map_label = QgsLayoutItemLabel(layout)
map_label.setText("Date: June 20, 2019") # This is where a new subtitle will be entered.
map_label.setFont(QFont("Arial", 12))
map_label.adjustSizeToText()
layout.addLayoutItem(map_label)
map_label.attemptMove(QgsLayoutPoint(208.454, 100.689, QgsUnitTypes.LayoutMillimeters))

# This section of code creates and formats a text box that projects the coordinate system used to create the map.
# For this particular mapping analysis the coordinate system should remain the same.
map_label = QgsLayoutItemLabel(layout)
map_label.setText("ESPG: 4326 - WGS 84")
map_label.setFont(QFont("Arial", 15))
map_label.adjustSizeToText()

```

Figure A.3 *Map production script.py continued*

```

map_label.adjustSizeToText()
layout.addLayoutItem(map_label)
map_label.attemptMove(QgsLayoutPoint(239.473, 199.485, QgsUnitTypes.LayoutMillimeters))

# This section of code creates and formats a text box showing the max depth that the ROV reached during the dive.
# This text box must be edited for the creation of a new map.
# Each dive will have a different max depth.
map_label = QgsLayoutItemLabel(layout)
map_label.setText("Max Depth: 788 Meters") # The text can be edited here.
map_label.setFont(QFont("Arial", 12))
map_label.adjustSizeToText()
layout.addLayoutItem(map_label)
map_label.attemptMove(QgsLayoutPoint(208.454, 106.151, QgsUnitTypes.LayoutMillimeters))

# This section of code creates and formats a text box showing the bottom time of the ROV during the dive.
# This box must be edited for the creation of a new map.
# Each dive will have a different bottom time.
map_label = QgsLayoutItemLabel(layout)
map_label.setText("Bottom Time: 5 Hours 16 Minutes 55 Seconds") # Bottom time can be edited here.
map_label.setFont(QFont("Arial", 12)) # Text can be edited here.
map_label.adjustSizeToText()
layout.addLayoutItem(map_label)
map_label.attemptMove(QgsLayoutPoint(208.454, 111.614, QgsUnitTypes.LayoutMillimeters))

# Creates NOAA logo.
img = QgsLayoutItemPicture(layout)
img.setPicturePath("C:\\Users\\Jakeb\\OneDrive\\Desktop\\Graduate Research- Desktop\\NOAA Automated Mapping Project\\NOAA Map Images\\NOAA.jpg")

# set the image size
img.attemptResize(QgsLayoutSize(68.9972, 69.0024, QgsUnitTypes.LayoutMillimeters))
layout.addLayoutItem(img)
# move to exact position
img.attemptMove(QgsLayoutPoint(213.652, 24.6785, QgsUnitTypes.LayoutMillimeters))

#Sets north arrow.
img = QgsLayoutItemPicture(layout)
img.setPicturePath("C:\\Users\\Jakeb\\OneDrive\\Desktop\\Graduate Research- Desktop\\NOAA Automated Mapping Project\\NOAA Map Images\\North_Arrow.JPG")

# set the image size
img.attemptResize(QgsLayoutSize(8.93797, 17.1283, QgsUnitTypes.LayoutMillimeters))
layout.addLayoutItem(img)
# move to exact position
img.attemptMove(QgsLayoutPoint(282.721, 4.89614, QgsUnitTypes.LayoutMillimeters))

layout = manager.layoutByName(layoutName)
exporter = QgsLayoutExporter(layout)
# This final section of code creates a final PDF document of the map.
# fn should equal your current working folder, or a desired folder you wish to store the final PDF output of the map.
fn = "C:\\Users\\Jakeb\\OneDrive\\Desktop\\Graduate Research- Desktop\\NOAA Automated Mapping Project\\NOAA Final Maps\\EX 1806\\Divetest.pdf"
#exporter.exportToImage(fn, QgsLayoutExporter.ImageExportSettings())
exporter.exportToPdf(fn, QgsLayoutExporter.PdfExportSettings())

```

Figure A.3 *Map production script.py continued*



## APPENDIX B

### ROV AUTOMATED MAPPING STANDARD OPERATING PROCEDURES

## Standard Operating Procedure #1

### Extracting annotation records and ROV data from SeaTube

Latest Revision: August 26, 2021 by J. Freeman

Purpose: This SOP describes how to download the ROV video annotation and vehicle data from SeaTube. These data are necessary to generate substrate maps for an ROV dive.

- 1) Open a web browser and navigate to <https://data.oceannetworks.ca/SeaTube>. *Note: SeaTube is an interactive online interface that allows users to view and annotate underwater video data. The SeaTube interface is used to record expert video annotations for NOAA OER ROV dives.*
- 2) When the SeaTube website is loaded, you will notice a list of various research cruises located on the left side of the screen under the tab "videos".
- 3) From this list, navigate to the research cruise that you wish to download annotation data for.
- 4) If you expand the selected cruise, you will see individual dives listed. If you click on one of the dives, a video of that dive will be displayed as well as a map of the dive and a list of time stamped video annotations.
- 5) At the top of the screen there is a set of menus (Preview/Data Search/Plotting Utility/Sea Tube/More). Click on "More" then "Annotations" then "Annotation Search" to bring up a window that lets you select specific dives for which to export annotations and vehicle data.

Resource Type: Dive

Resource: EX1803-Dive01 2018-04-12 01:00:00.0 G

☒ Include annotations up topology tree

Date From (UTC): dd-MMM-yyyy hh:mm:ss

Date To (UTC): dd-MMM-yyyy hh:mm:ss

► Fields

► Owner

► Origin

Search Export... Save Search

Figure B.1 ROV Automated Mapping SOP #1

This SOP describes how to download the ROV video annotation and vehicle data from *SeaTube*. These data are necessary to generate substrate maps for an ROV dive.

- 6) To do this, under "Resource Type" select "Dive" and under "Resource" select the dive of interest that you want to export. The export button will prompt you to select Excel format or .csv. For this procedure, .csv should be selected.
- 7) Once the desired .csv files are downloaded, store them on a local directory in folders named by the research cruise/expedition the data were collected on:

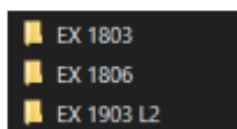


Figure B.1      SOP #1 Continued

## Standard Operating Procedure #2

### Extracting 1HZ data from the Ocean Exploration Digital Atlas

Purpose: The purpose of this standard operating procedure is to describe to the user how to effectively download 1HZ data from NOAA's OER digital atlas.

Although the extraction of the ROV 1Hz data file is unnecessary for the classification of ROV videography, it is useful for the cartographic representation of the entire dive path. The ROV file with navigation data sampled at 1 Hz was downloaded from the NOAA OER Digital Atlas for each dive conducted on all three expeditions. The OER Digital Atlas is designed to provide the public with access to data that has been previously collected during OER expeditions. Within the ROV summary products, multiple different datasets containing a compilation of dive-related data, products, documents, and images are available (i.e., ROV Data Access by Dive, Event Logs, Cruise Video Collection Self-Service Portal, Collected Specimens, and Submersible Navigation/Sensor Data (ASCII)) (National Oceanic and Atmospheric Administration – Office of Ocean Exploration and Research, 2015). The 1Hz file contains ROV navigation and attitude data recorded every second for all dives including the latitude and longitude coordinates of the ROV. These data allow the dive path of the ROV to be mapped with a high degree of spatiotemporal resolution. This contrasts with the dive annotation file extracted from ONC's Seatube V2 which only contains the latitude and longitude at which an annotation was recorded during the dive, which generally occurred at a much lower frequency than 1 Hz. If only the dive annotation file was extracted for the automated mapping of the seafloor the only visible dive path would thus be the path the ROV Deep Discoverer followed while recording real-time dive annotations, and not the entire dive path from start to finish. When integrated, both the annotation and 1Hz datasets will allow the entire ROV dive path to be mapped and the position of annotations to be plotted.

- 1) Open a web browser and navigate to <https://www.ncei.noaa.gov/maps/oer-digital-atlas/mapsOE.htm>. This Digital Atlas is an online map portal that allows the public access to data that is collected during OER expeditions.
- 2) When the digital atlas is loaded, you will notice a search box located on the left side of the screen.

Figure B.2 ROV Automated Mapping SOP #2

This SOP describes how to download 1Hz ROV vehicle data from NOAA's OER Digital Atlas. ROV navigation data recorded with frequency of 1 Hz are necessary to generate substrate maps for an ROV dive.

Search Layers Help

Use any combination of the provided dropdown lists to define search criteria

Year:

- ☐ 2021
- ☐ 2020
- ☐ 2019
- ☐ 2018
- ☐ 2017

Ocean Places:

All

Mission Groups:

All

Platforms:

All

Enter Search Text:

Search

© 2017 Creative Commons

- 3) This search box is where we will navigate to the different research expeditions, we wish to download 1HZ data from.
- 4) For this project we will need to access 1HZ data for (EX1803 / EX1806 / EX1903L2). To access all three of these expeditions, we must check both 2018 and 2019 in the “Year” box.

Year:

- ☐ 2021
- ☐ 2020
- ☒ 2019
- ☒ 2018
- ☐ 2017

- 5) When the correct expedition year(s) are selected we must then select “Ocean Places”, “Mission Groups”, and “Platforms”. For this project “Ocean Places” will remain “All”, “Mission Groups” will be set to “Okeanos ROV Cruises”, and “Platforms” will remain “All”.

Figure B.2 SOP #2 Continued

Search Layers Help

Use any combination of the provided dropdown lists to define search criteria

Year:

☐ 2021  
☐ 2020  
☒ 2019  
☒ 2018  
☐ 2017

Ocean Places:  
All

Mission Groups:  
Okeanos ROV Cruises

Platforms:  
All

Enter Search Text:

Search

0 Cruises Displayed

- 6) When all parameters inside the search box are selected click "Search". The map frame will then load the OER expeditions we selected.



- 7) When the map is loaded, we can then select the expedition we want to download data from. Each expedition is labeled with a point on the map. By clicking on the point, you will see a box appear showing you the details of the expedition you have chosen.

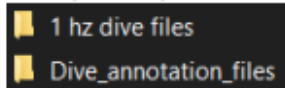
Figure B.2 SOP #2 Continued

- 8) Navigate to "ROV Data Access" and click on it. You will then see a variety of options under "ROV Summary Products", directly underneath click on "ROV Data Access by Dive".
- 9) When a "ROV Data Access by Dive" is selected a new page will then appear on your screen. This page will allow you to select any dive that was conducted during the research expedition you selected from the map frame.
- 10) Select the dive you want to download 1HZ data for and scroll to the bottom of the screen. You will see a list of downloadable files at the bottom of the screen.

#### Download & View Files

Dive Summary Report (PDF - 921 KB)	<a href="#">View/Download</a>
Dive Track (KML - 67 KB)	<a href="#">View/Download</a>
ROV Ancillary Data (Zip - 3.74 MB)	<a href="#">Download</a>
ROV CTD/Sensor Data (Zip - 5.85 MB)	<a href="#">Download</a>
Camera Sled CTD/Sensor Data (Zip - 16.8 MB)	<a href="#">Download</a>
Low-Resolution Video Clips (Zip - 10.9 GB)	<a href="#">Download</a>
Underwater Still Images (Zip - 80.3 MB)	<a href="#">Download</a>
Dive Video Collection Self-Service Portal	<a href="#">Open</a>

- 11) The data we want to download is "ROV Ancillary Data". This data will be downloaded through a zipped folder. Once the data is unzipped it should be stored within the same working folder that you stored the annotation data that we previously downloaded from SeaTube.



- 12) Like the dive annotation files folder, each dive within the 1HZ data folder should be organized by each dive in a separate folder.



Figure B.2 SOP #2 Continued

## Standard Operating Procedure #3

### Python Script that revises substrate annotation to fine resolution CMECS classes.

**Purpose:** The purpose of this python script is to assign certain CMECS classification values based on substrate annotations that are stored in the .csv annotation file we downloaded from SeaTube.

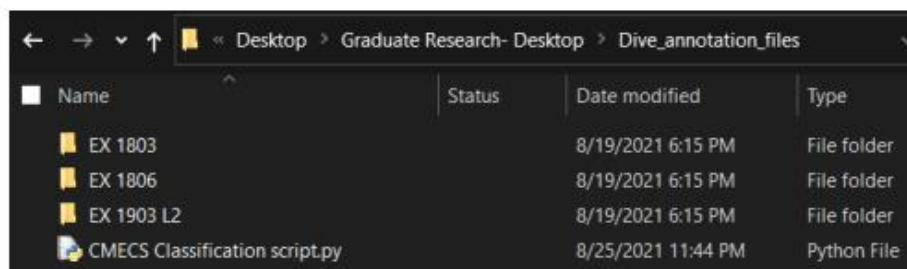
This process was done through a for loop within the Python code, a for loop is generally used for the iteration of a sequence. For – example, for each substrate annotation to be reclassified within each of the ROV dive annotation files the script must continuously run until all substrate annotations within the file have been successfully reclassified into the CMECS-compliant format. The script then reclassified the row heading into a constant of “Substrate”. Once this was done the CMECS classification script then accurately located the start index of “Primary” within the string and pulls the entire classification string (i.e., Primary Unconsolidated Secondary Unconsolidated). The entire original string was then split into a list, and “Primary” was then located within the new list and the next list value (i.e., Unconsolidated) was located. This process was then repeated for “Secondary” and anything that remains after the secondary classification was therefore deleted. For – example the annotation (Unconsolidated: Sloping (5 to <30 Degrees) simply became (Unconsolidated) and was then assigned to the secondary class variable. An empty string was then generated and the new compliant CMECS classification was then constructed. To effectively reclassify each substrate annotation to the correct compliant CMECS classification an (if/elif/else) was used. This statement allowed the Python script to effectively check for multiple expressions. If the condition for it is “False” it will then move on to the next condition that is located in the following “elif” block. If all the conditions are false the else condition will then be executed. For – example, if the primary portion of the substrate annotation is equal to “unconsolidated” then it will be converted to “Fine”. However, if the primary classification string is not equal to “unconsolidated” the script will then moon onto the next “Elif” block until it finds a correct match. In this case if the primary string is not reclassified the end “else” statement returns “Primary: Not able to Classify” thus letting the user know that the primary substrate classification string was not reclassified. This allowed the CMECS classification script to be efficient for all substrate units within the CMECS schema. The same process was then carried out for the secondary classification portion of the substrate string. When both the primary and secondary portions of the substrate string were created the script then wrote a new string to the “Substrate” row found within the ROV dive annotation file. This allowed the CMECS classification script to be efficient for all substrate units within the CMECS schema.

- 1) The CMECS classification script should be stored in the same folder where the dive annotations are stored.

### Figure B.3 ROV Automated Mapping SOP #3

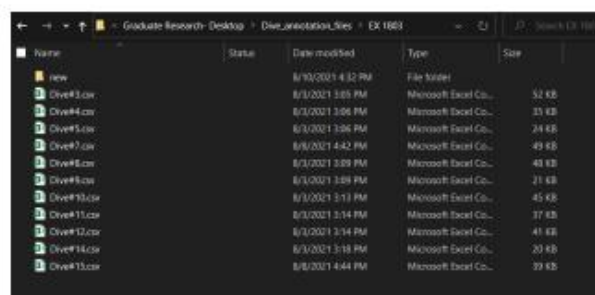
This SOP describes the Python script CMECS classification script.py that converts seafloor substrate annotations recorded in a legacy format into a new format compliant with the current CMECS standard.





- 2) Open the "CMECS classification script".
- 3) "file" is where you will input the pathway to the annotation .csv we want to assign the CMECS classes to.
- 4) "outfile" is where you will assign the new annotation .csv to be stored, once the reclassification is complete.  

```
# this is where the input and output annotation files are entered. The outfile must be set in a new file folder within the current working folder.
file = ("C:\\Users\\jandb\\OneDrive\\Desktop\\Graduate Research- Desktop\\Dive_annotation_files\\EX 1806\\Dive1806.csv")
outfile = ("C:\\Users\\jandb\\OneDrive\\Desktop\\Graduate Research- Desktop\\Dive_annotation_files\\EX 1806\\new\\Dive1806_new.csv")
```
- 5) The "outfile" must be located within the current annotation dive folder. For example, the image below is the EX1803 dive annotation folder. The "new" folder that we want the reclassified .csv annotation files to be stored in, must be located inside of the current working folder, or in this case inside the EX1803 annotation folder.



- 6) As you reclassify each annotation .csv, the new .csv with the correct CMECS substrate classifications will be stored in the "new" file.

Figure B.3 SOP #3 Continued

7) Example of ROV annotation .csv before ran through the "CMECS Classification Script":

[illegible]

8) Example of ROV annotation .csv after being ran through the “CMECS Classification Script”:

Figure B.3      SOP #3 Continued



## **Standard Operating Procedure #4**

### **Generation of dive path and viewshed shapefiles in QGIS**

Latest Revision: August 26, 2021 by J. Freeman

Purpose: This SOP describes the process of using the automated python script *Mapping Script.py* to generate digital shapefiles of the ROV dive path (line feature) and imaged as well as classified seafloor viewsheds (polygon features).

- 1) Download and install QGIS (<https://qgis.org/en/site/forusers/download.html>). QGIS is open-source desktop geographic information system software that works on most common computing platforms/ operating systems.
- 2) Open QGIS.
- 3) Create a new project and name it "ROV dive mapping program".
- 4) At the top of the screen, click on "Plugins"
- 5) Click on "Python Console".
- 6) Within the python console in the top left corner, you will notice five different icons:



- 7) Click on notepad icon. This is the third icon from the left.
- 8) Once the python editor is open, select the yellow file icon in the top left corner of the python editor. This will allow you to open *Mapping Script.py*, which is saved in your working folder.



- 9) Each module of the script and its purpose is described below.

---

Figure B.4 ROV Automated Mapping SOP #4

This SOP describes the process of using the automated Python script *Mapping Script.py* to generate digital shapefiles of the ROV dive path (line feature) and imaged as well as classified seafloor viewsheds (polygon features).

#### 10) "Create points layer from table" tool:

##### Code:

- result = processing.run("qgis:createpointslayerfromtable",
- {'INPUT': "C:\\Users\\Jakeb\\OneDrive\\Desktop\\Graduate Research- Desktop\\1 hz dive files\\EX 1803 Dives\\EX1803\_DIVE03\_20180416\\DIVE031Hz.csv",
- 'XFIELD': 'LON\_DD',
- 'YFIELD': 'LAT\_DD',
- 'ZFIELD': None,
- 'TARGET\_CRS': QgsCoordinateReferenceSystem("EPSG:4326"),
- 'OUTPUT': "C:\\Users\\Jakeb\\OneDrive\\Desktop\\Graduate Research- Desktop\\NOAA Shapefiles\\path\_dive\_EX1803#03.shp"})
- onehzpath = result['OUTPUT']

11) "INPUT" of this algorithm will always remain a 1HZ data file from the ROV in .csv format. The directory of this input should be set to whichever dive expedition you are currently wanting to map. For example, in the previous step you can see I have the input set to a folder where 1HZ data for dive expedition EX1803 is found.

12) Each time you want to generate a new map for the dive expedition you are working on, simply change the number, and date of dive within the directory. For example:

- "C:\\Users\\Jakeb\\OneDrive\\Desktop\\Graduate Research- Desktop\\1 hz dive files\\EX 1803 Dives\\EX1803\_DIVE03\_20180416\\DIVE031Hz.csv"
- As you generate a map for each dive within a certain expedition, the dive number and date will need to change within the directory.
- This dive was conducted on April 16, 2018. The date format within the directory is "EX\_1803\_DIVE03\_20180416" the next dive "Dive #4" was conducted on April 17, 2018, the format would then change to "EX1803\_DIVE03\_20180417".
- In the last part of the directory, you will find the actual .csv that is being called by the "Create points layer from table" tool "DIVE031HZ", as you change the date format for each dive, the 1HZ .csv for that dive will have to be set as well. For example, if you are wanting to map dive 4 from EX1803 you will change "DIVE031HZ" to "DIVE041HZ". The names of the .csv files can be changed by the user before they are entered into the program. This may make the process more efficient depending on the user's preference.

#### 13) "Points to path" tool:

- This points to path algorithm creates a dive path from 1HZ dives points that were generated from the previous "create points layer from table".

Figure B.4 SOP #4 Continued

A line feature and resultant shapefile were created with the “points to path” tool from the 1Hz dive points that were generated by the method described in the section above titled “create points layer”. The “points to path” function performed a simple conversion on the newly generated 1Hz points shapefile. This was done by the “ORDER\_FIELD” parameter within the function being set to the “DATE” column found within the ROV 1Hz dataset. Thus, ordering the generation of a line shapefile (dive path) to be drawn in a start-to-finish sequence. The “points to path” tool was only applied to the points created from the ROV 1Hz dataset. It was not applied to the ROV dive annotation dataset since the annotation file only contained navigational data from the ROV Deep Discoverer where seafloor annotations were recorded during each dive.

```
Code:
result = processing.run('qgis:pointstopath',
{'INPUT': onchzpath,
'ORDER_FIELD': 'DATE',
'GROUP_FIELD': None,
'DATE_FORMAT': ''},
'OUTPUT', "C:\\Users\\Jakeb\\OneDrive\\Desktop\\Graduate
Research- Desktop\\NOAA Shapefiles\\Dive Path.shp")
divepath = result['OUTPUT']
```

- “INPUT” of this algorithm will always remain as the output parameter that is set in the previous algorithm. (Example: onchzpath).
- “ORDER\_FIELD” will remain “DATE”.
- “GROUP\_FIELD” will remain “None”.
- “DATE\_FORMAT” will remain (“”).
- “OUTPUT” should be the working folder of the user, or a desired folder for the output shapefile to be stored. For the dive path to be generated correctly the name of the shapefile should remain “Dive Path.shp”.
- divepath will always remain [“OUTPUT”].

**Code:**

```
result = processing.run("qgis:pointstopath",
{'INPUT': onchzpath,
'ORDER_FIELD': 'DATE',
'GROUP_FIELD': None,
'DATE_FORMAT': ''},
'OUTPUT', "C:\\Users\\Jakeb\\OneDrive\\Desktop\\Graduate Research- Desktop\\NOAA
Shapefiles\\Dive Path.shp")
divepath = result['OUTPUT']
```

**14) “Create points layer from table”: Annotation .csv**

This was done with the QGIS “Create points layer from table” function, which creates a points layer from an external ASCII (.csv) table. The “points layer from table” function in QGIS was used twice throughout the automated substrate map

Figure B.4 SOP #4 Continued



generation process. The tool was first executed with the 1Hz dataset to generate a set of points based on the exact location of the ROV every second of the dive. The "Input" field of this function always remained the ROV 1Hz dataset that was to be mapped in the form of a point shapefile. The function then used the latitude and longitude of the ROV 1Hz dataset to obtain the correct spatial location of the 1Hz point shapefiles. The "XFIELD" remained longitude and the "YFIELD" remained latitude to obtain an accurate geospatial location. For this analysis, the "TARGET\_CRS" was set to a constant of (EPSG: 4326). The output folder for the resulting 1Hz point shapefile was set to the current working folder or the desired folder for the output shapefile to be stored for further analysis. The process was then repeated with the ROV annotation dataset, thus creating a separate set of points representing the exact location where each substrate annotation was recorded. Similarly, to the 1Hz dataset, the "XFIELD" was set to "Environment – Longitude" and the "YFIELD" was set to "Environment – Latitude". Although a point shapefile was generated from the 1Hz dataset, it is not portrayed cartographically in the final dive map but was necessary for the creation of the ROV "Dive Path" line shapefile.

#### Code:

```
Code:
result = processing.run("qgis:createpointslayerfromtable",

{'INPUT': "C:\\Users\\Jakeb\\OneDrive\\Desktop\\Graduate
Research- Desktop\\1 hz dive files\\EX 1803
Dives\\EX1803_DIVE03_20180416\\DIVE031Hz.csv",
'XFIELD': 'LON_DD',
'YFIELD': 'LAT_DD',
'ZFIELD': None,
'TARGET_CRS': QgsCoordinateReferenceSystem('EPSG:4326'),
'OUTPUT': "C:\\Users\\Jakeb\\OneDrive\\Desktop\\Graduate
Research- Desktop\\NOAA
Shapefiles\\path_dive_EX1803#03.shp"})
onehzpath = result['OUTPUT']
```

The above portion of the Mapping script.py is designed to create for the generation of navigational points based on the ROV 1Hz dataset.

Figure B.4 SOP #4 Continued

```

Code:
result = processing.run("qgis:createpointslayerfromtable",
{ 'INPUT': "C:\\Users\\Jakeb\\OneDrive\\Desktop\\Graduate
Research- Desktop\\Dive_annotation_files\\EX
1803\\new\\Dive#3_new.csv",
'XFIELD': 'Environment - Longitude',
'YFIELD': 'Environment - Latitude',
'ZFIELD': None,
'TARGET_CRS': QgsCoordinateReferenceSystem('EPSG:4326'),
'OUTPUT': "C:\\Users\\Jakeb\\OneDrive\\Desktop\\Graduate
Research- Desktop\\Graduate Research\\NOAA
Shapefiles\\annotation_points_EX1803_Dive#3.shp"})
annotation = result['OUTPUT']

```

The above portion of the Mapping script.py is designed to create for the generation of annotation points based on the ROV annotation dataset.

```

- result = processing.run("qgis:createpointslayerfromtable",
- {'INPUT': "C:\\Users\\Jakeb\\OneDrive\\Desktop\\Graduate Research-
- Desktop\\Dive_annotation_files\\EX 1803\\new\\Dive#3_new.csv",
- 'XFIELD': 'Environment - Longitude',
- 'YFIELD': 'Environment - Latitude',
- 'ZFIELD': None,
- 'TARGET_CRS': QgsCoordinateReferenceSystem('EPSG:4326'),
- 'OUTPUT': "C:\\Users\\Jakeb\\OneDrive\\Desktop\\Graduate Research-
- Desktop\\Graduate Research\\NOAA
- Shapefiles\\annotation_points_EX1803_Dive#3.shp"})
- annotation = result['OUTPUT']

```

15) As we discussed in step 3, “INPUT” of this algorithm will always remain an annotation data file from the ROV in .csv format. The directory of this input should be set to whichever dive expedition you are currently wanting to map. For example, in step 3 you can see I have the input set to a folder where 1HZ data for dive expedition EX1803 is found. However, for this “Create points layer from table” we will be using an annotation .csv instead of a 1HZ .csv.

16) Each time you want to generate a new map for the dive expedition you are working on, simply change the number, and date of dive within the directory. For example, in step 3 the input is set to:

Figure B.4 SOP #4 Continued



- "C:\Users\Jakeb\OneDrive\Desktop\Graduate Research- Desktop\1 hz dive files\EX 1803 Dives\EX1803\_DIVE03\_20180416\DIVE031Hz.csv"

17) For this "create points layer from table" we will set the directory to:

- {"INPUT": "C:\Users\Jakeb\OneDrive\Desktop\Graduate Research- Desktop\Dive\_annotation\_files\EX 1803\new\Dive#3\_new.csv"

18) This "INPUT" will be set to pull the new annotation .csv that was reclassified in the previous SOP.

- "INPUT" of this algorithm will always remain a Dive annotation data file from the ROV in csv format, that is acquired from SeaTube. The Input data file must be .csv that was reclassified within the CMECS reclassification python script.
- "XFIELD" will remain 'Environment - Longitude' if the user does not make manual corrections to the annotation csv file.
- "YFIELD" will remain 'Environment - Latitude' if the user does not make manual corrections to the annotation csv file.
- "ZFIELD" will remain "None".
- "TARGET\_CRS" will remain "QgsCoordinateReferenceSystem('EPSG:4326')".
- "OUTPUT" should be the working folder of the user, or a desired folder for the output shapefile to be stored.
- "annotation" will always remain "[OUTPUT]".

19) **Wedge Buffer tool:** This algorithm will create wedge buffers showing the viewing direction of the ROV as well as substrate type.

This manual process is done through the ability to change the (WIDTH, OUTER\_RADIUS, and INNER\_RADIUS) parameters that are found within the Python wedge buffer function. Respectively, for the analysis and classification of the substrate along the seafloor the viewing distance and viewing width were set as those in Ruby (2017) to reduce the potential of overlapping substrate features found within the video data. The viewshed polygons created by the wedge buffer tool are assigned coordinates and a substrate attribute based on the points generated in the "create points layer from table" from the ROV annotation dataset. Therefore, the only attribute assigned directly to the wedge buffer is the ROV heading, which is the same value as the camera heading. Because the ROV remains approximately level throughout the dive, the ROV roll, and pitch attributes were not accounted for in the creation of the viewsheds polygons (Ruby, 2017). The boundary between overlapping viewshed polygons with common substrate classifications is joined to make a single polygon with the "Dissolve" function. This results in a single polygon representing extended portions of the seafloor with the same substrate classification. This is useful for maintaining clarity of the produced substrate maps because an abundance of

Figure B.4 SOP #4 Continued

overlapping classifications within the final ROV substrate map would create confusion when conducting further analyses regarding what the forward-facing camera was viewing at that exact annotation coordinate.

```
Code:
result = processing.run("native:wedgebuffers",
{'INPUT': annotation,
'AZIMUTH':
QgsProperty.fromExpression("attribute('Environm_1')"),
'WIDTH': 44,
'OUTER_RADIUS': 0.0005,
'INNER_RADIUS': 0,
'OUTPUT': "C:\\Users\\Jakeb\\OneDrive\\Desktop\\Graduate
Research- Desktop\\Graduate Research\\NOAA
Shapefiles\\wedge_buffers_EX1903_dive#3.shp"})
wedgebuffer = result['OUTPUT']
```

- "INPUT" will remain "annotation", this allows the wedges to be placed where observations were made along the seafloor.
- "AZIMUTH" will always remain "QgsProperty.fromExpression("attribute('Environm\_1')")".
- "WIDTH" is currently set to "44". This parameter can be changed each time the code is ran if needed.
- "OUTER\_RADIUS" the output of this parameter is generated best if the radius is set to "0.0005".
- "INNER\_RADIUS" should remain "0".
- "OUTPUT" should be the working folder of the user, or a desired folder for the output shapefile to be stored. The end of the Directory needs to change each time a new dive is being mapped within an expedition. This will create a new shapefile each time a new dive is mapped.
- "wedgebuffer" will always remain "[OUTPUT]".

```
Code:
result = processing.run("native:wedgebuffers",
{'INPUT': annotation,
'AZIMUTH': QgsProperty.fromExpression("attribute('Environm_1')"),
'WIDTH': 44,
'OUTER_RADIUS': 0.0005,
'INNER_RADIUS': 0,
'OUTPUT': "C:\\Users\\Jakeb\\OneDrive\\Desktop\\Graduate Research-
Desktop\\Graduate Research\\NOAA Shapefiles\\wedge_buffers_EX1903_dive#3.shp"})
wedgebuffer = result['OUTPUT']
```

**20) Dissolve tool:** This algorithm pulls the output of the wedgebuffer and takes common substrate observations and dissolves boundaries between them.

Figure B.4 SOP #4 Continued

```

Code:
result = processing.run('qgis:dissolve',
{'INPUT': wedgebuffer ,
'DISSOLVE_ALL': None,
'FIELD': 'Substrate',
'OUTPUT': "C:\\Users\\Jakeb\\OneDrive\\Desktop\\Graduate
Research- Desktop\\NOAA Shapefiles\\CMECS Substrate
Units.shp"})
dissolvebuffer = result ['OUTPUT']

```

The above image represents an example of the dissolve tool function that is found in the Mapping script.py Python code.

- "INPUT" will always remain "wedgebuffer".
- "DISSOLVE\_ALL" will always remain "None".
- "FIELD" will always remain 'Substrate'
- "OUTPUT" should be the working folder of the user, or a desired folder for the output shapefile to be stored. The output name must remain "CMECS Substrate Units. Shp" for the map to be properly displayed.
- "dissolvebuffer" will always remain "= result ['OUTPUT']".

```

Code:
result = processing.run('qgis:dissolve',
{'INPUT': wedgebuffer ,
'DISSOLVE_ALL': None,
'FIELD': 'Substrate',
'OUTPUT': "C:\\Users\\Jakeb\\OneDrive\\Desktop\\Graduate Research- Desktop\\NOAA
Shapefiles\\CMECS Substrate Units.shp"})
dissolvebuffer = result ['OUTPUT']

```

**21) Loading and classifying substrate shapefiles into QGIS:** This section of the program loads the substrate shapefiles and classifies the shapefiles previously generated.

- "qmlfile" sets the exact color and symbology for the dissolve buffer. This should always remain the same. The substrate qml file should be stored in the same working folder as all other shapefiles and scripts used to generate the maps.
- "fileInfo" will always remain "QFileInfo(dissolvebuffer)".
- "basename" will always remain "fileInfo.baseName()".
- "rlayer" will always remain "QgsVectorLayer(dissolvebuffer, baseName)".
- The if statement and the parameters found within it should all remain the same.

**Code:**

Figure B.4 SOP #4 Continued

```

qmlfile = "C:\\Users\\Jakeb\\OneDrive\\Desktop\\Graduate Research-
Desktop\\NOAA qml files\\Substrate type by colorv2.qml"
fileInfo = QFileInfo(dissolvebuffer)
baseName = fileInfo.baseName()
rlayer = QgsVectorLayer(dissolvebuffer, baseName)
if rlayer.isValid():
    print('Layer loaded!')
QgsProject.instance().addMapLayer(rlayer)
processing_run("native:setlayerstyle", {'INPUT': dissolvebuffer, 'STYLE': qmlfile})
##returns nothing.....
extent = rlayer.extent()
rlayer.triggerRepaint()

```

**22) Loading and classifying the dive path: This section of the code loads and classifies the dive path previously generated, into QGIS.**

- “qmlfile” sets the exact color and symbology for the dive path. This should always remain the same. The dive path qml file should be stored in the same working folder as all other shapefiles and scripts used to generate the maps.
- “fileInfo” will always remain “QFileInfo(divepath)”.
- “basename” will always remain “fileInfo.baseName()”.
- “rlayer” will always remain “QgsVectorLayer(divepath, baseName)”.
- The if statement and the parameters found within it should all remain the same.

**Code:**

```

qmlfile= "C:\\Users\\Jakeb\\OneDrive\\Desktop\\Graduate Research-
Desktop\\NOAA qml files\\divepath.qml"
fileInfo = QFileInfo(divepath)
basename = fileInfo.baseName()
rlayer = QgsVectorLayer(divepath, basename)
if rlayer.isValid():
    print('Layer loaded!')
QgsProject.instance().addMapLayer(rlayer)
processing_run("native:setlayerstyle", {'INPUT': divepath, 'STYLE': qmlfile})
extent = rlayer.extent()
rlayer.triggerRepaint()

```

- 23) If the script is running correctly, you will notice two “Layer Loaded” icons show up on the left panel in the python editor. This lets you know both the line shapefile, and substrate shapefiles have successfully loaded in the QGIS canvas.

Figure B.4 SOP #4 Continued

```

Python Console
1 Python Console
2 Use iface to access QGIS API interface or Type help(iface) for more info
3 Security warning: typing commands from an untrusted source can harm your computer
4 >>> exec(open('C:/Users/Jakeb/OneDrive/Desktop/Graduate Research- Desktop/NOAA Python
  scripts/Final_Mapping_ScriptEX1806.py'.encode('utf-8')).read())
5 Layer loaded!
6 Layer loaded!
7

```

24) The QGIS map canvas should show both the “Dive Path” and “CMECS Substrate Units” shapefiles.



Figure B.4      SOP #4 Continued



## Standard Operating Procedure #5

### Python Script Creating Final .pdf output of Dive Map

Purpose: The purpose of this SOP is to walk the user through the process of creating a Final .pdf output by using the “Final Map Production” script.

A portion of the Mapping Script.py code, which is presented in Appendix A, carries out this processing step to generate the substrate viewshed polygons with a uniform color for every ROV dive. This is done with a Qt Modeling Language (QML) file that was constructed specifically for each substrate class in the CMECS classification schema. The “qmlfile” parameter within the “dissolve” portion of the final Mapping script always remained the same. Once all shapefiles’ layers were created and loaded into the QGIS map extent, the dissolve function then used the qml file to color the substrate with a uniform color. This was done so that every map contained the same substrate symbology style (Color). The qml file is very effective for automated mapping due to its ability to classify multiple shapefile outputs with the same cartographic and symbolic representations depicted in the final ROV substrate map. In this case, a qml file was used to assign both the substrate (viewshed) and dive path (line) shapefiles with uniform color for the ROV substrate final map. The ROV dive path was assigned a uniform color (black) within the qml file, although this was not completely necessary, one constant dive path color for every final substrate map avoids the confusion of the reader and provides the final map with a standard and organized cartographic representation. Once all integrated QGIS tools and functions have successfully been executed in the Mapping script.py the final automated script Map production script.py is used to create a final ROV substrate map in .pdf format. This script performs a variety of tasks to effectively create an organized final ROV substrate map. Cartographic features (i.e., title, legend, scalebar, and map text) can all be manually adjusted and edited (i.e., font, size, location, text) within the Map production script.py. For every final ROV substrate map a “Line, Ticks Up” scale bar assigned to use the unit meters was assigned to each map. The location of the scale bar and legend for each individual was assigned an exact location within the map layout, thus allowing every final map product to look the same in the sense of cartographic feature representation. However, the size of the scale bar would vary at times due to some ROV dive paths being shorter than others. The scale bar and legend, along with the title and added text can all be manually moved throughout the layout extent. However, for the production of the final ROV substrate maps the location of these cartographic features remained constant.

- 1) After successfully running the “ROV Dive Mapping” python script, open the “Final Map Production” script in the python plugin in QGIS.
- 2) Unlike the “ROV Dive Mapping” script, for the “Final Map Production” script does not require any input directories to be set.

Figure B.5 ROV Automated Mapping SOP #5

This SOP describes the process of creating a .pdf format output map with Map production script.py

### 3) Setting the Layers that are to be mapped:

- This section of code allows you to enter the map layers you want to project in the layout.
- For this automated map, we want both the Dive Path, and CMECS Substrate Units to be mapped.
- Both the "Dive Path" and "CMECS Substrate Units" layers should always be called for them to be mapped in the final layout.

#### Code Example:

```
layers = QgsProject.instance().mapLayersByName('Dive Path')
layer = layers[0]
layers2 = QgsProject.instance().mapLayersByName('CMECS Substrate Units')
layer2 = layers2[0]
project = QgsProject.instance()
manager = project.layoutManager()
```

### 4) Creation of map layout:

- "layoutName" allows you to assign a name to the map layout that will be created in the current map canvas. For this layout some simple such as "Dive Map Layout" will work. This parameter does not have to remain "Dive Map Layout" it can be renamed each time a new map is produced.

#### Code Example:

```
layoutName = 'Dive Map layout'
```

### 5) Editing the map Title:

- Each time a new ROV dive map is created, the title of the dive can manually be edited in the "Title" section of the script.

#### Code example:

```
title.setText("EX 1903 L2 Dive 19")
```

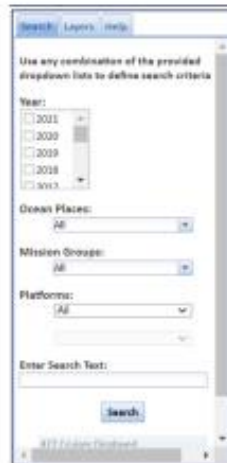
### 6) Editing the "Date", "Max Depth", and "Bottom Time" of the ROV dive:

- In this portion of the script, you can edit the date that the dive was conducted.
- Each time a new automated map is created, the date of the dive must be manually entered.
- The date that each dive is conducted can be accessed by opening your web browser and navigating to (<https://www.ncei.noaa.gov/maps/oer-digital-atlas/mapsOE.htm>.)

Figure B.5 SOP #5 Continued

This Digital Atlas is an online map portal that allows the public access to data that is collected during OER expeditions.

- When the digital atlas is loaded, you will notice a search box located on the left side of the screen.

A screenshot of a web-based search interface for a digital atlas. The interface is titled "Search" and includes links for "Layers" and "Help". Below the title, it says "Use any combination of the provided dropdown data to define search criteria". There are three dropdown menus: "Year:" with options from 2001 to 2013, "Ocean Places:" with "All" selected, "Mission Groups:" with "All" selected, and "Platforms:" with "All" selected. Below these is a text input field labeled "Enter Search Text:" and a "Search" button. At the bottom, it says "© 2013 OER Expeditions".

- This search box is where we will navigate to the different research expeditions, we wish to access.
- When the correct expedition year(s) are selected we must then select "Ocean Places", "Mission Groups", and "Platforms". For this project "Ocean Places" will remain "All", "Mission Groups" will be set to "Okeanos ROV Cruises", and "Platforms" will remain "All".

Figure B.5 SOP #5 Continued



- When all parameters inside the search box are selected click "Search". The map frame will then load the OER expeditions we selected.



- When the map is loaded, we can then select the expedition we want to access the "Date", "Max Depth" and "Bottom Time" from. Each expedition is labeled with a

Figure B.5 SOP #5 Continued

point on the map. By clicking on the point, you will see a box appear showing you the details of the expedition you have chosen.

- Navigate to "ROV Data Access" and click on it. You will then see a variety of options under "ROV Summary Products", directly underneath click on "ROV Data Access by Dive".
  - When a "ROV Data Access by Dive" is selected a new page will then appear on your screen. This page will allow you to select any dive that was conducted during the research expedition you selected from the map frame.
  - Select the dive you are currently mapping.
  - When you have selected the dive, you are currently mapping, you will see an "Overview" section.
  - In this overview section you will find "Date", "Max Depth", and "Bottom Time".
  - Once you have updated the "Date", "Max Depth", and "Bottom Time" for the automated map, you are ready to set the name for the final map .pdf, as well as what folder you wish to store your final map in.
  - **Code Example for "Date":**  
`map_label.setText("Date: -----July 11, 2019")`
  - **Code Example for "Max Depth":**  
`map_label.setText("Max Depth: 1623 Meters")`
  - **Code Example for "Bottom Time":**  
`map_label.setText("Bottom Time: 5 Hours 58 Minutes 00 Seconds")`
- 7) **Creating the final .pdf map and output folder:**
- Navigate to the current folder you have been saving previous data and files in for this project.
  - Create a new folder titled "Final Maps".
  - Refer to the last section of the "Final Map Production" script and set the "fn" directory to the new "Final Maps" folder.
  - At the very end of the directory is where you will change the name of each final .pdf map that is created.

**Final Output Directory Example:**

Figure B.5      SOP #5 Continued

```

# This final section of code creates a final PDF document of the map.
# fn should equal your current working folder, or a desired folder you wish to store the final PDF output of the map.
fn = "C:\\Users\\Jahab\\OneDrive\\Desktop\\Graduate Research- Desktop\\NOAA Final Maps\\EX 1903 L2\\Divebest.pdf"
#exporter.exportImage(fn, qgsLayoutExporter.mapExportSettings())
#exporter.exportToPdf(fn, qgsLayoutExporter.pdfExportSettings())

```

#### 8) Check the output folder for final map:

- Navigate to the folder you that you directed your final map to be saved to.

#### 9) Example of Final Map:

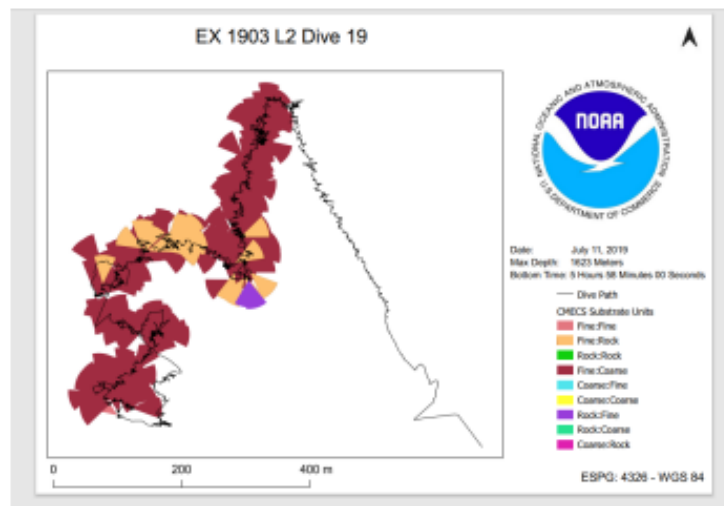


Figure B.5 SOP #5 Continued

APPENDIX C

FINAL ROV SUBSTRATE MAPS

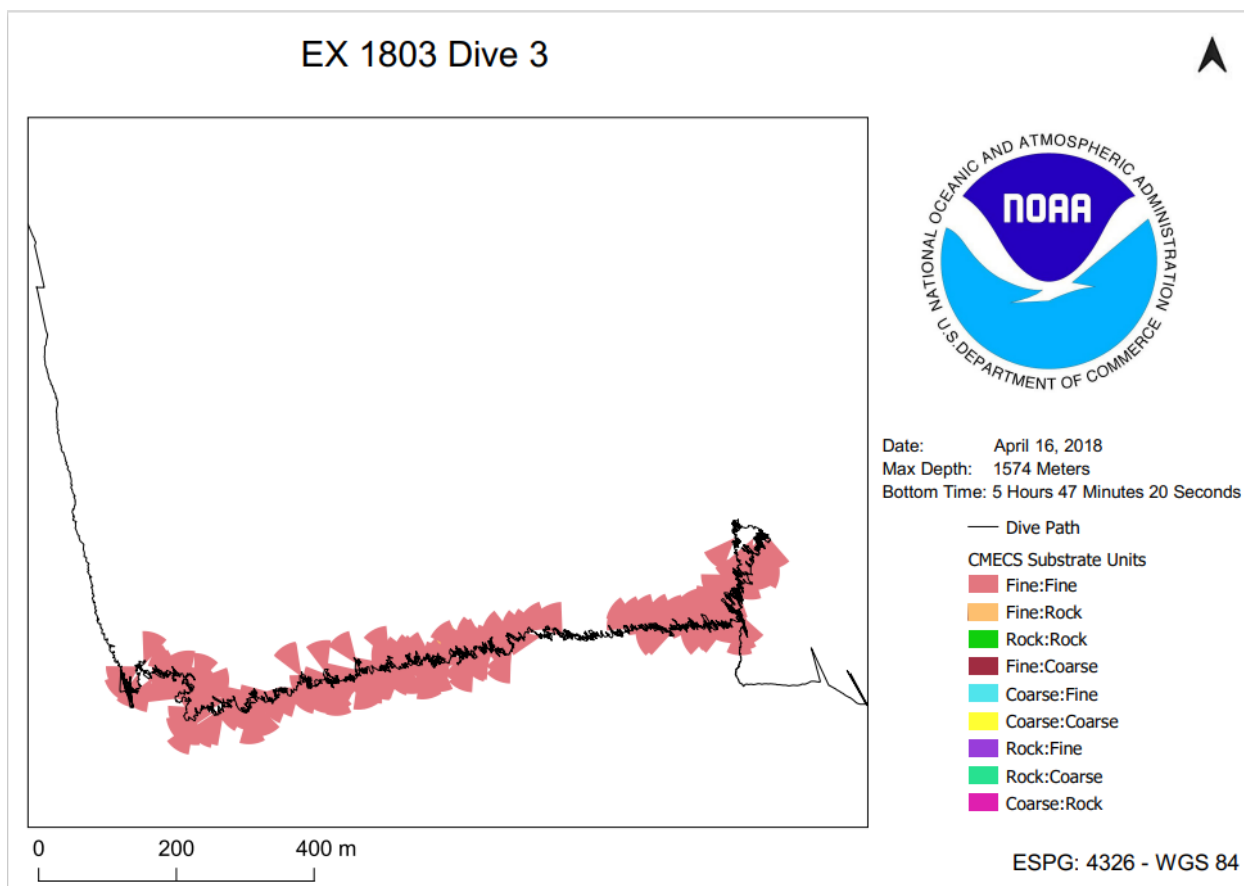


Figure C.1 Classification of substrate EX 1803 Dive 03

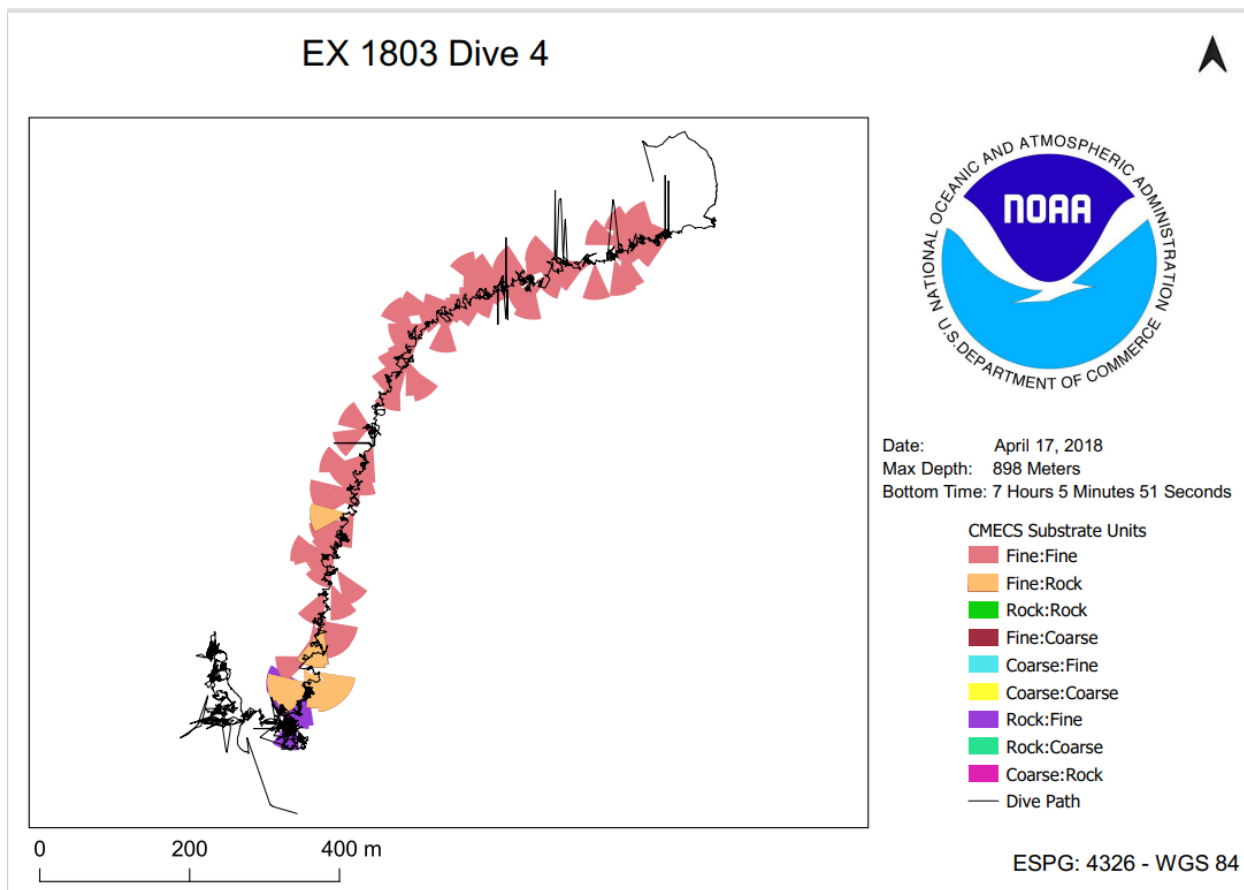


Figure C.2 Classification of substrate EX 1803 Dive 04

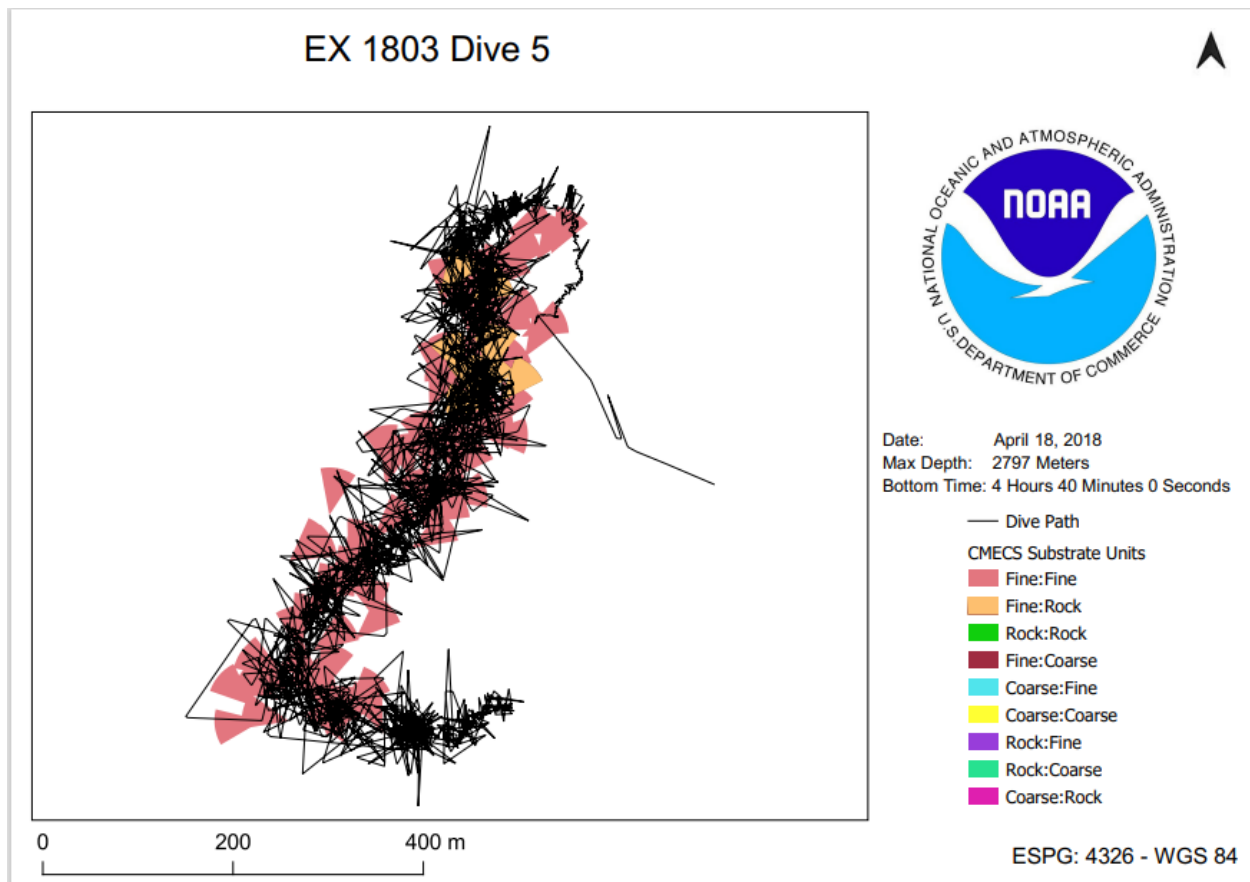


Figure C.3 Classification of substrate EX 1803 Dive 05

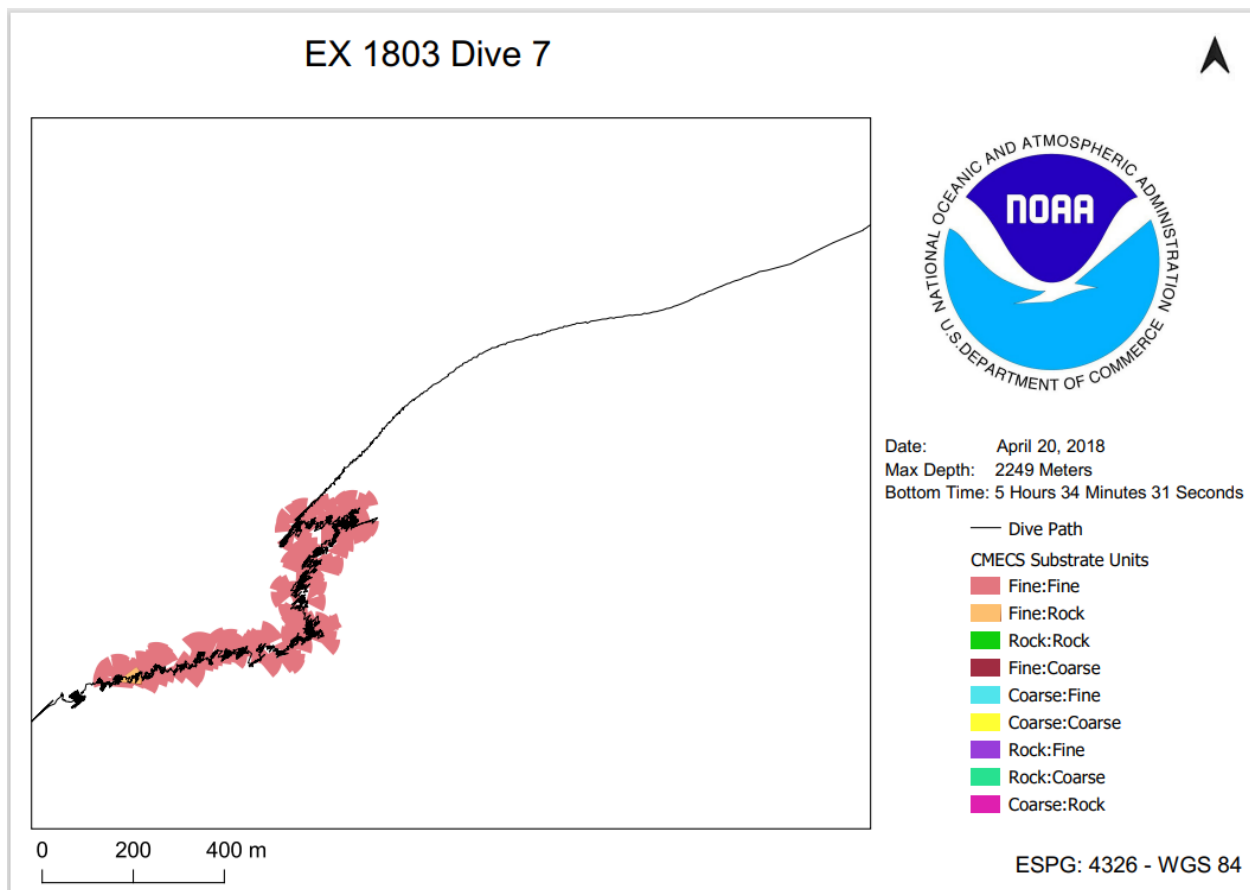


Figure C.4 Classification of substrate EX 1803 Dive 07



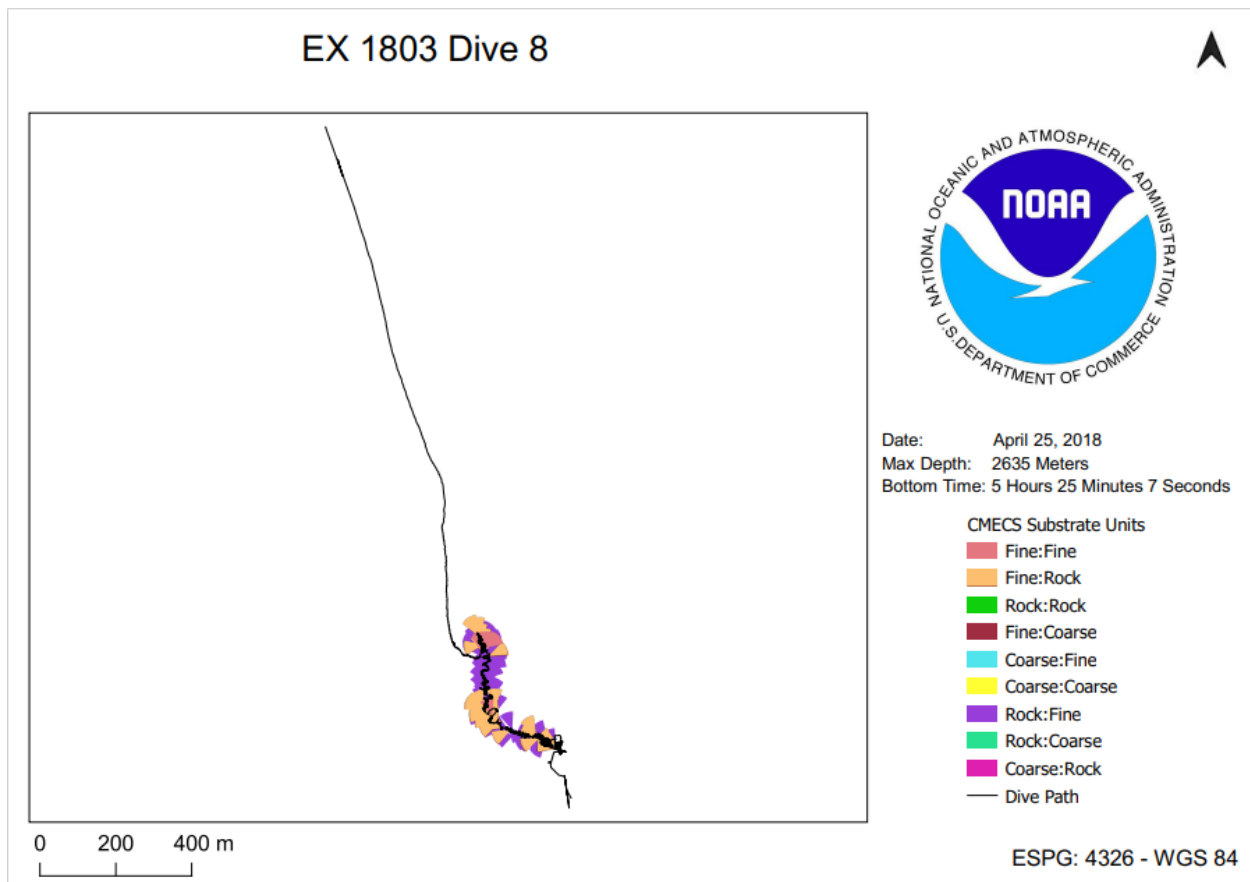


Figure C.5 Classification of substrate EX 1803 Dive 08

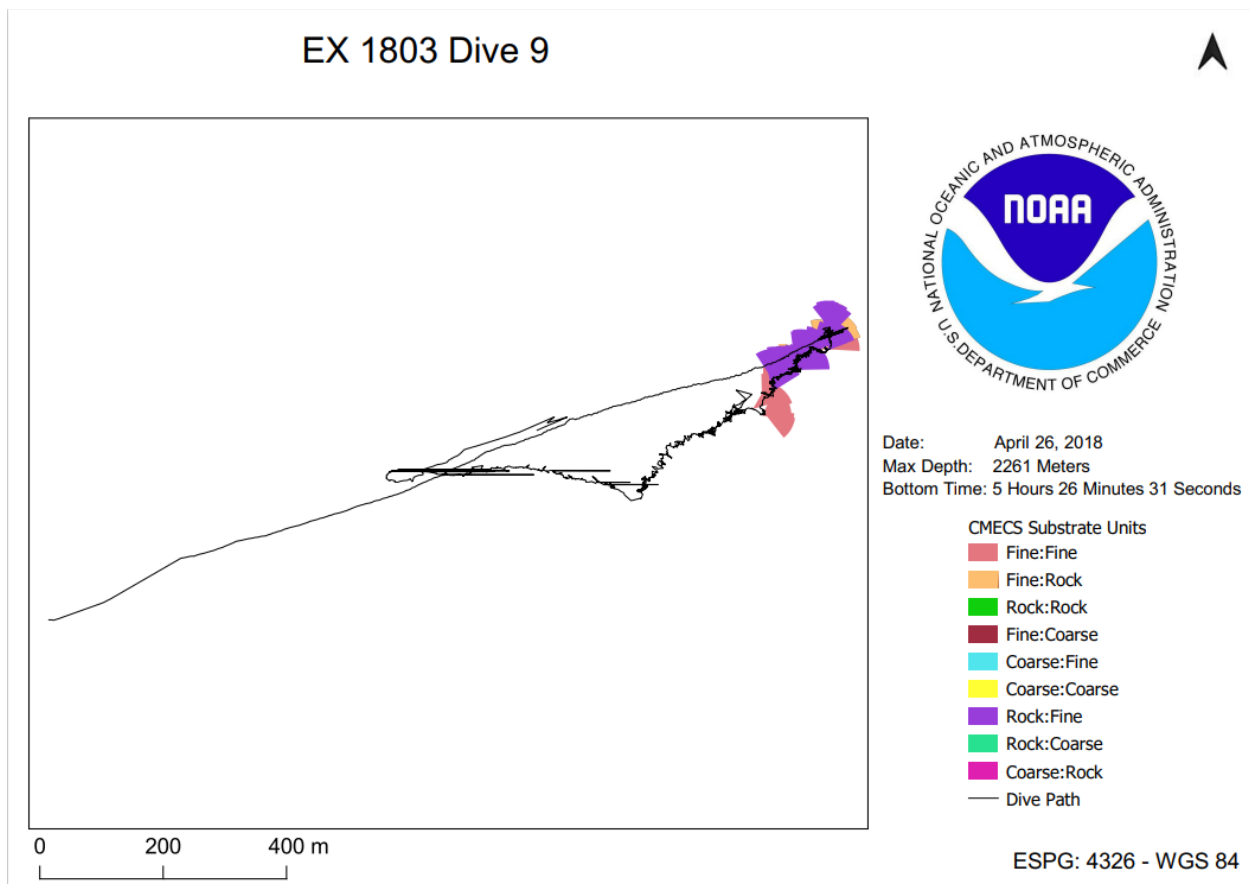


Figure C.6 Classification of substrate EX 1803 Dive 09

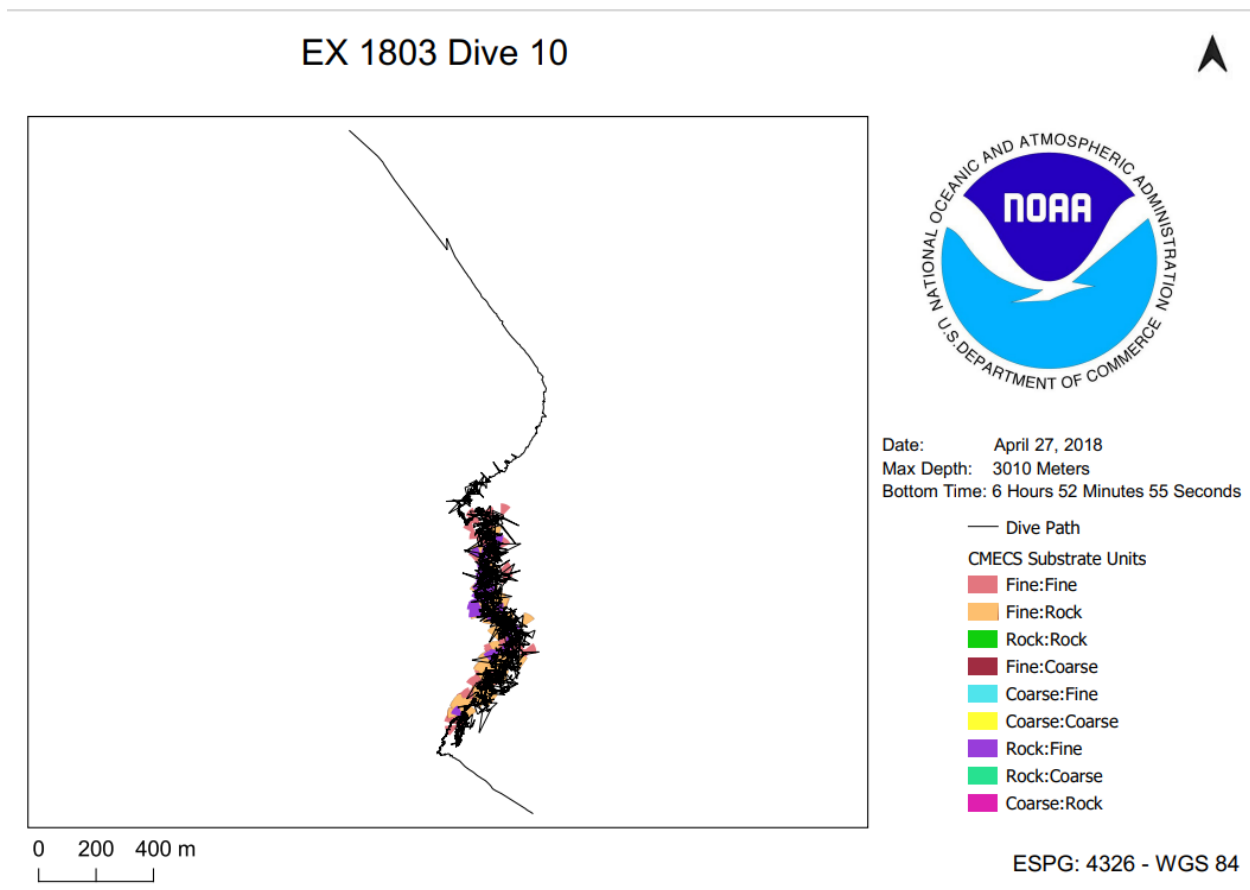


Figure C.7 Classification of substrate EX 1803 Dive 10

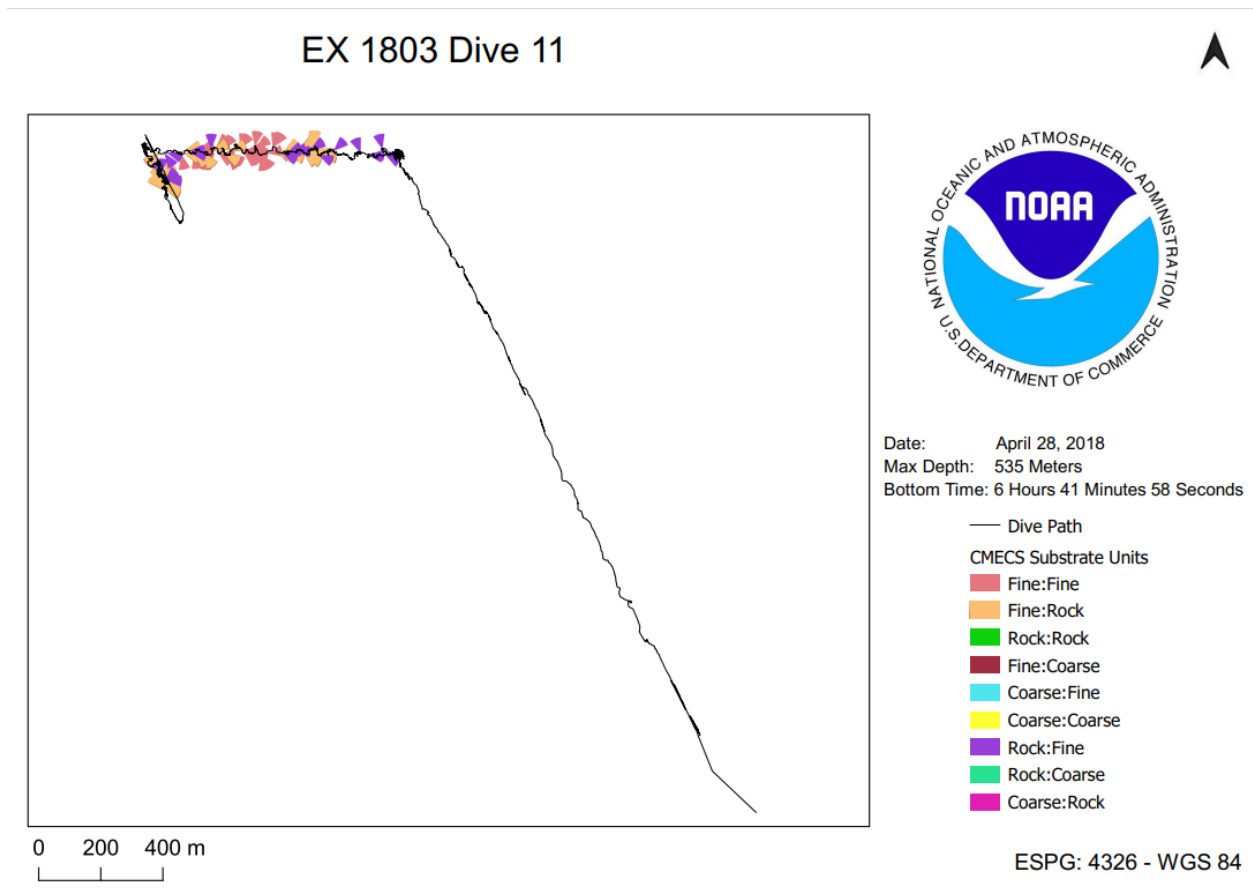


Figure C.8 Classification of substrate EX 1803 Dive 11

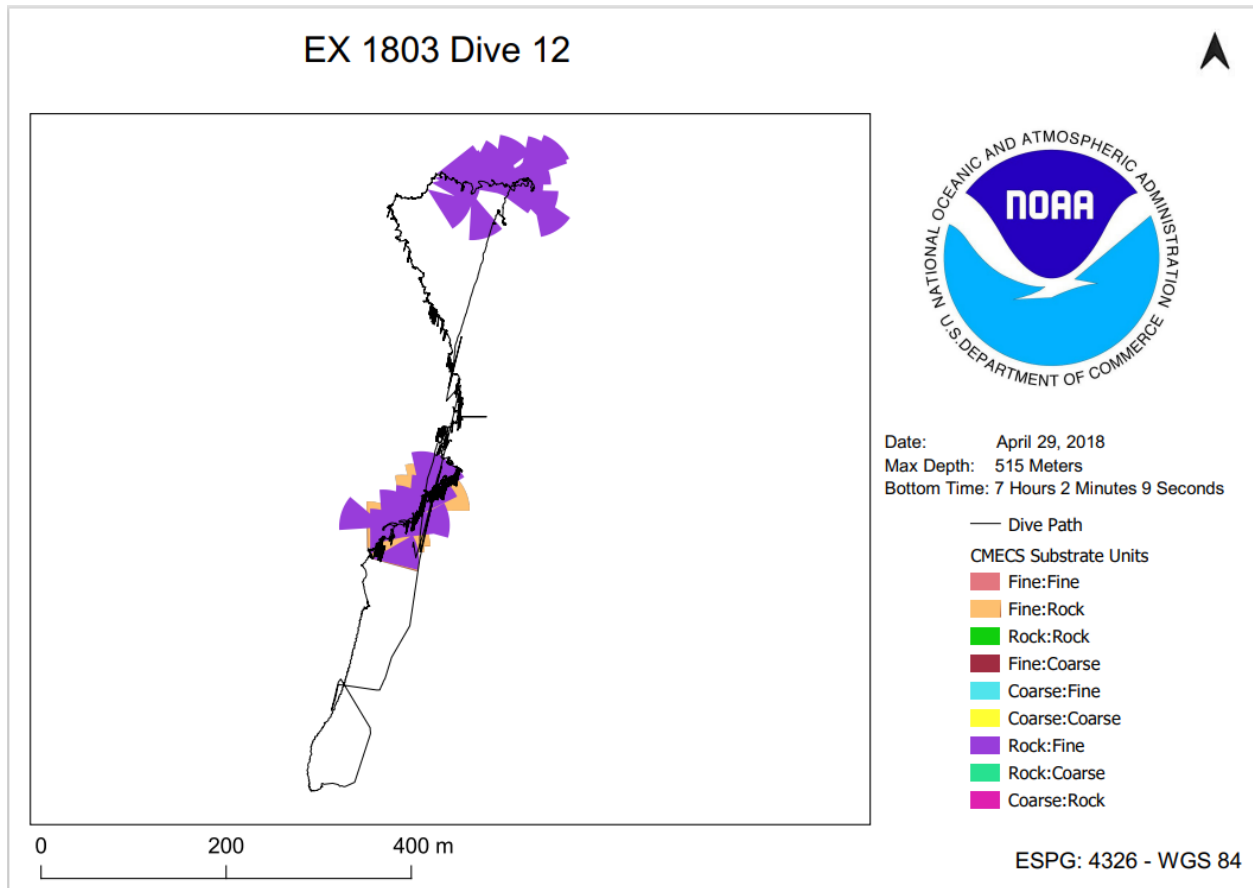


Figure C.9 Classification of substrate EX 1803 Dive 12

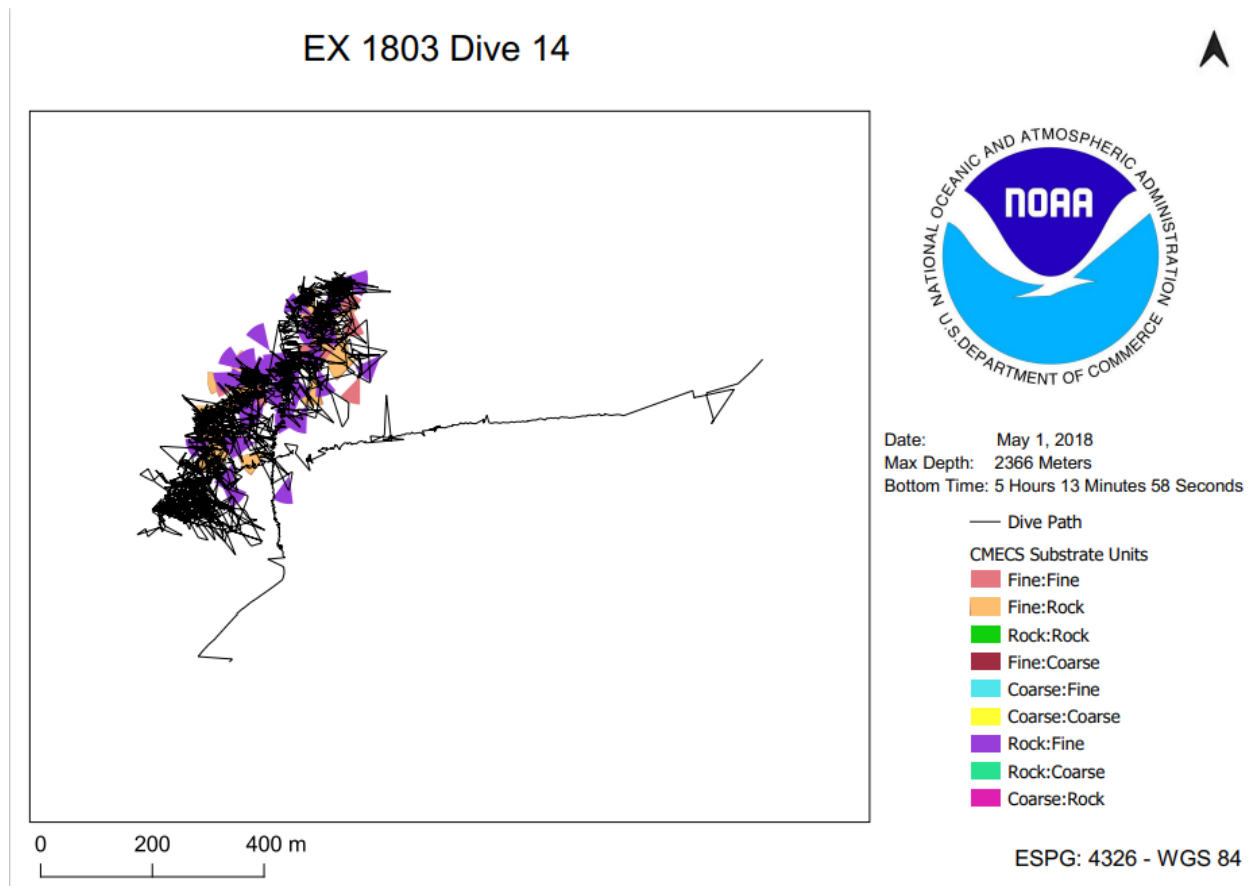


Figure C.10 Classification of substrate EX 1803 Dive 14

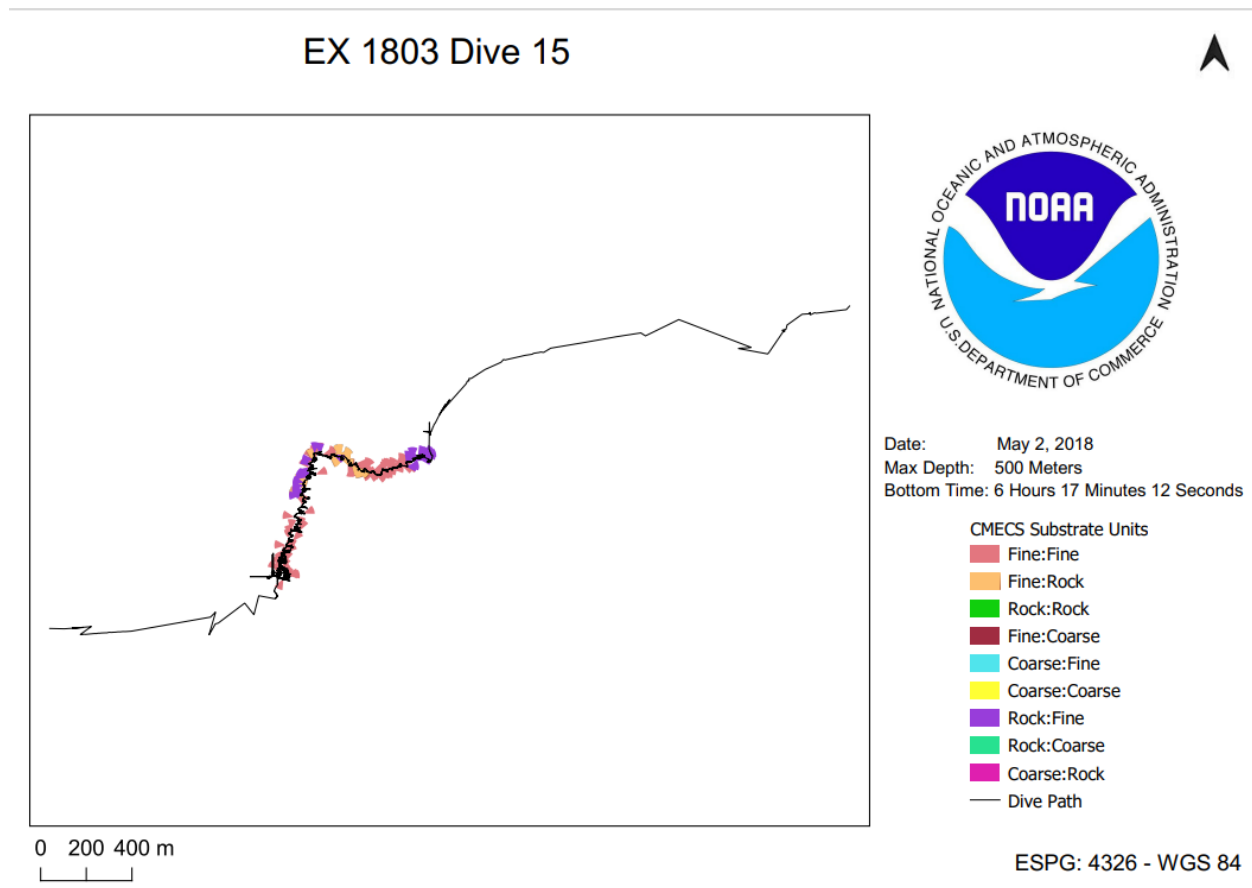


Figure C.11 Classification of substrate EX 1803 Dive 15

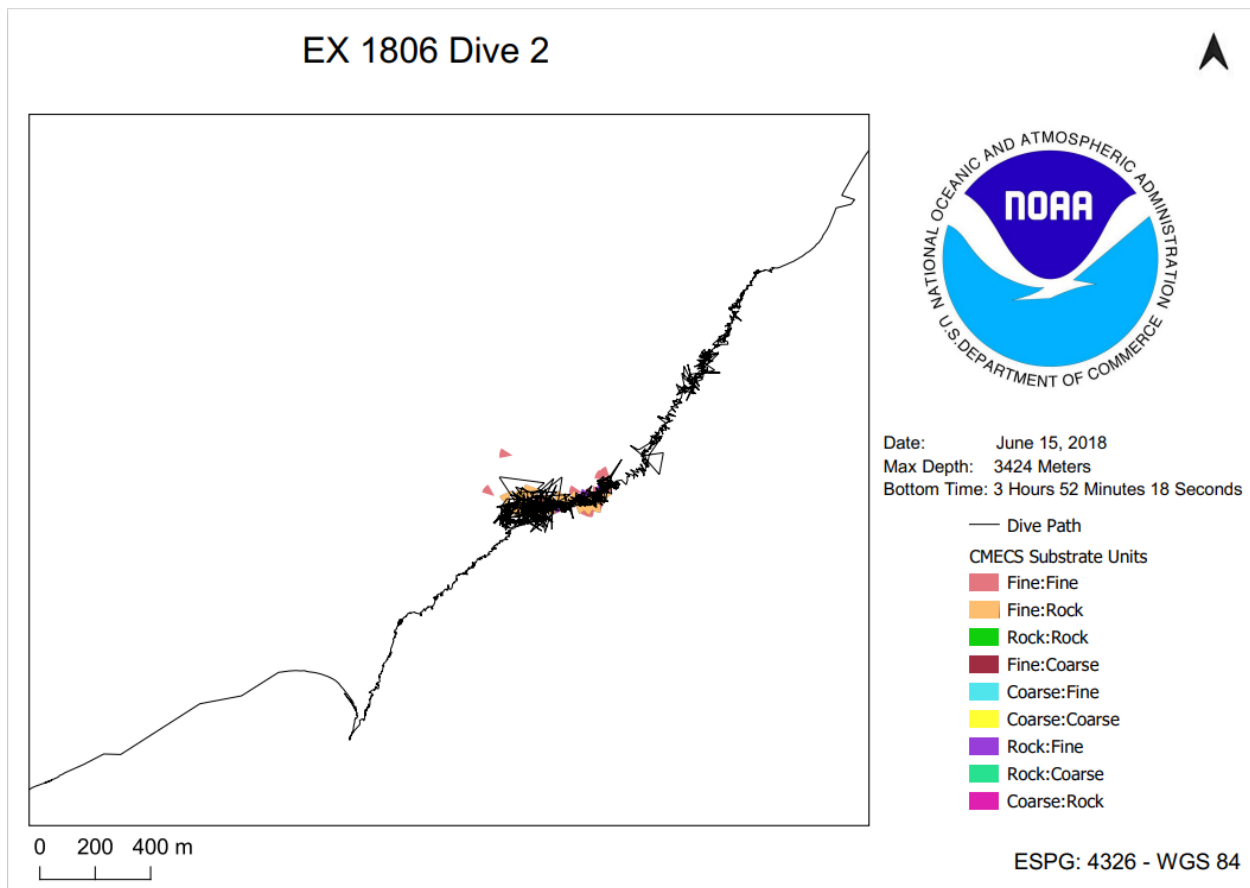


Figure C.12 Classification of substrate EX 1806 Dive 02



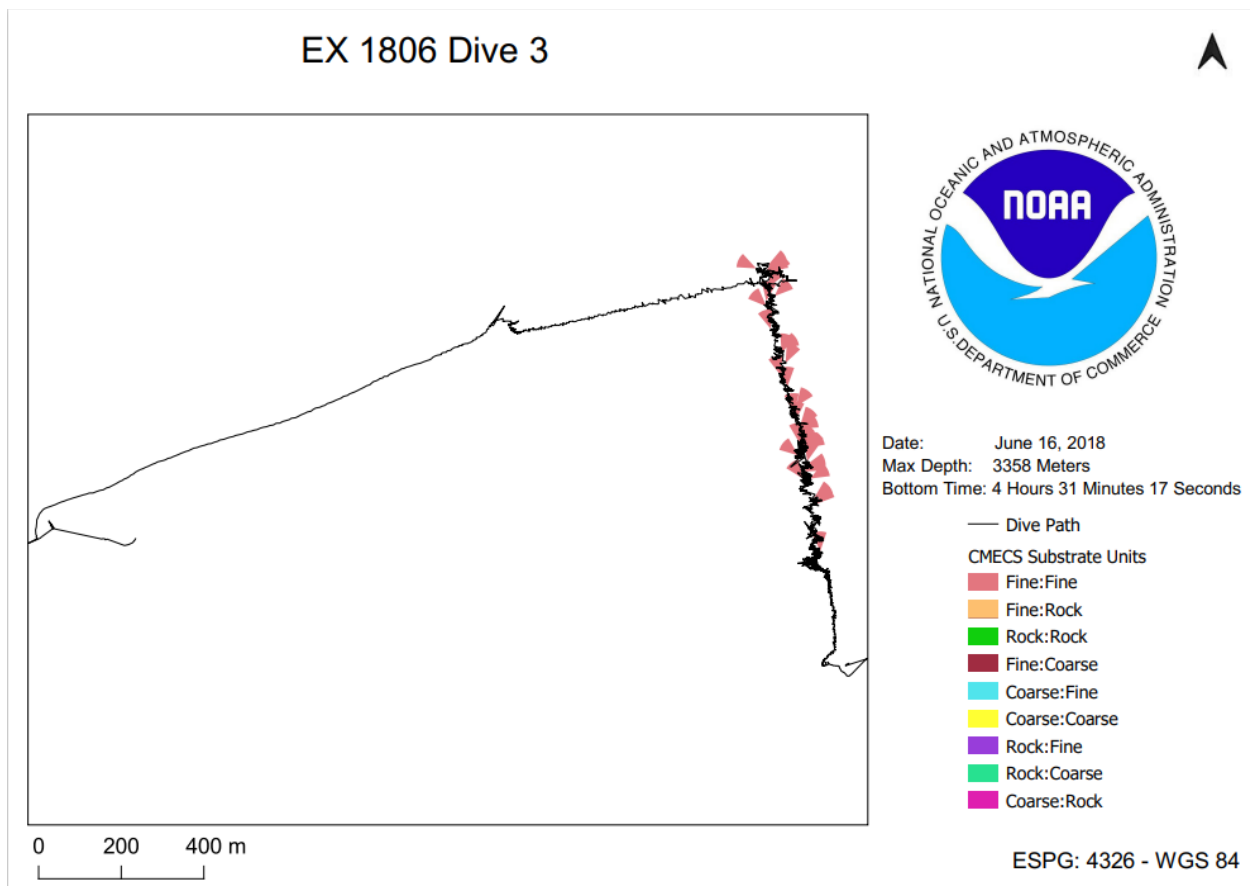


Figure C.13 Classification of substrate EX 1806 Dive 03

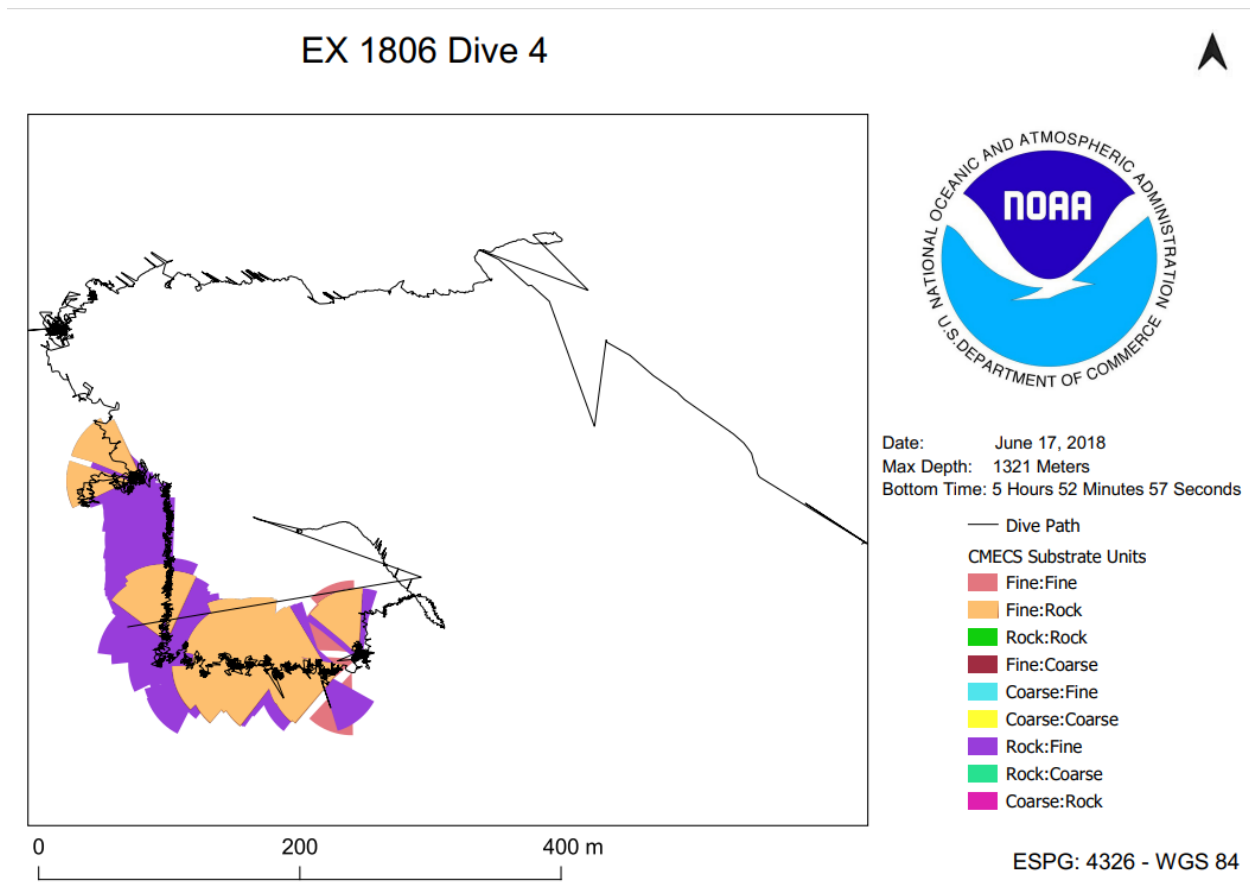


Figure C.14 Classification of substrate EX 1806 Dive 04

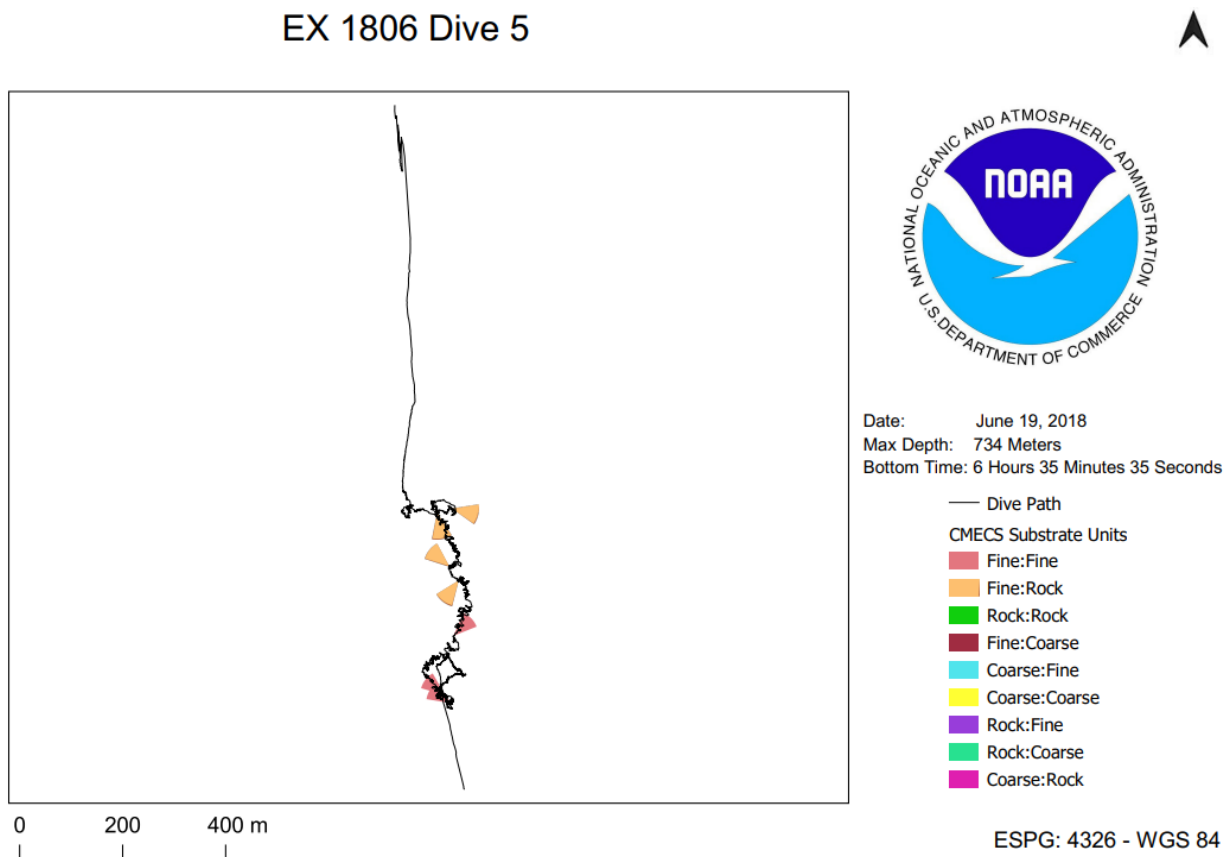


Figure C.15 Classification of substrate EX 1806 Dive 05

## EX 1806 Dive 6

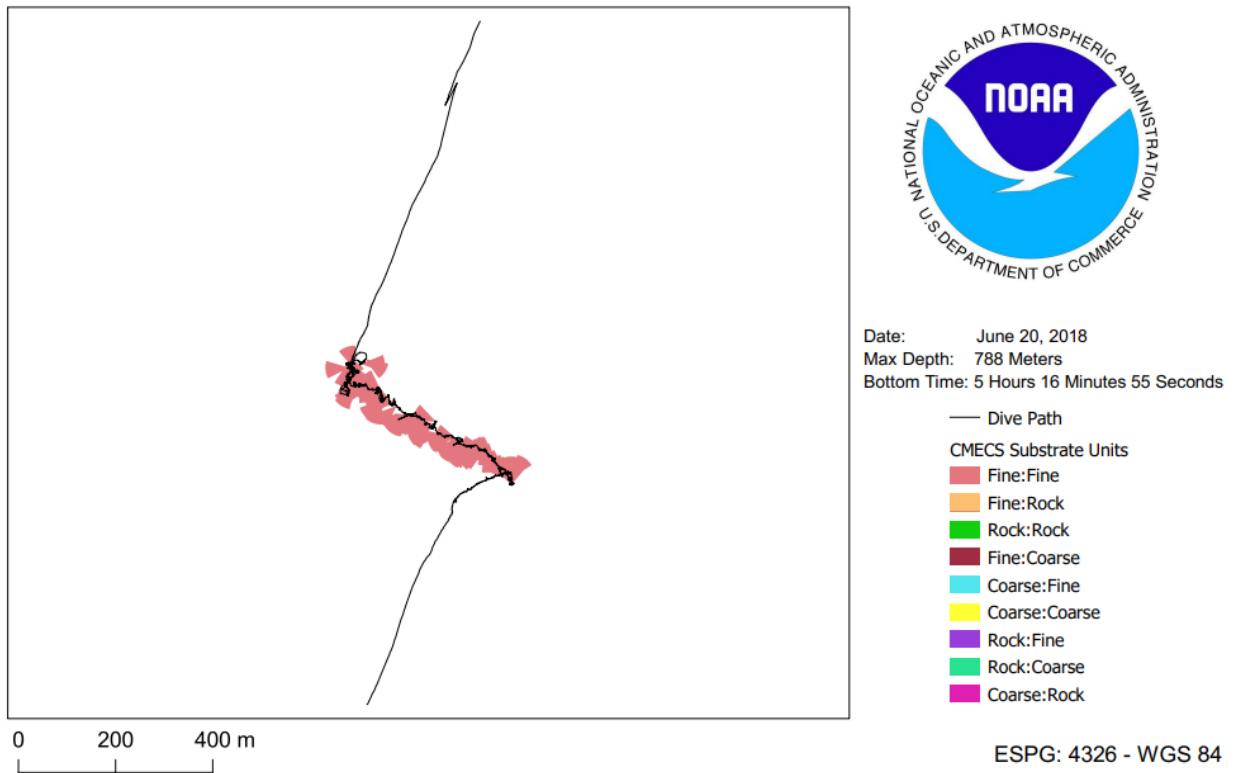


Figure C.16 Classification of substrate EX 1806 Dive 06

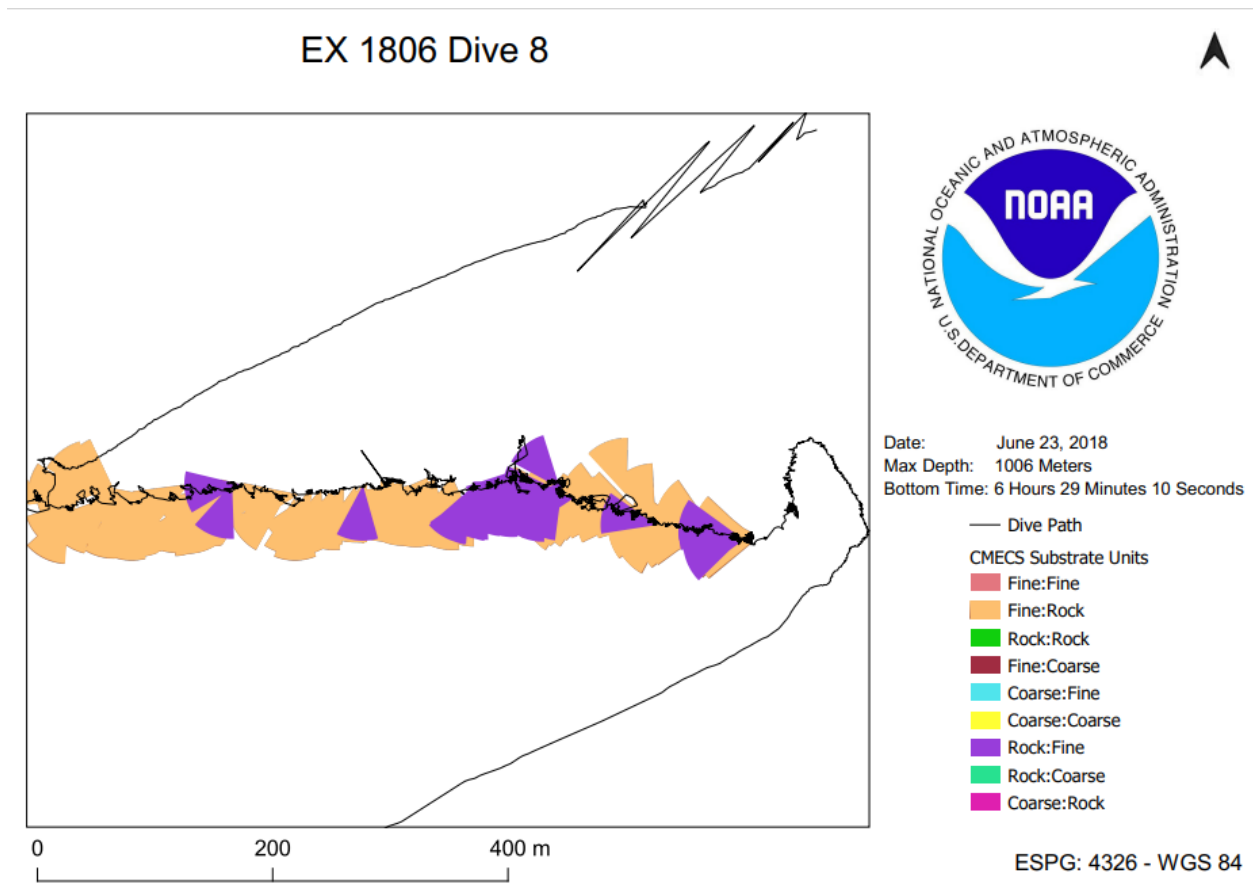


Figure C.17 Classification of substrate EX 1806 Dive 08

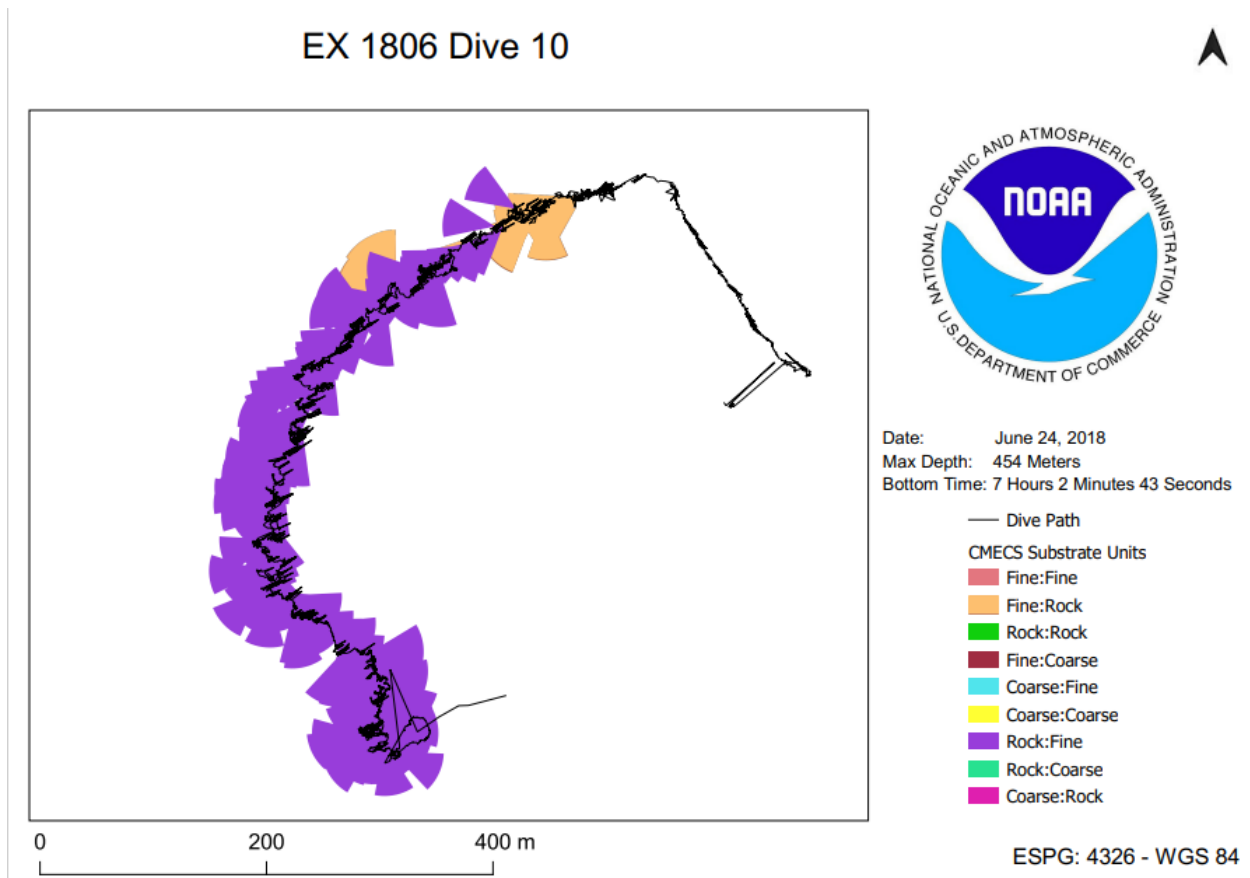
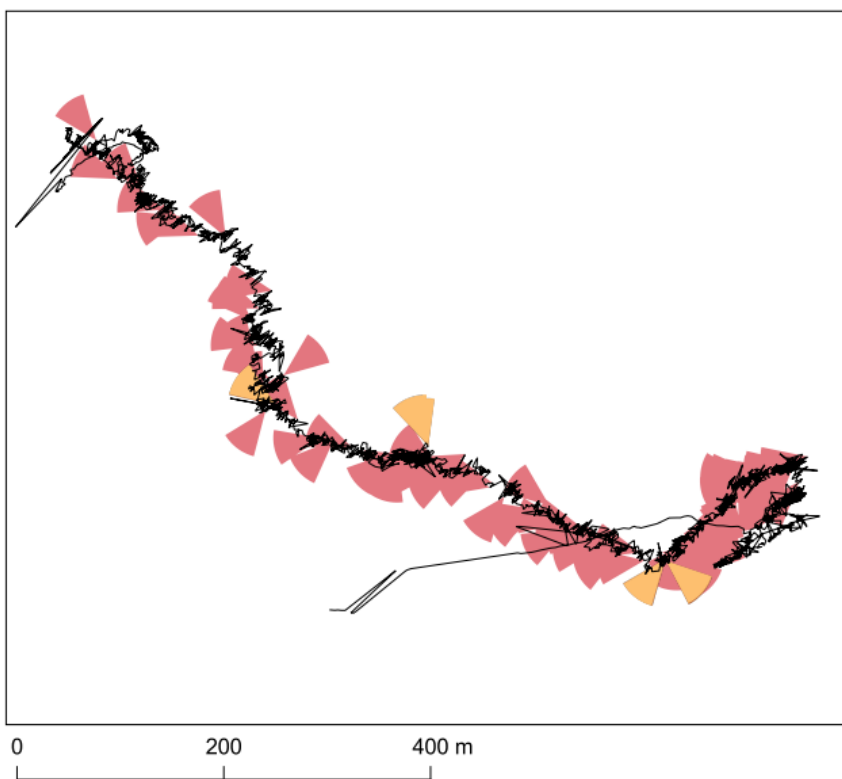


Figure C.18 Classification of substrate EX 1806 Dive 10

## EX 1806 Dive 11



Date: June 25, 2018  
 Max Depth: 1716 Meters  
 Bottom Time: 6 Hours 2 Minutes 42 Seconds

- Dive Path
- CMECS Substrate Units
- Fine:Fine
  - Fine:Rock
  - Rock:Rock
  - Fine:Coarse
  - Coarse:Fine
  - Coarse:Coarse
  - Rock:Fine
  - Rock:Coarse
  - Coarse:Rock

ESPG: 4326 - WGS 84

Figure C.19 Classification of substrate EX 1806 Dive 11

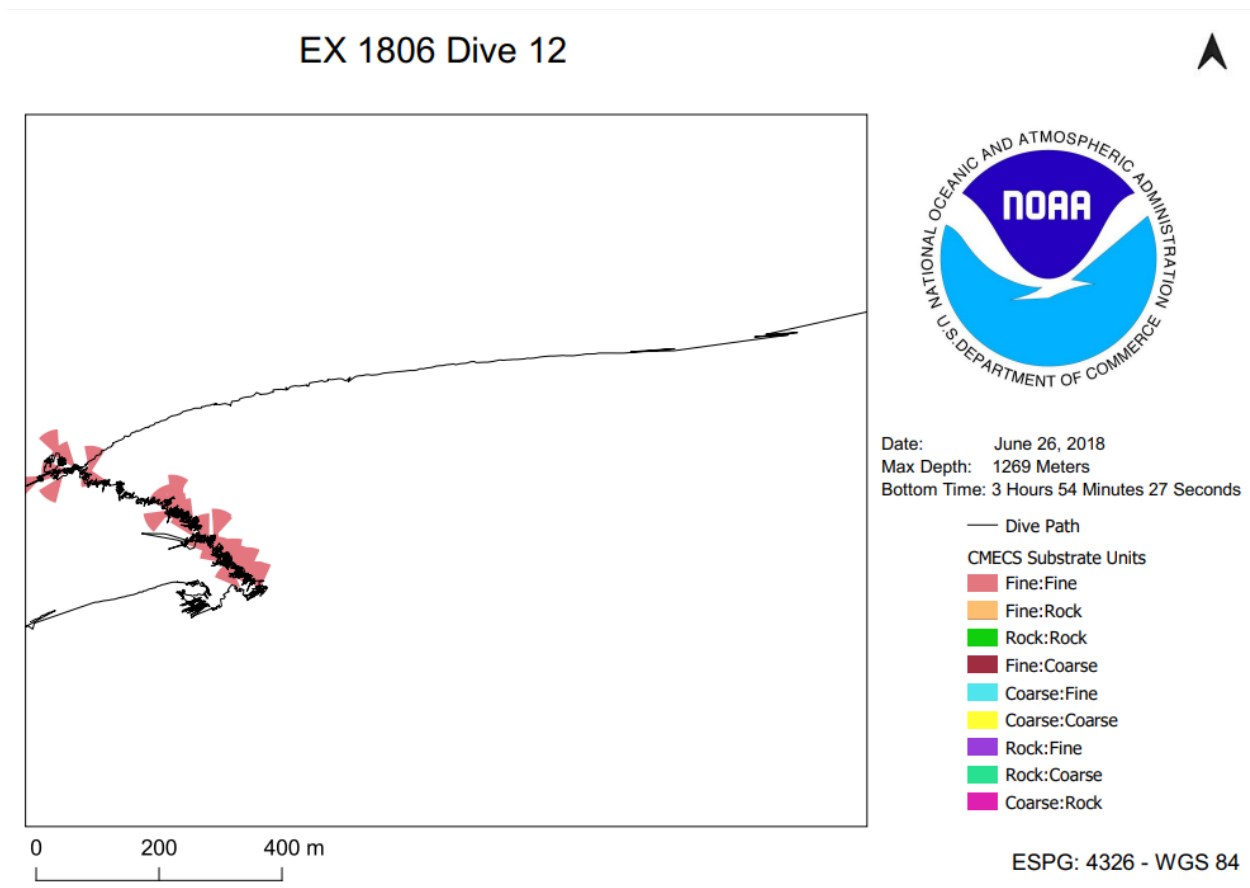


Figure C.20 Classification of substrate EX 1806 Dive 12



## EX 1806 Dive 13

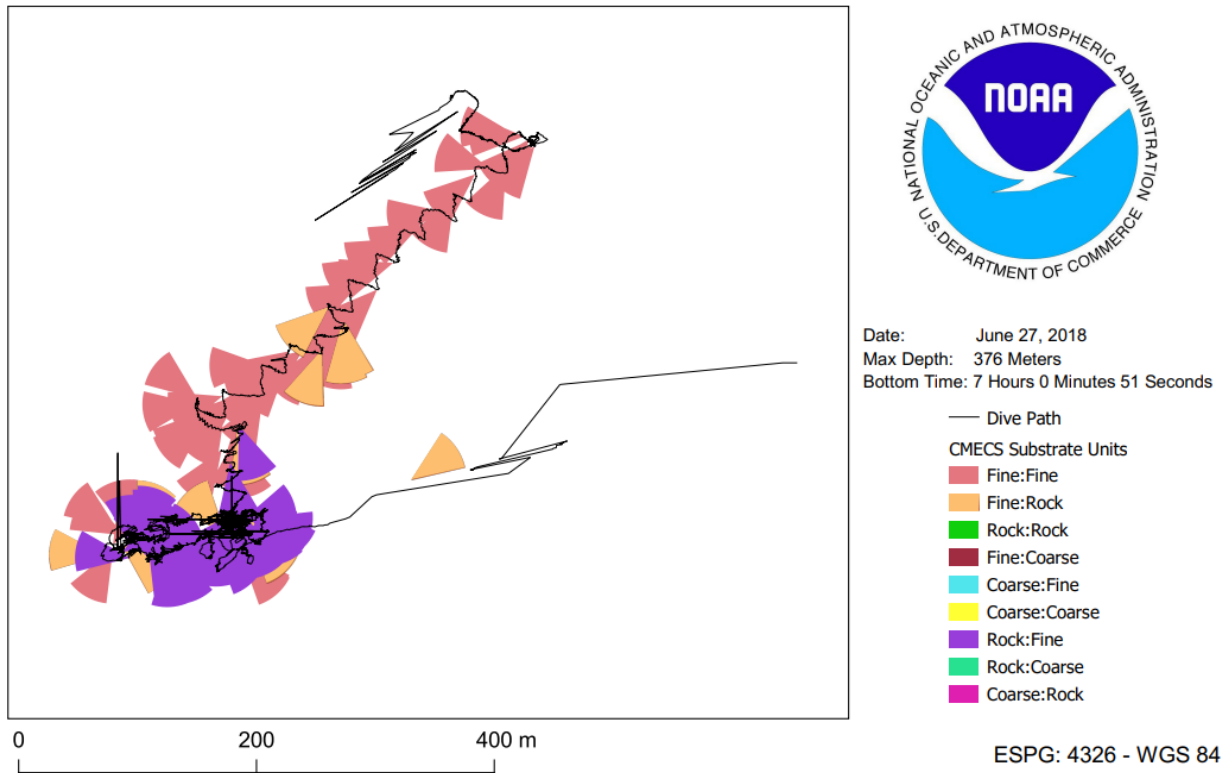


Figure C.21 Classification of substrate EX 1806 Dive 13

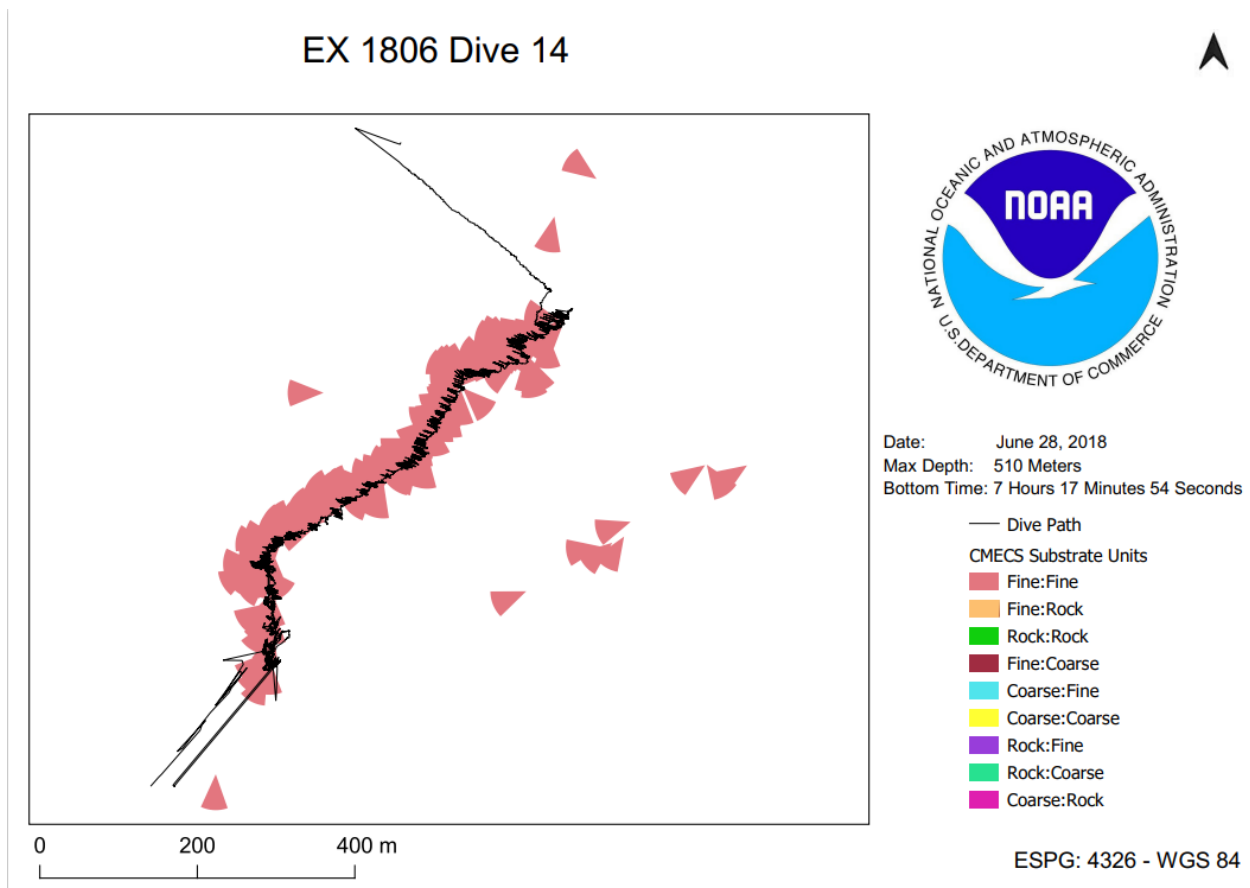


Figure C.22 Classification of substrate EX 1806 Dive 14

## EX 1806 Dive 16

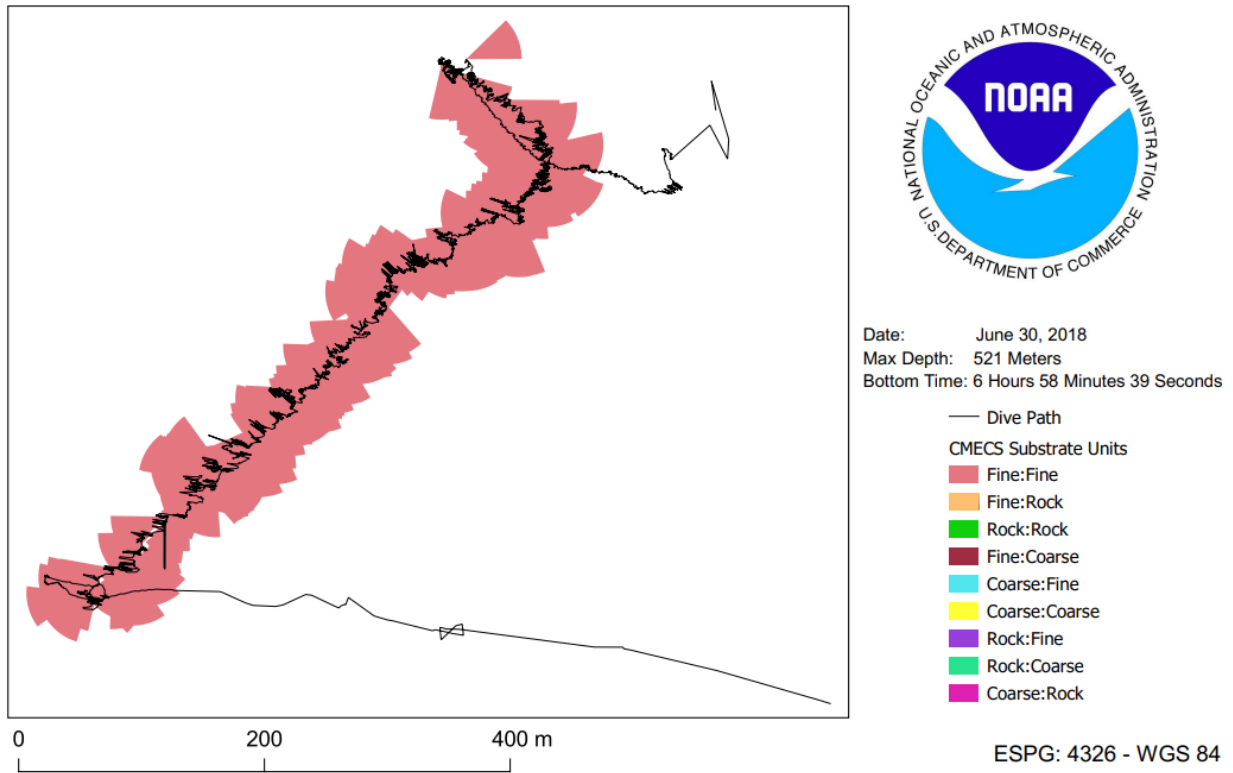


Figure C.23 Classification of substrate EX 1806 Dive 16

## EX 1806 Dive 17

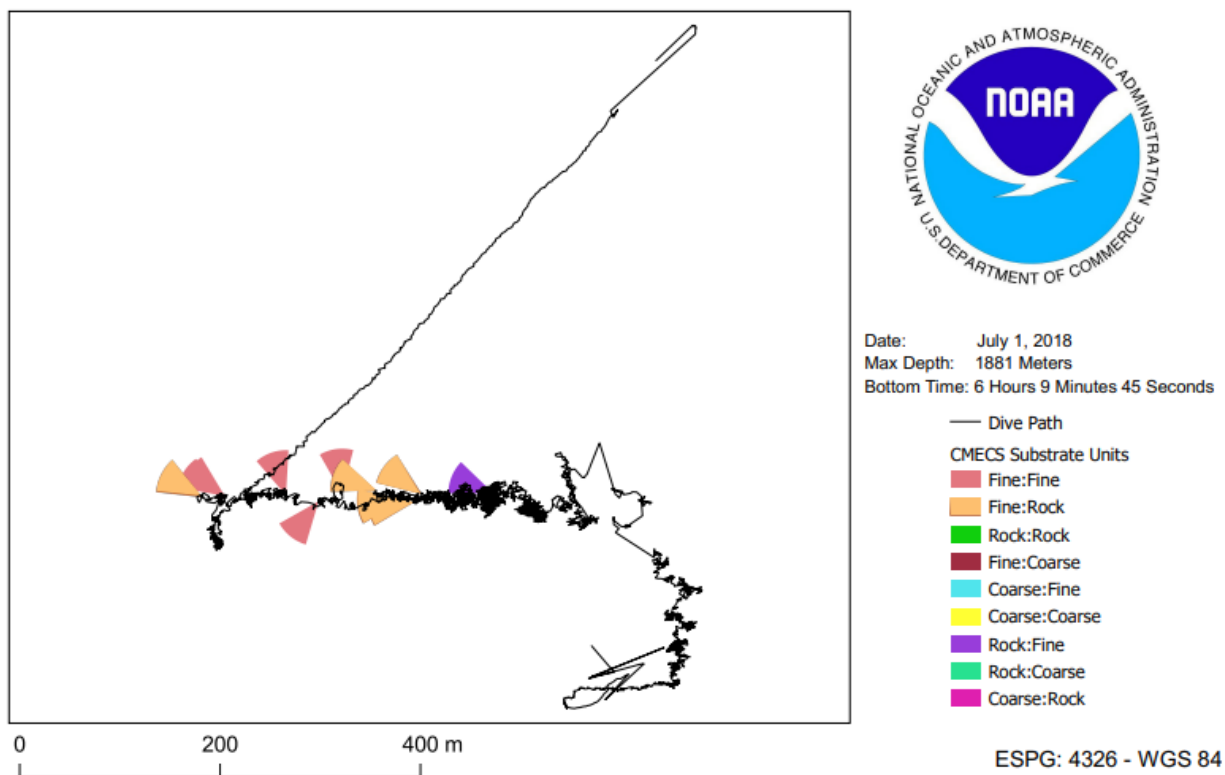


Figure C.24 Classification of substrate EX 1806 Dive 17

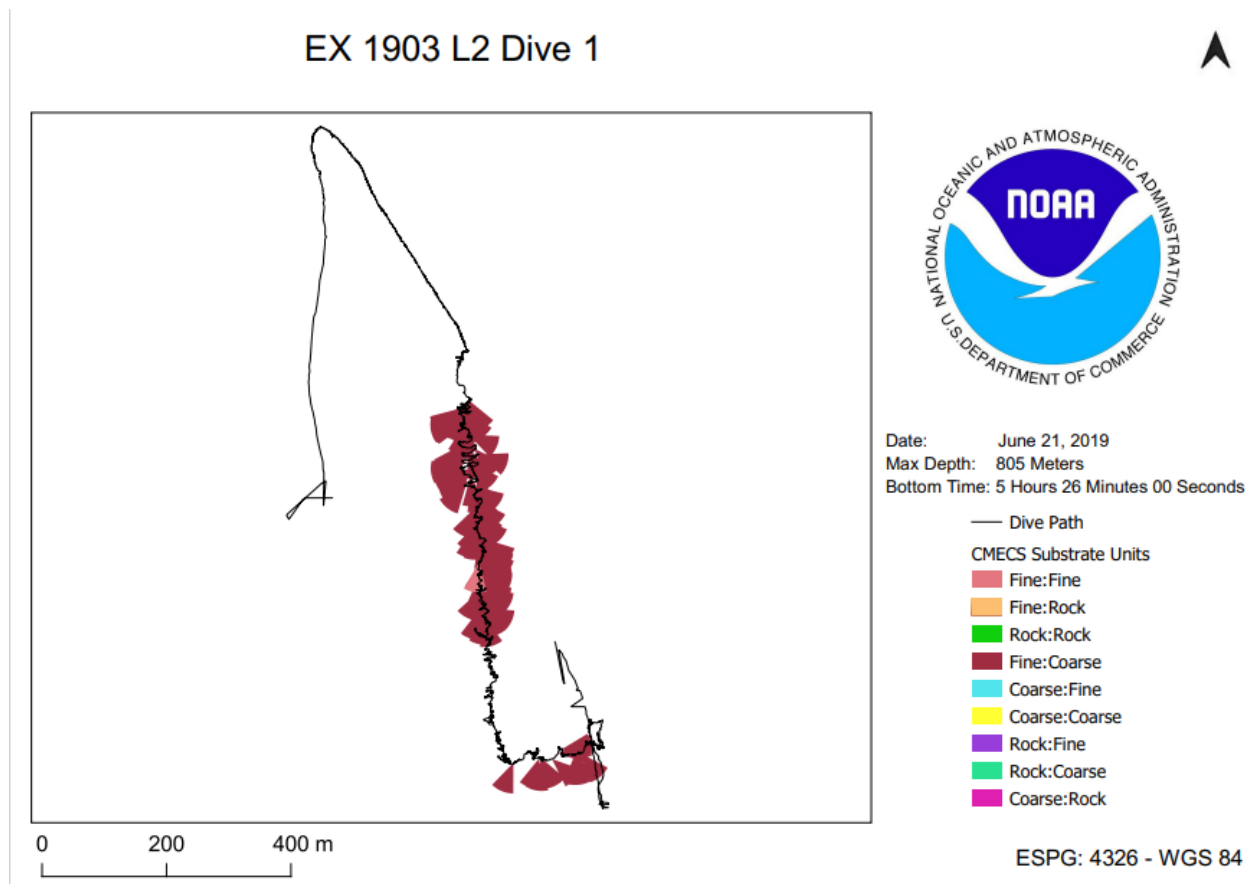


Figure C.25 Classification of substrate EX 1903 L2 Dive 01

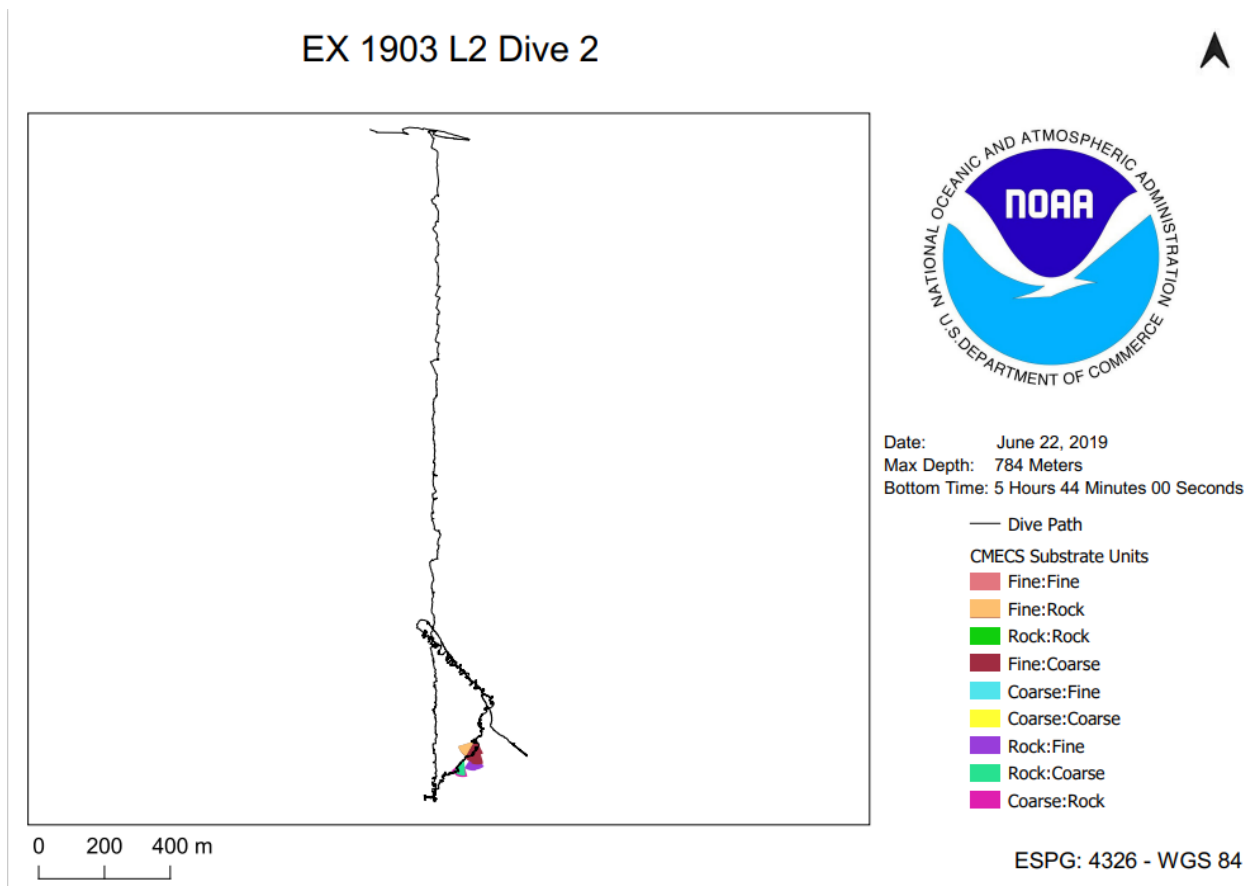


Figure C.26 Classification of substrate EX 1903 L2 Dive 02

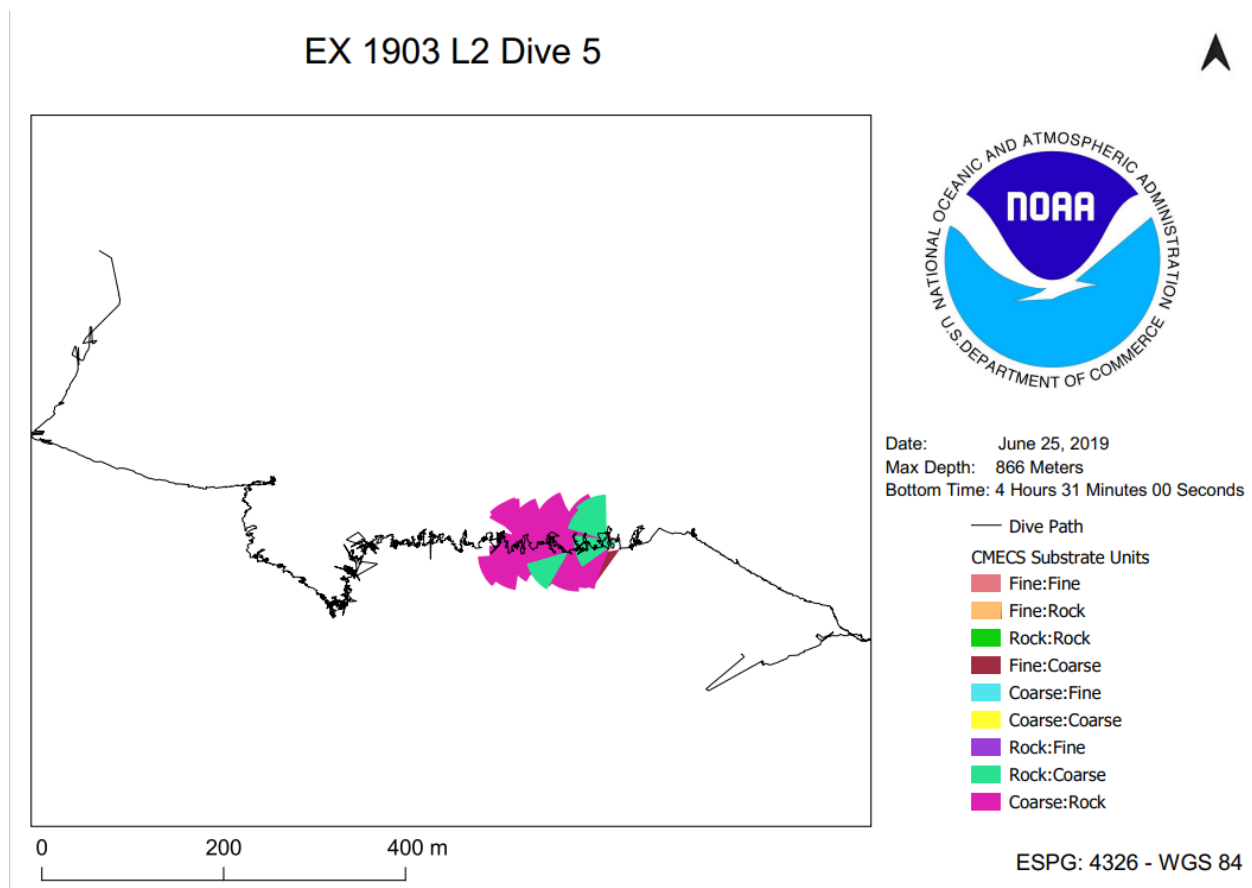
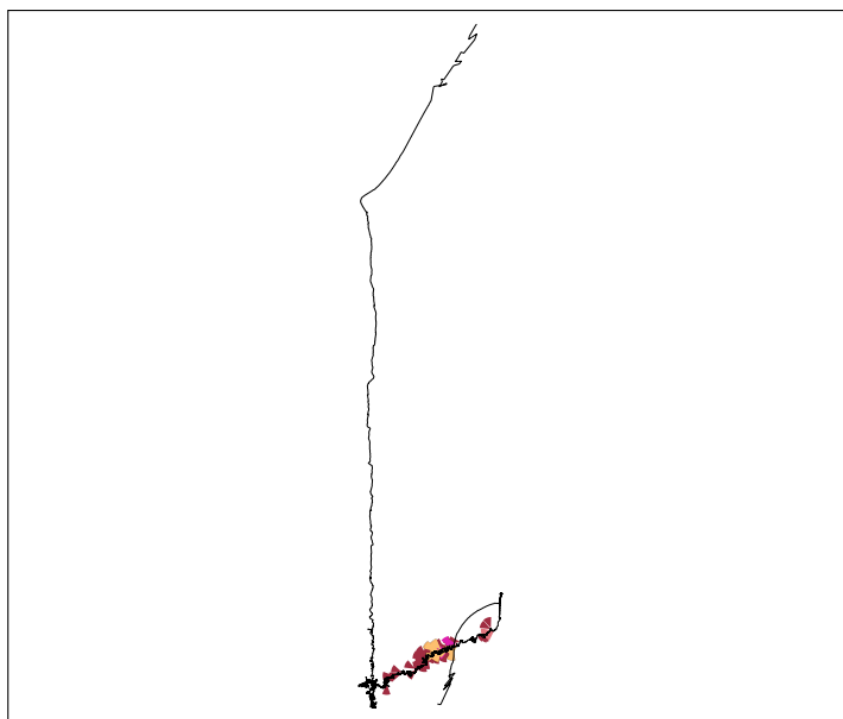
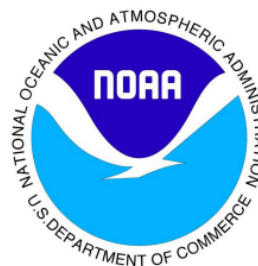


Figure C.27 Classification of substrate EX 1903 L2 Dive 05

## EX 1903 L2 Dive 6



0 200 400 m



Date: June 27, 2019  
 Max Depth: 841 Meters  
 Bottom Time: 5 Hours 52 Minutes 00 Seconds

- Dive Path
- CMECS Substrate Units
- Fine:Fine
- Fine:Rock
- Rock:Rock
- Fine:Coarse
- Coarse:Fine
- Coarse:Coarse
- Rock:Fine
- Rock:Coarse
- Coarse:Rock

ESPG: 4326 - WGS 84

Figure C.28 Classification of substrate EX 1903 L2 Dive 06



## EX 1903 L2 Dive 8

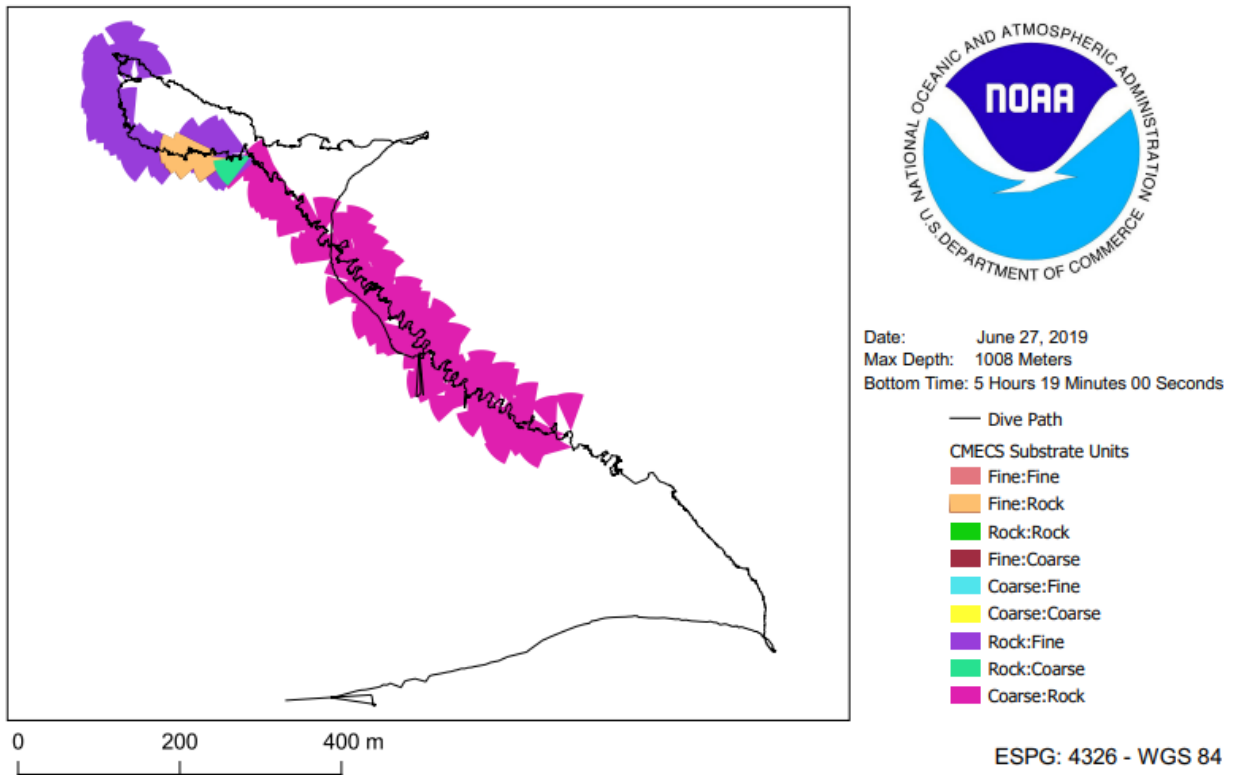


Figure C.29 Classification of substrate EX 1903 L2 Dive 08

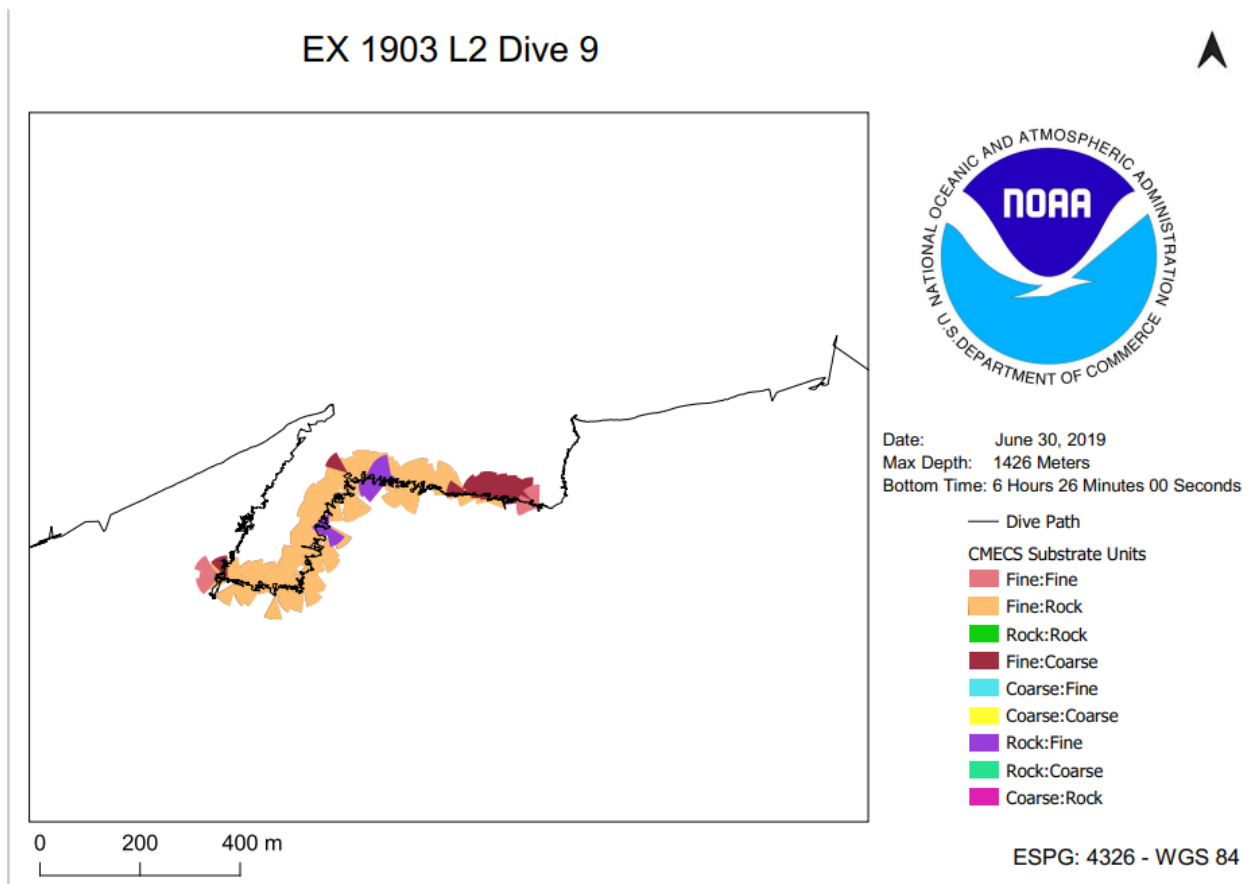


Figure C.30 Classification of substrate EX 1903 L2 Dive 09

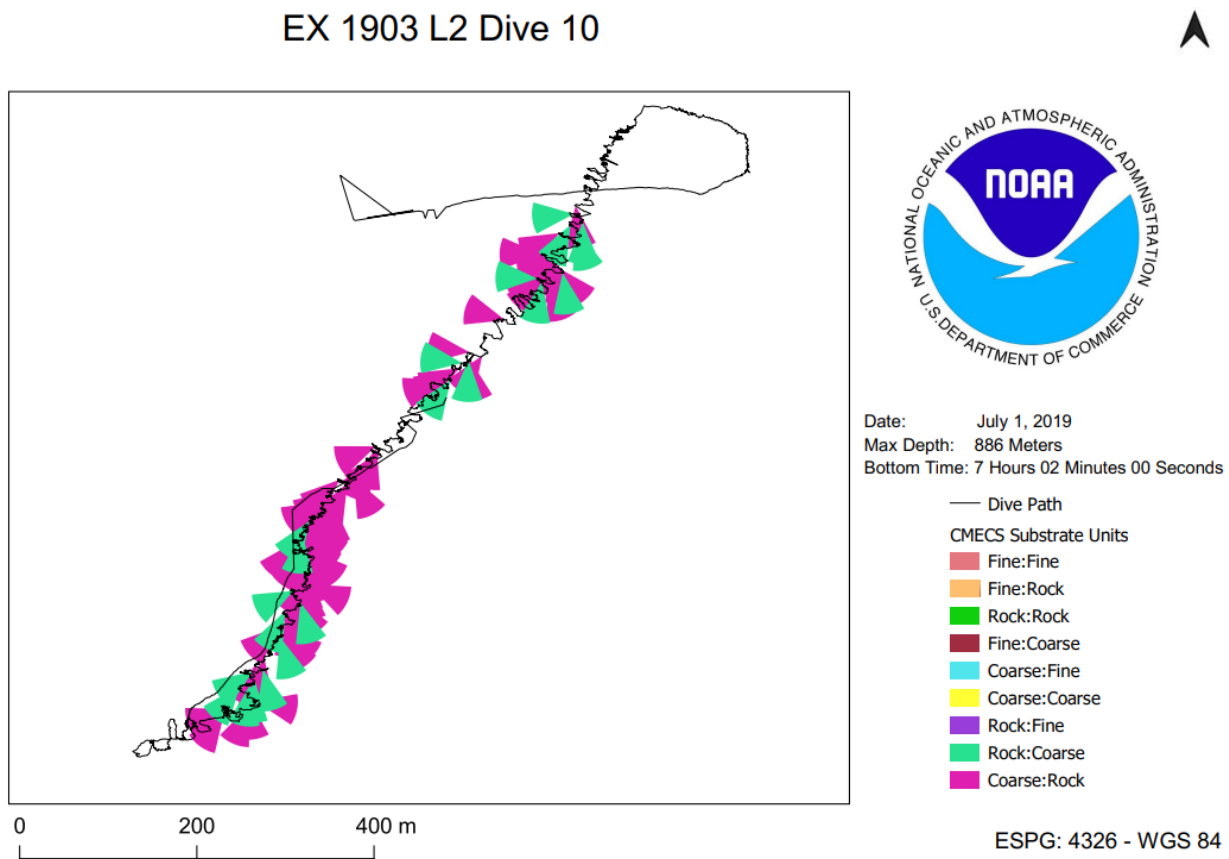
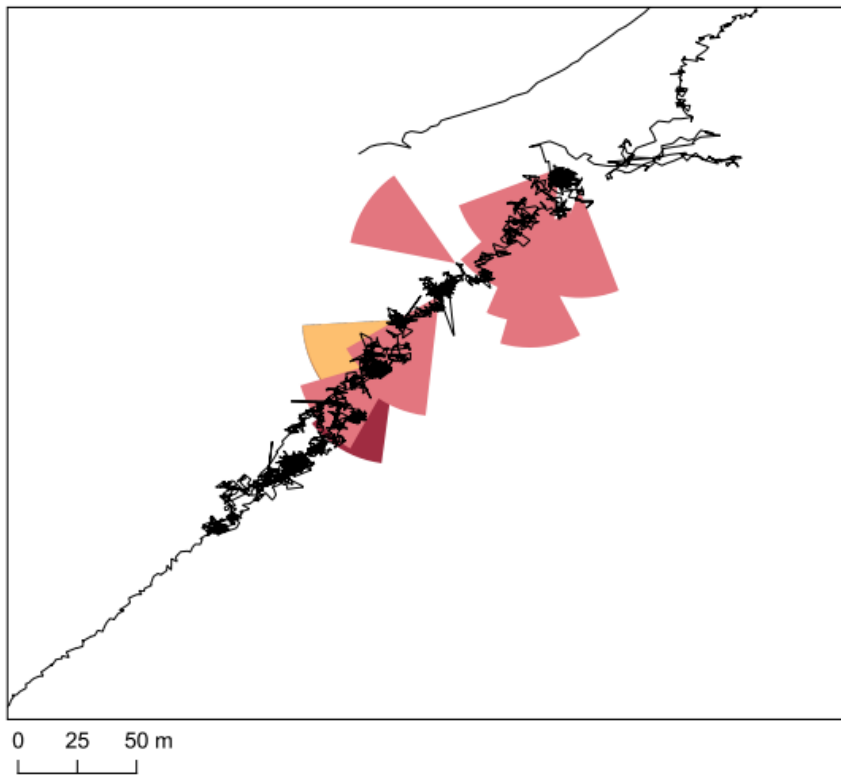


Figure C.31 Classification of substrate EX 1903 L2 Dive 10

## EX 1903 L2 Dive 11



Date: July 3, 2019  
Max Depth: 1348 Meters  
Bottom Time: 3 Hours 29 Minutes 00 Seconds

### CMECS Substrate Units

- Fine:Fine
- Fine:Rock
- Rock:Rock
- Fine:Coarse
- Coarse:Fine
- Coarse:Coarse
- Rock:Fine
- Rock:Coarse
- Coarse:Rock
- Dive Path

ESPG: 4326 - WGS 84

Figure C.32 Classification of substrate EX 1903 L2 Dive 11

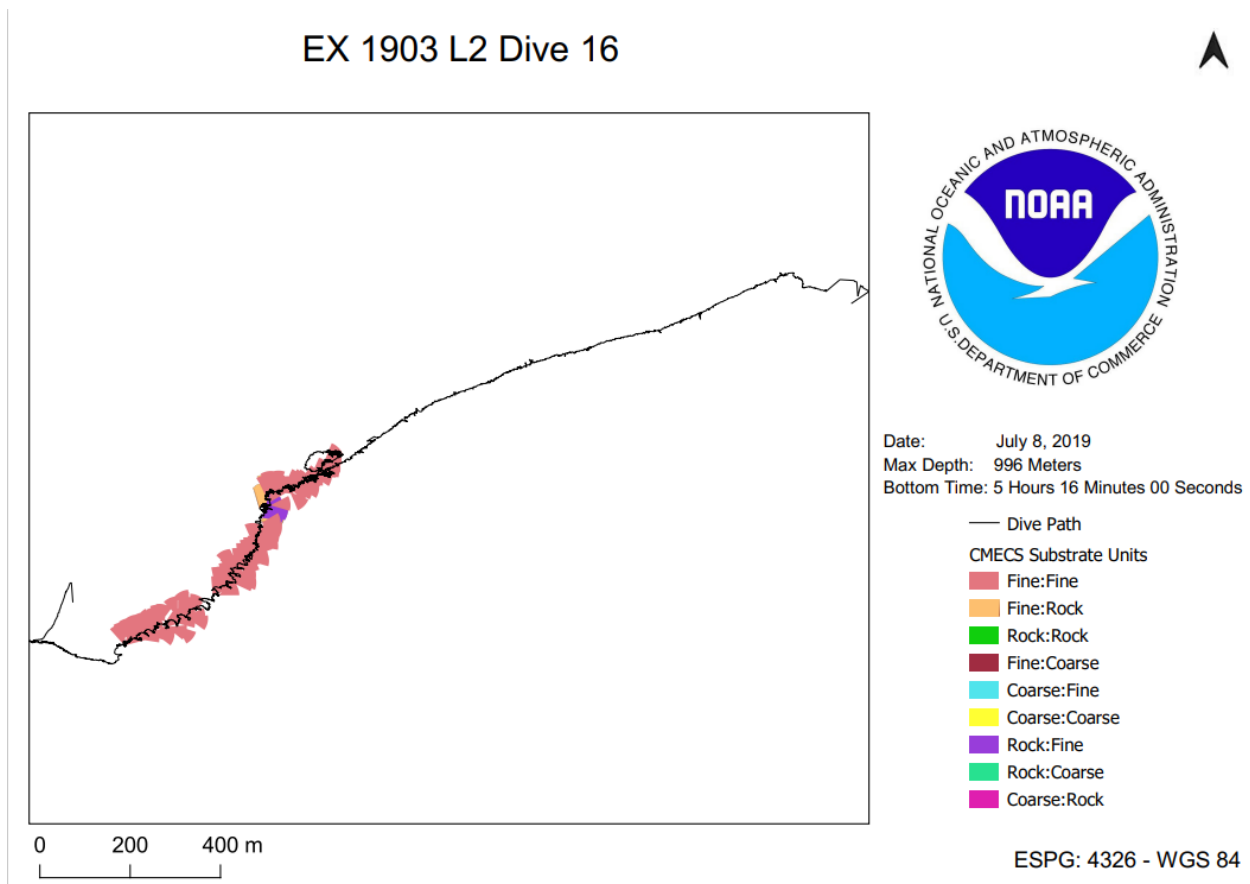


Figure C.33 Classification of substrate EX 1903 L2 Dive 16

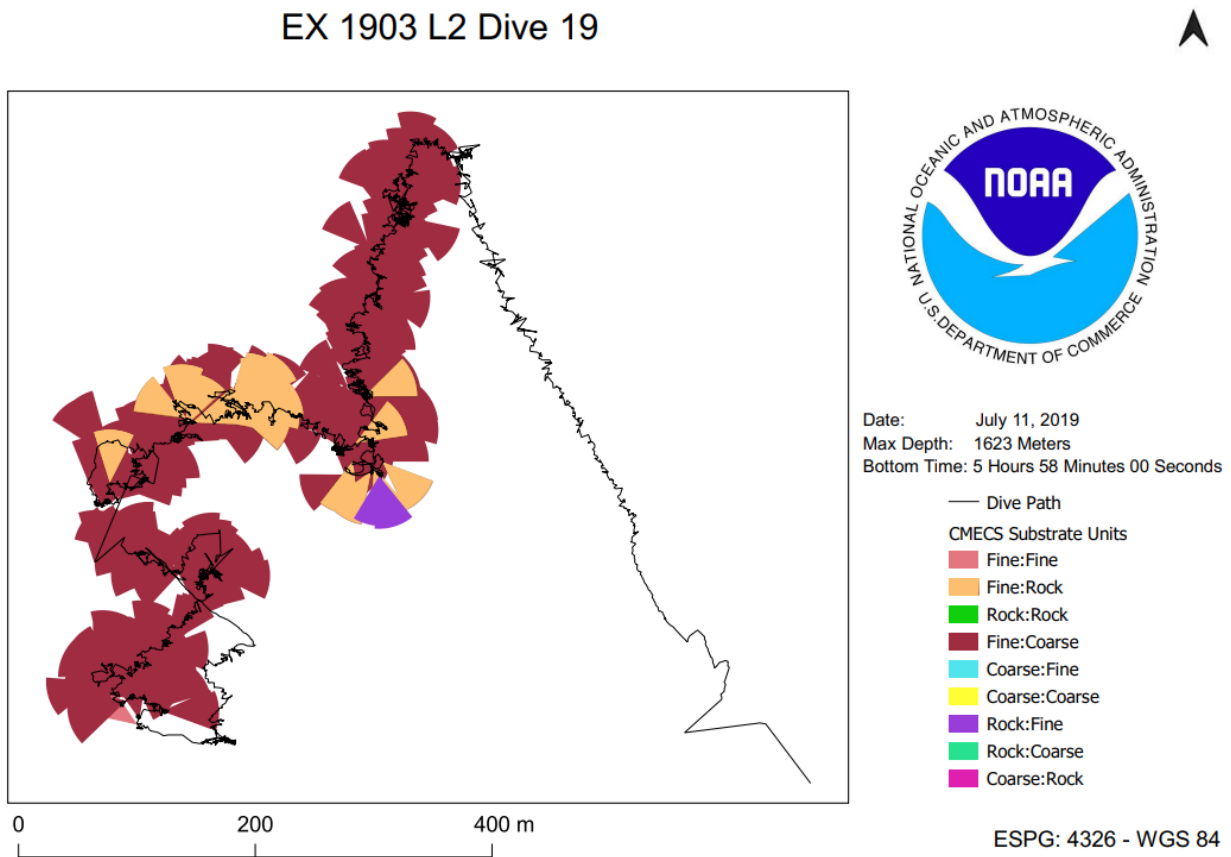


Figure C.34 Classification of substrate type throughout EX 1903 L2 Dive 19

## APPENDIX D

### BACKSCATTER COMAPARISON OF ROV SUBSTRATE MAPS & DISTRIBUTION

#### BOXPLOTS

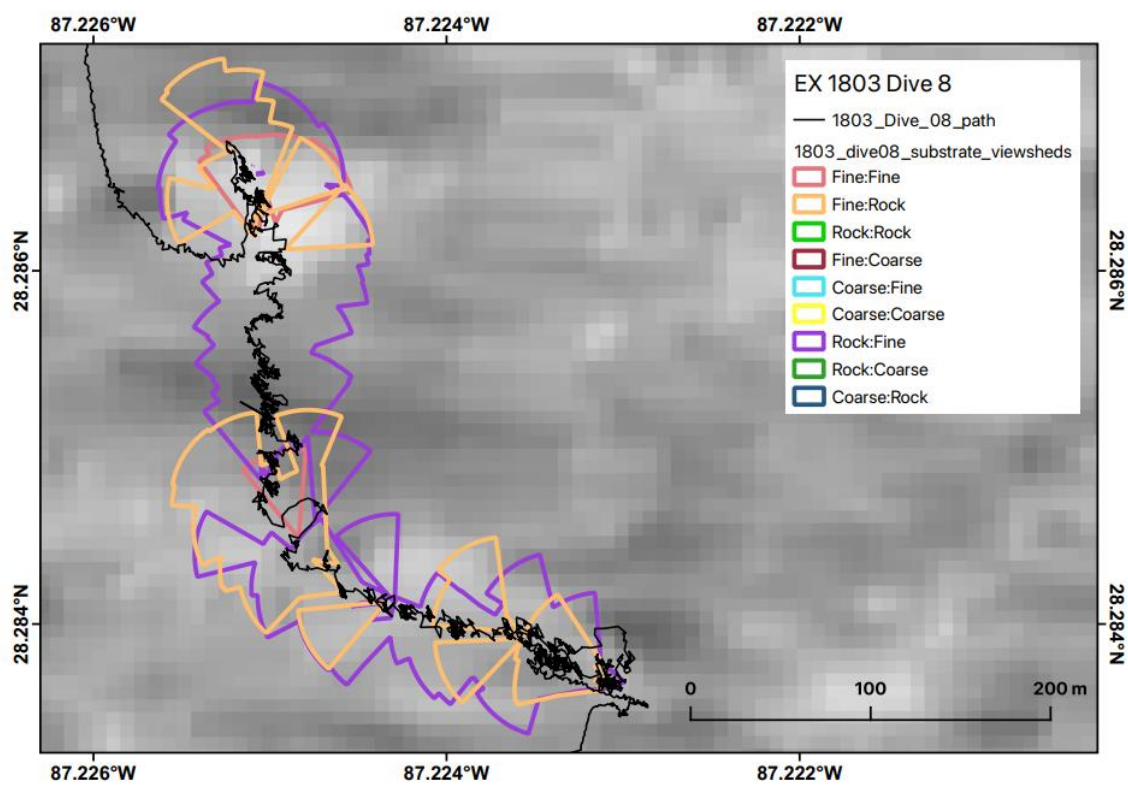


Figure D.1 Backscatter Comparison of EX 1803 Dive 08



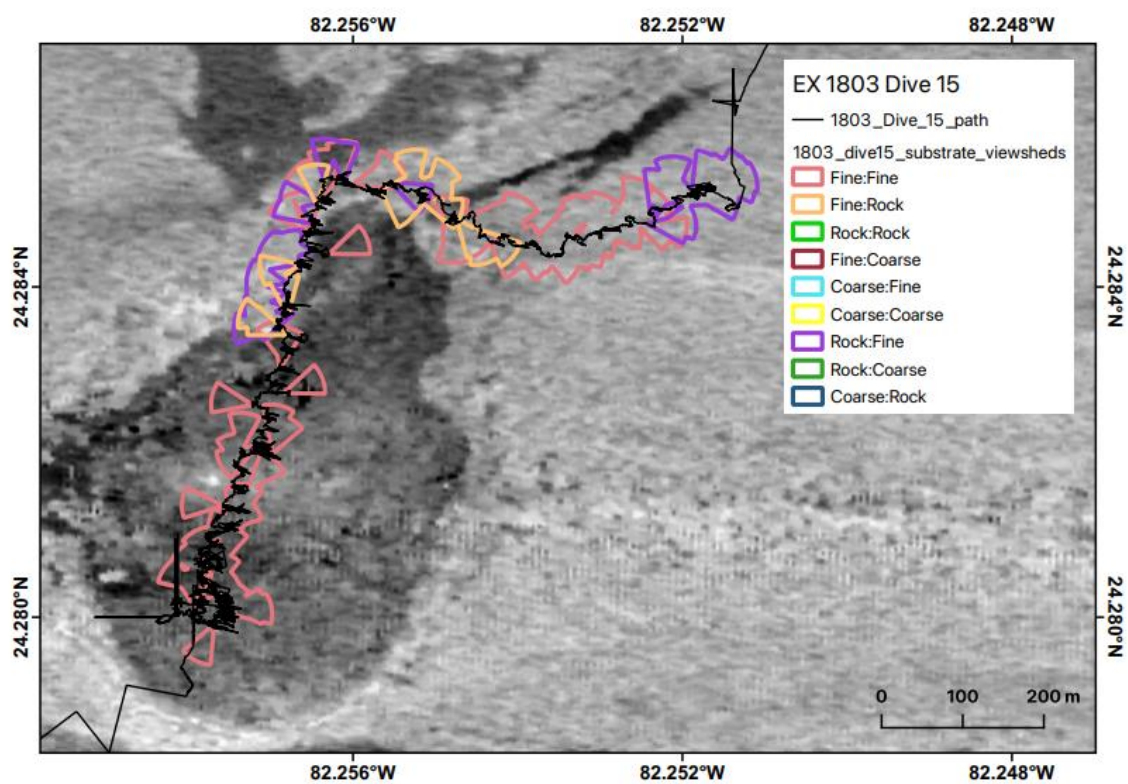


Figure D.2 Backscatter Comparison of EX 1803 Dive 15

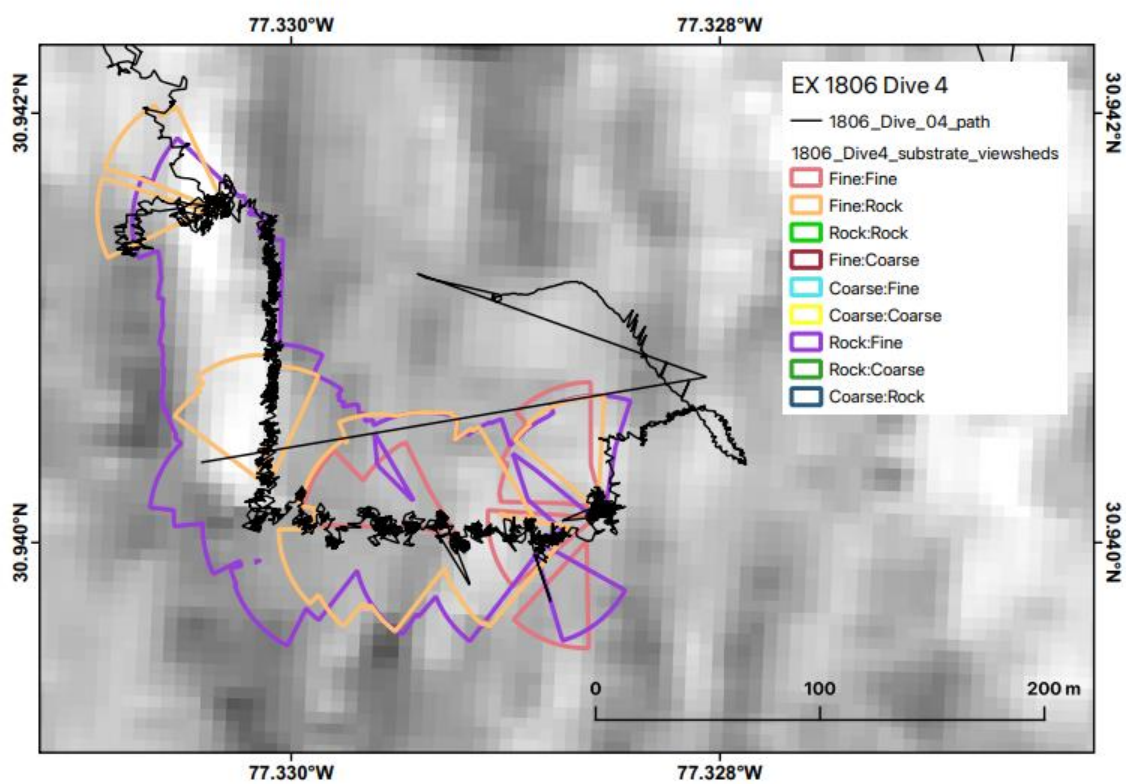


Figure D.3 Backscatter Comparison of EX 1806 Dive 04

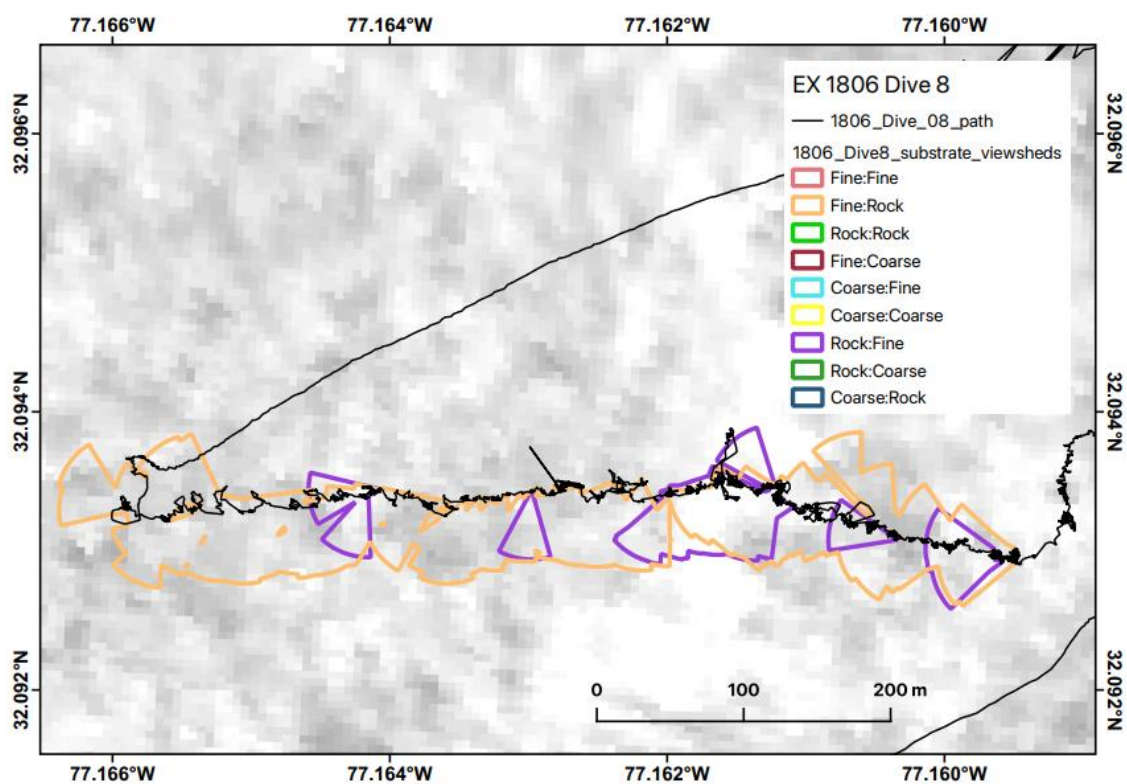


Figure D.4 Backscatter Comparison of EX 1806 Dive 08

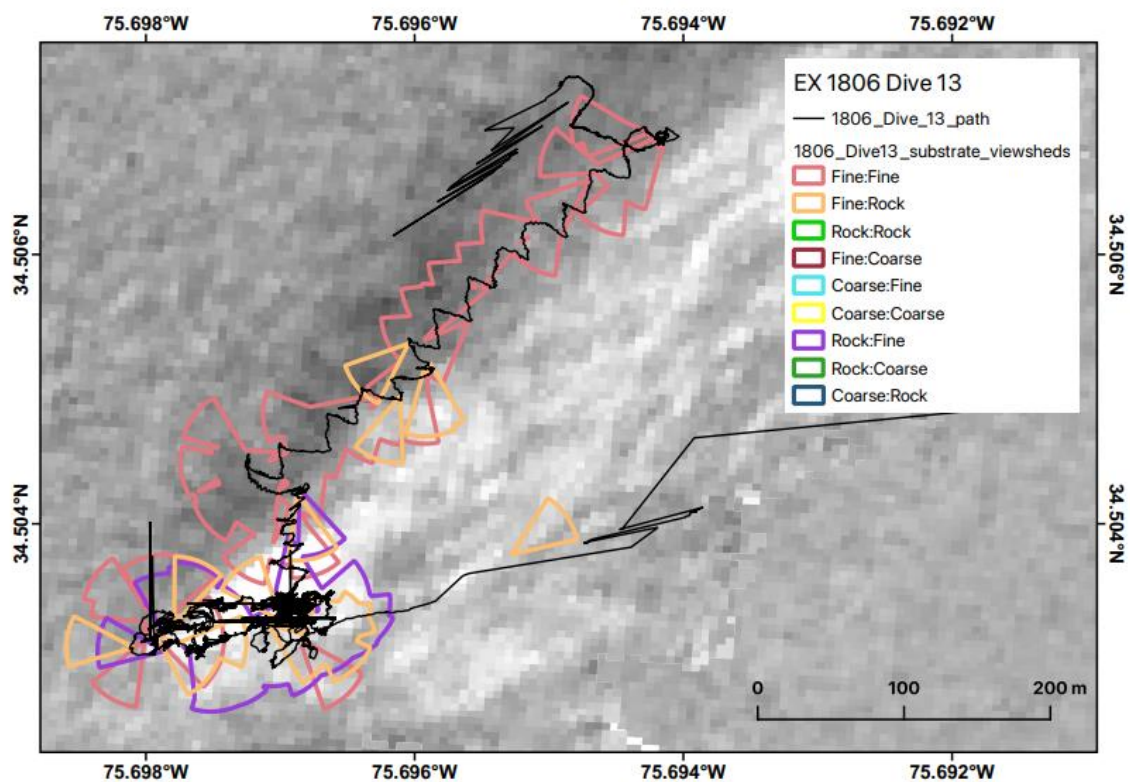


Figure D.5 Backscatter Comparison of EX 1806 Dive 13

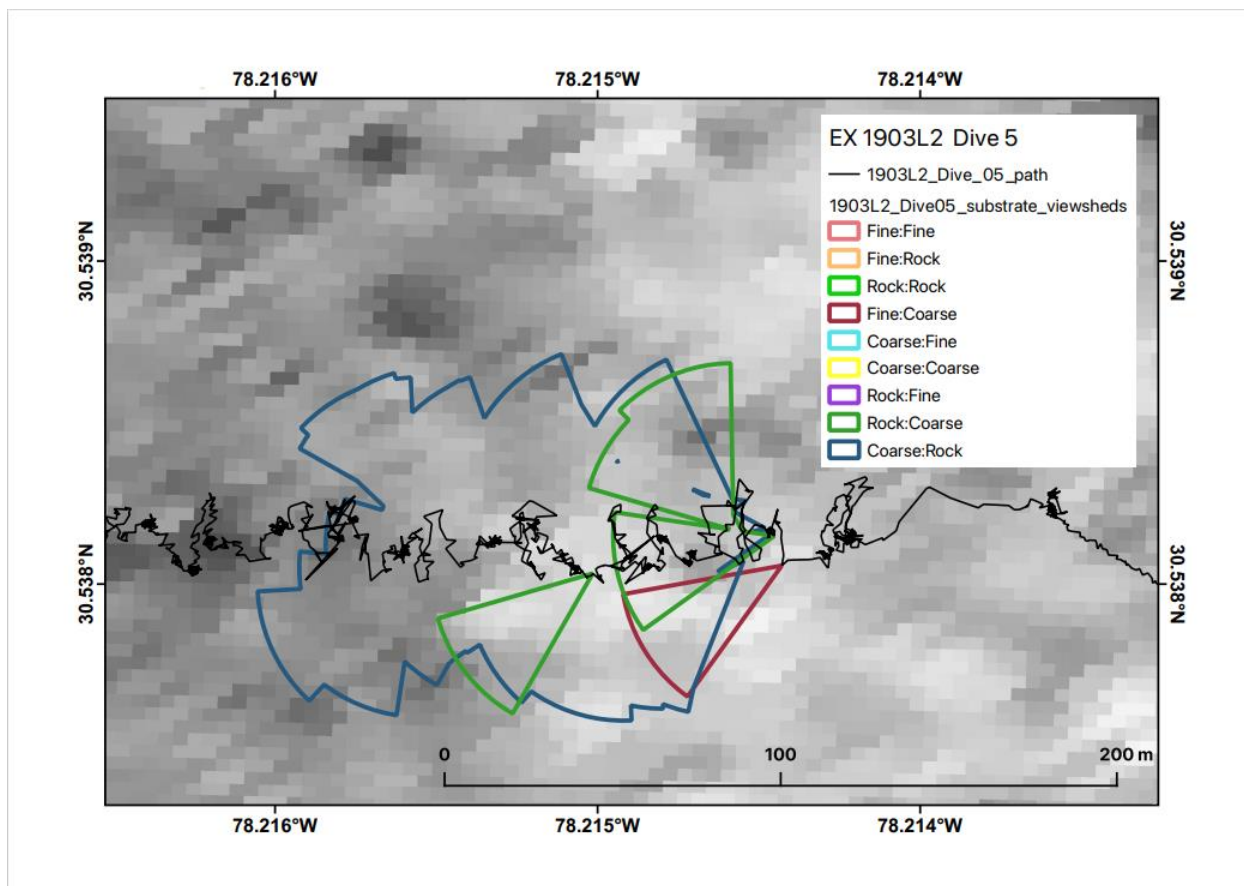


Figure D.6 Backscatter Comparison of EX 1903 L2 Dive 05



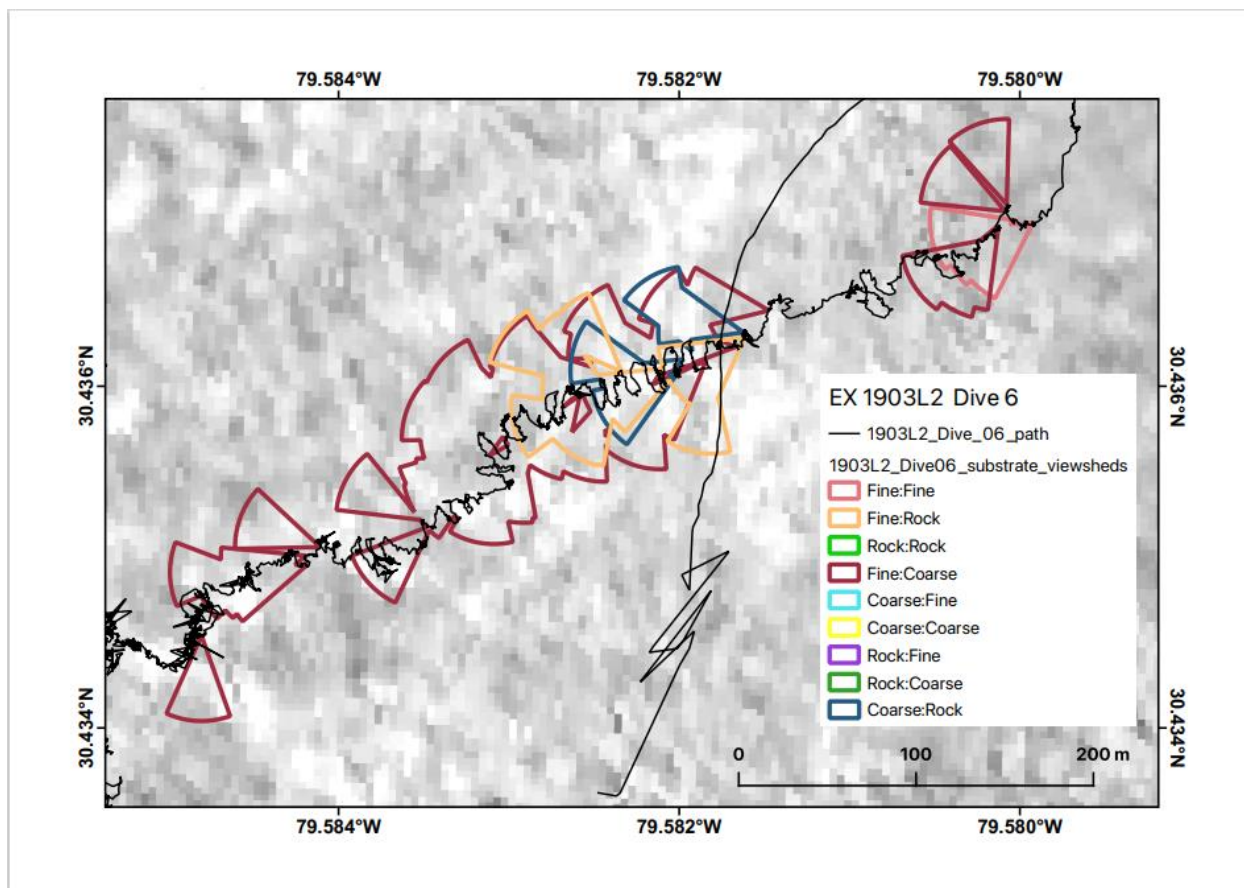


Figure D.7 Backscatter Comparison of EX 1903 L2 Dive 06

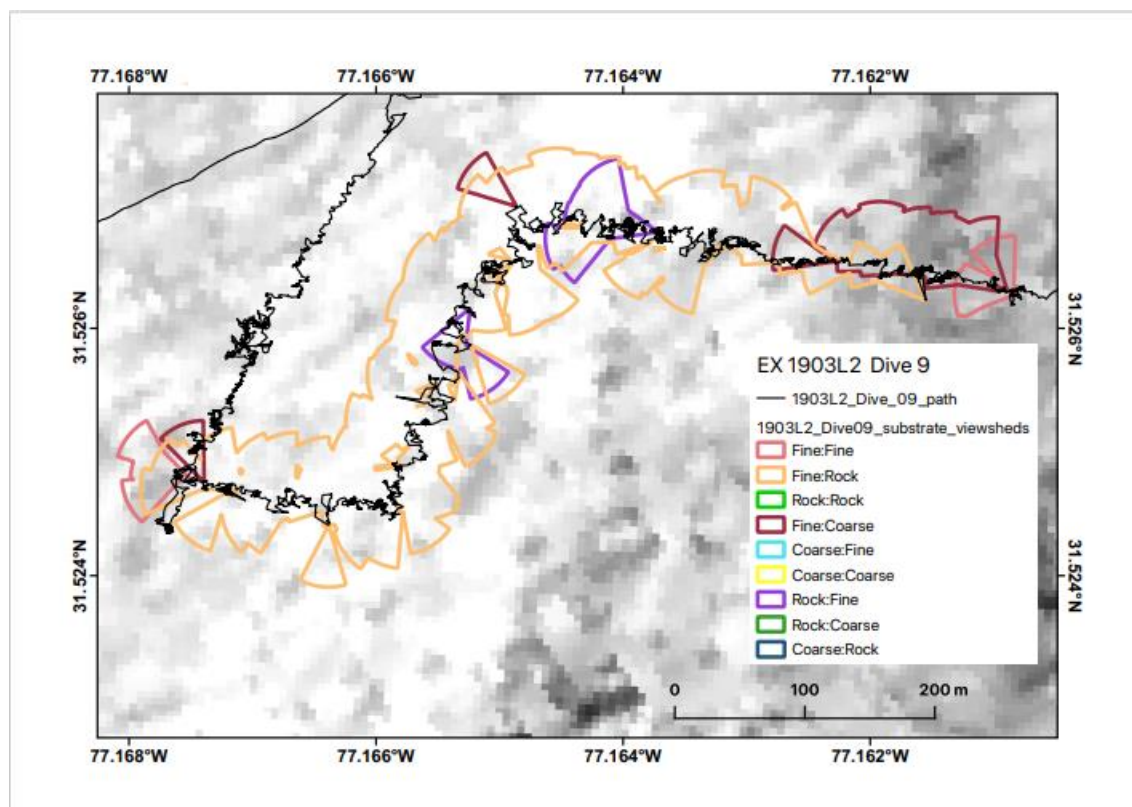


Figure D.8 Backscatter Comparison of EX 1903 L2 Dive 09

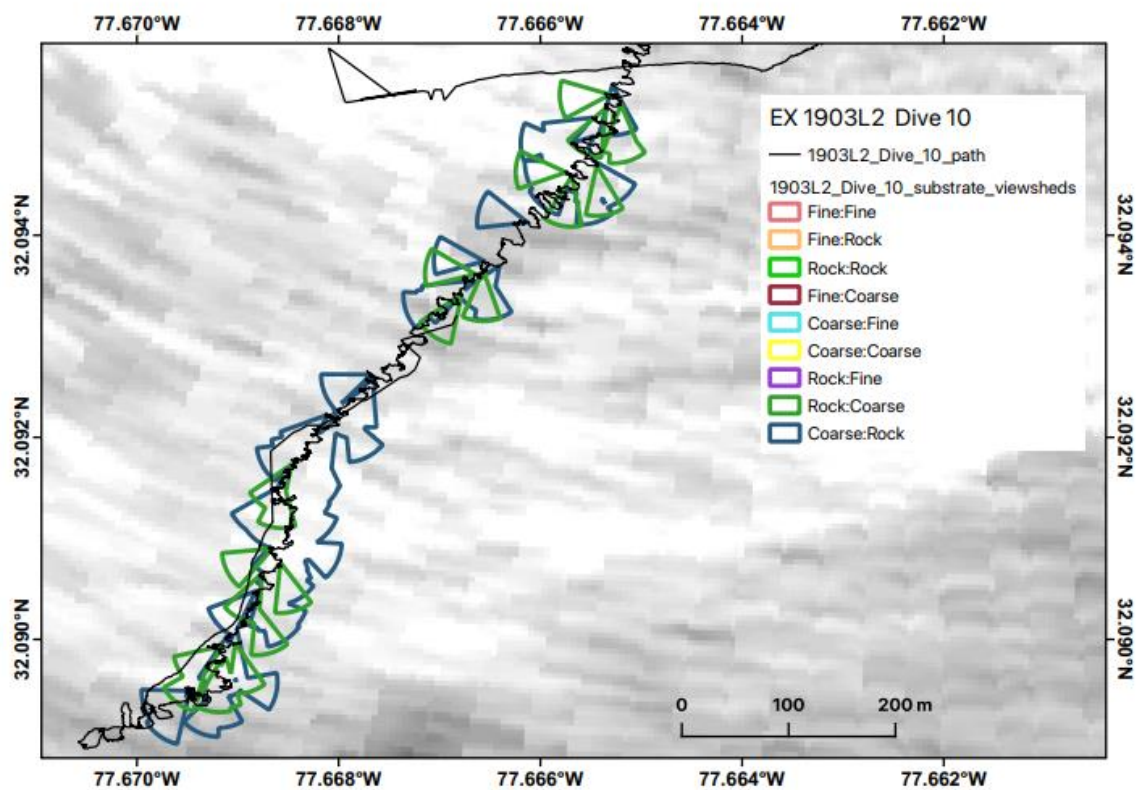


Figure D.9 Backscatter Comparison of EX 1903 L2 Dive 10



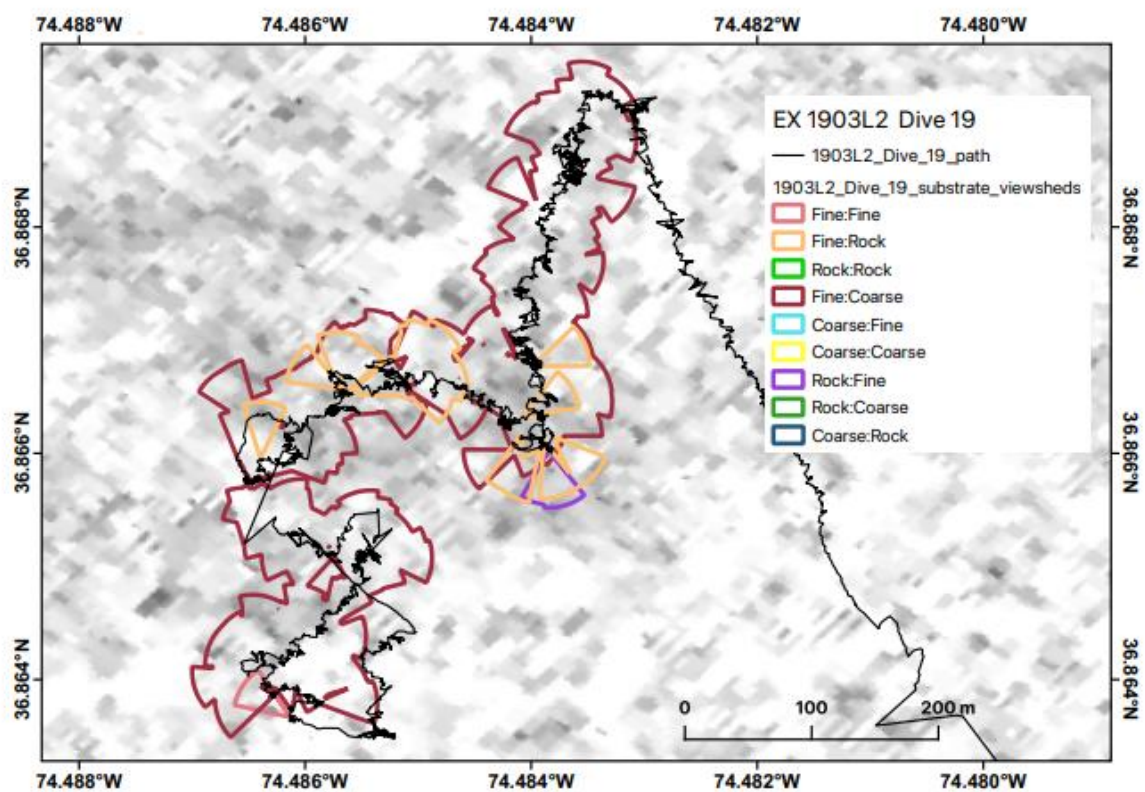


Figure D.10 Backscatter Comparison of EX 1903 L2 Dive 19

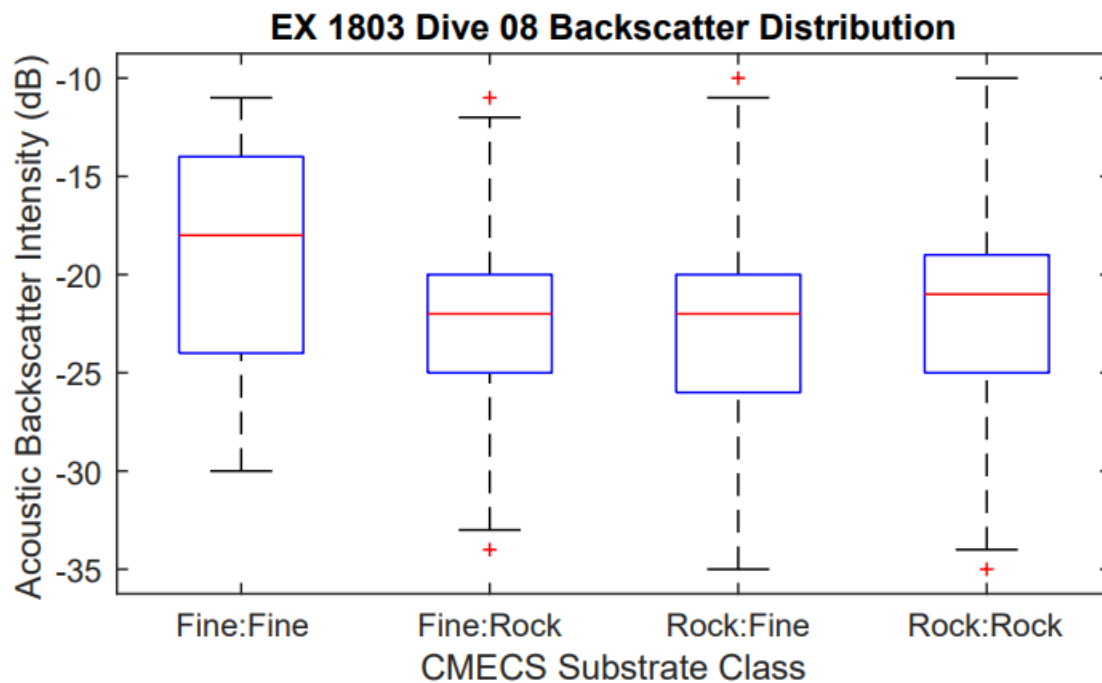


Figure D.11 EX 1803 Dive 08 Backscatter Distribution Boxplot

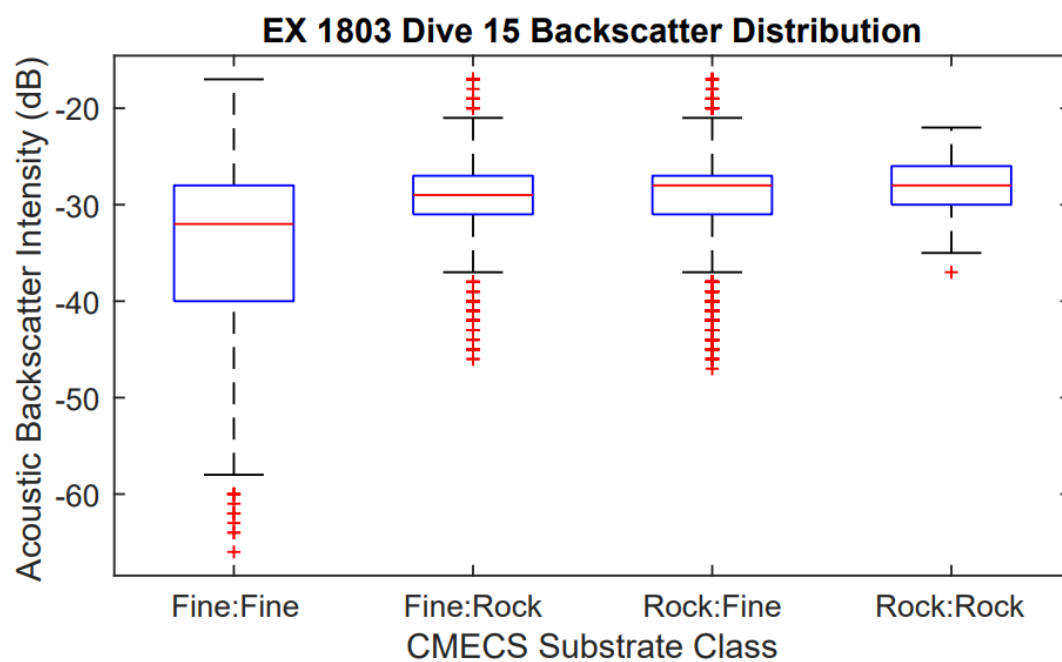


Figure D.12 EX 1803 Dive 15 Backscatter Distribution Boxplot

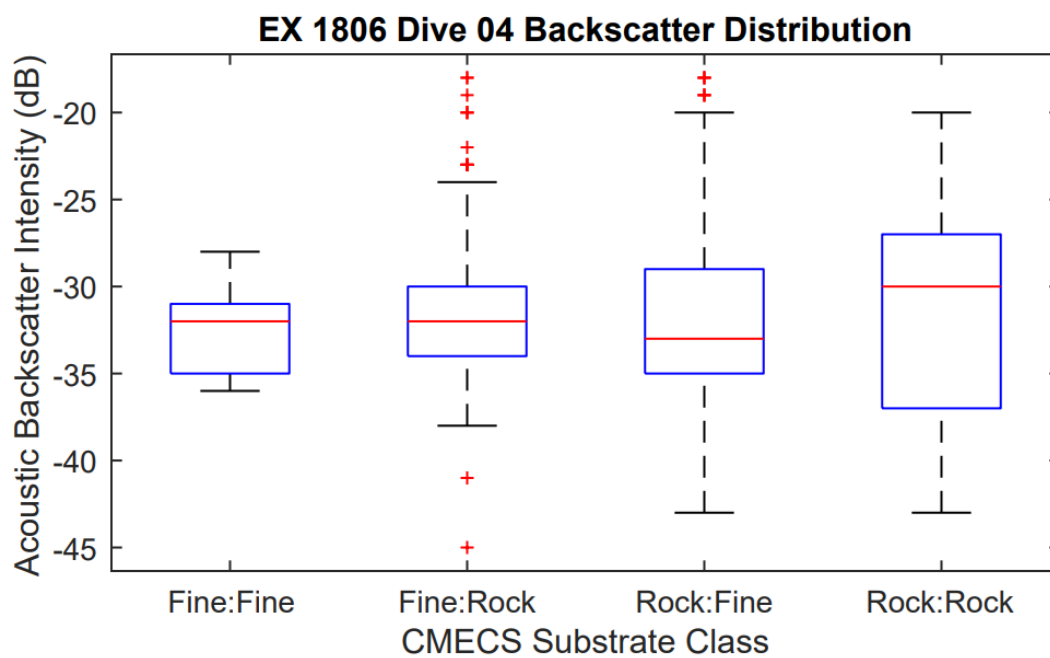


Figure D.13 EX 1806 Dive 04 Backscatter Distribution Boxplot

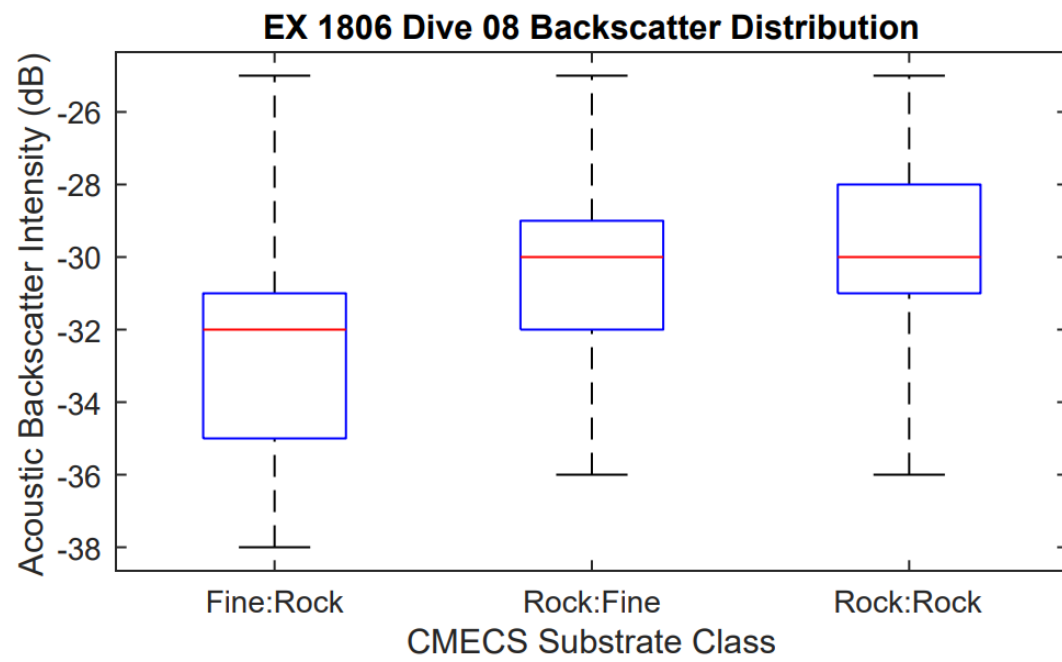


Figure D.14 EX 1806 Dive 08 Backscatter Distribution Boxplot

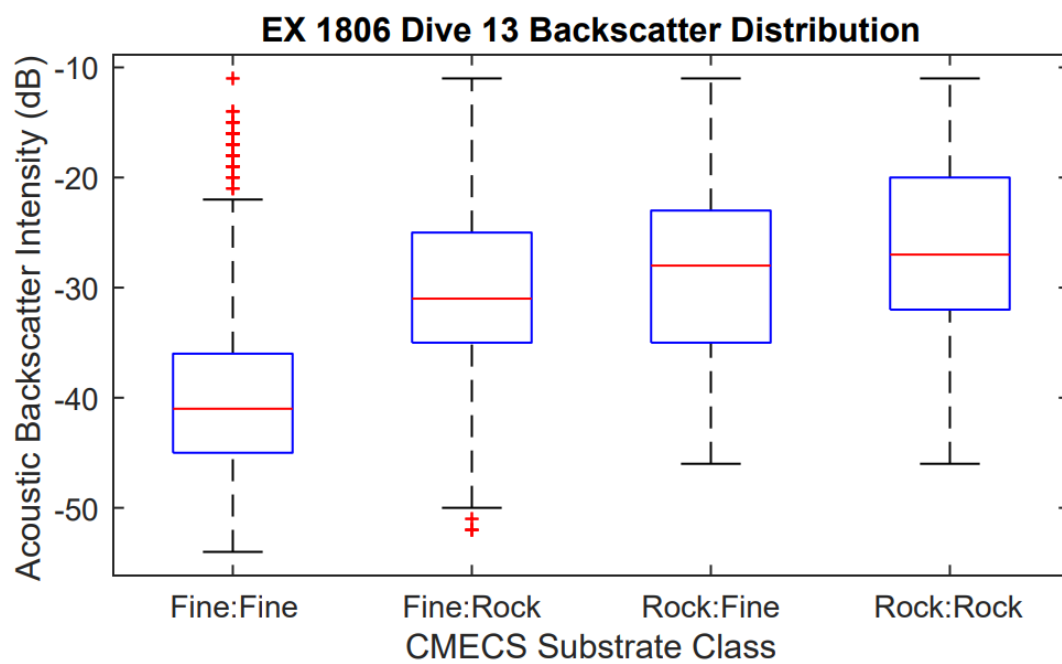


Figure D.15 EX 1806 Dive 13 Backscatter Distribution Boxplot

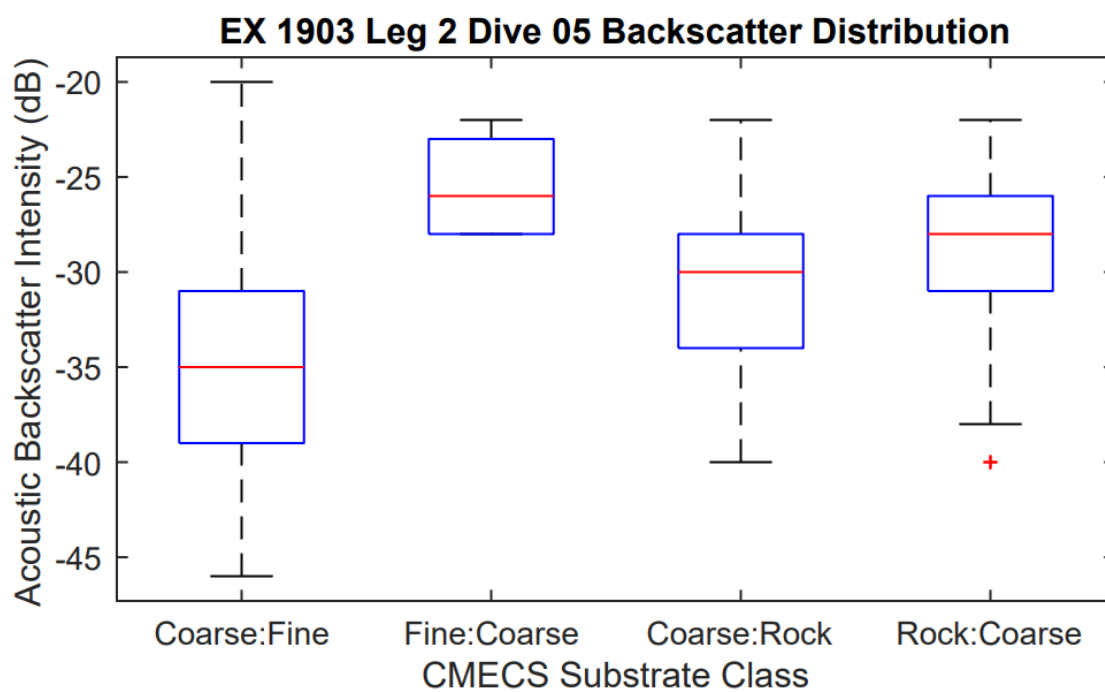


Figure D.16 EX 1903 Leg 2 Dive 05 Backscatter Distribution Boxplot

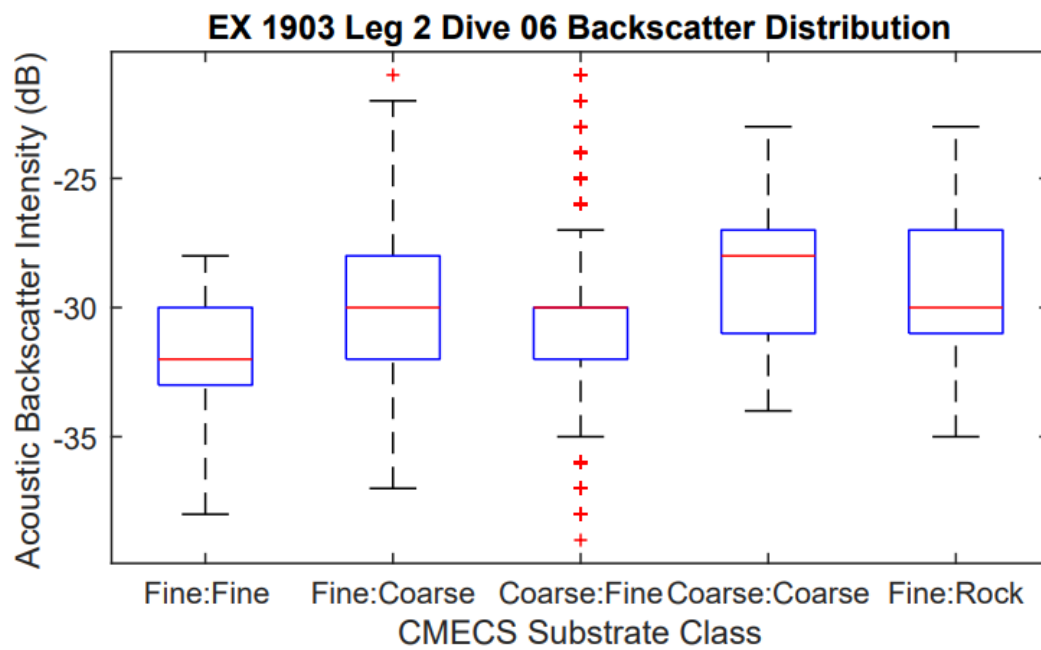


Figure D.17 EX 1903 Leg 2 Dive 06 Backscatter Distribution Boxplot



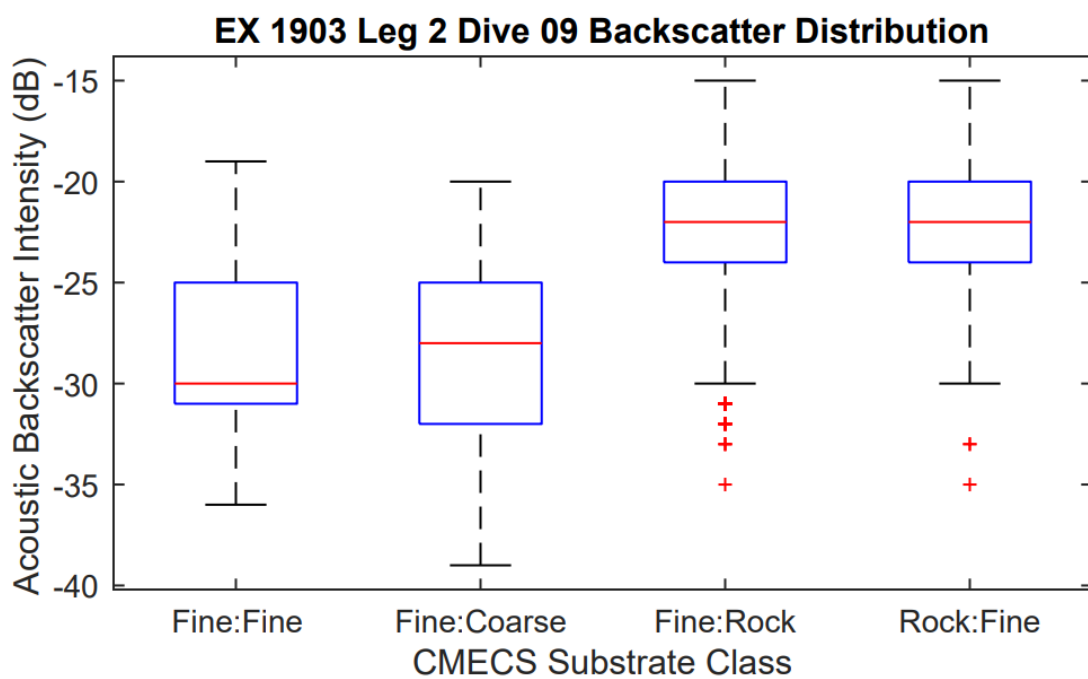


Figure D.18 EX 1903 Leg 2 Dive 09 Backscatter Distribution Boxplot

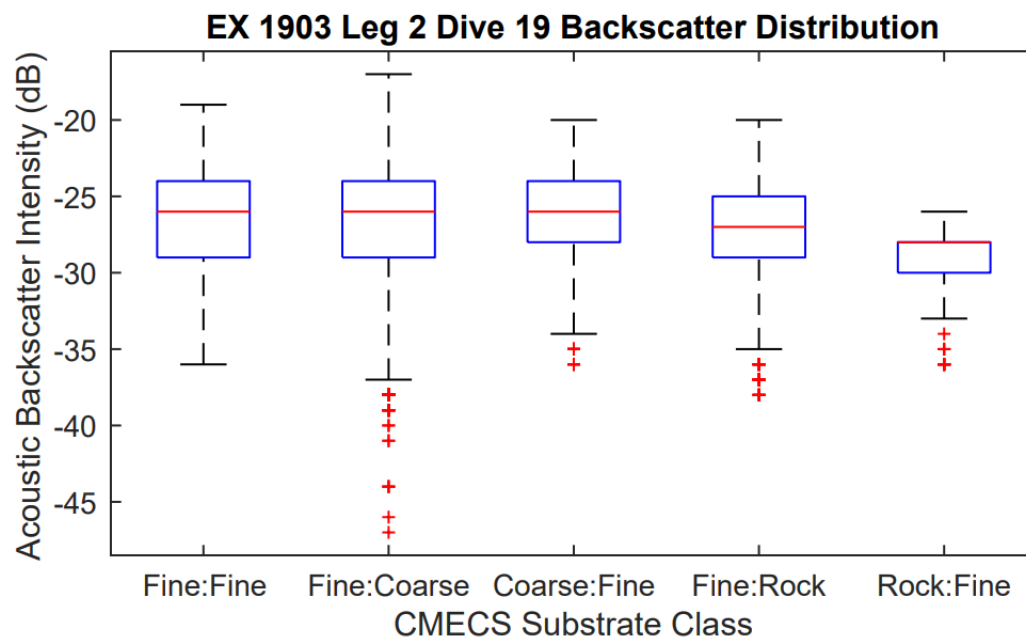


Figure D.19 EX 1903 Leg 2 Dive 19 Backscatter Distribution Boxplot

## APPENDIX E

### ROV AND sUAS FRAME EXTRACTION AND TIMESTAMP SCRIPTS

```

import os, sys
import subprocess
import glob
import pdb
import time
import math

# Extract key frames from MOV file.

exestring = 'ffmpeg -i ' + sys.argv[1] + ' -vf "select=eq(pict_type,I)" -vsync vfr ' + \
+ sys.argv[1].split('.')[0] + '_frame-%04d.png'
print (exestring)
subprocess.run(exestring, shell=True)

frameList = glob.glob('*.*png')
print (frameList)

# Generate timestamps from MOV file.

exestring = 'ffprobe -select_streams v:0 -show_entries packet=pts_time,flags -of csv=print_section=0 "' + \
+ sys.argv[1] + '" > "' + sys.argv[1].split('.')[0] + '_timestamps.txt"'
print (exestring)
subprocess.run(exestring, shell=True)

f = open(sys.argv[1].split('.')[0] + "_timestamps.txt", 'r')
lines = f.readlines()
f.close()

#print (lines)

keyTimestamps = []
for line in lines:
    linesplit = line.split(',')
    print (linesplit)
    if linesplit[1] == 'K\n':
        keyTimestamps.append(float(linesplit[0]))

#print (keyTimestamps)

```

Figure E.1 ROV Frame Extraction and Timestamp Script

---

```

# Find the start of the video file.
"""
exestring = 'ffprobe -v quiet -select_streams v:0 -show_entries stream_tags=creation_time -of default=noprint_wrappers=1:nokey=1 ' + sys.argv[1]
result = subprocess.run(exestring, shell=True, stdout=subprocess.PIPE, text=True)
videoStartTime = result.stdout.split('.')[0]

# Need to convert this to the proper time zone. For whatever reason, ffprobe thinks this is already
# in GMT.

# 2019-05-22T17:40:12Z

tstruct = time.strptime(videoStartTime, "%Y-%m-%dT%H:%M:%S")
ttemp = time.mktime(tstruct)
tgmt = time.gmtime(ttemp)
videoStartTime = time.strftime("%Y-%m-%dT%H:%M:%S", tgmt) + 'Z'
"""
#videoStartTime = sys.argv[3]
videoStartTime = "2019-06-27T14:14:06Z"
tstruct = time.strptime(videoStartTime, "%Y-%m-%dT%H:%M:%SZ")
unixtime = time.mktime(tstruct)
videoStartTime = unixtime
print (videoStartTime)
#pdb.set_trace()

# Now open the CSV file.

f = open(sys.argv[2], 'r')
lines = f.readlines()
f.close()

# Get line indices.

header = lines[0].rstrip('\r\n').split(',')
#pdb.set_trace()
#startVideo = header.index(" CAMERA.isVideo")
gpsTime = header.index("UNIXTIME")
latIndex = header.index("LAT_DD")
lonIndex = header.index("LON_DD")
altIndex = header.index("ALT")
depthIndex = header.index("DEPTH")
#flyTime = header.index("UNIXTIME")
#rollIndex = header.index("IMU_ATT(0):roll:C")
#pitchIndex = header.index("IMU_ATT(0):pitch:C")
#yawIndex = header.index("IMU_ATT(0):yaw:C")

# Look for start of video.

```

Figure E.1 ROV Frame Extraction and Timestamp Script Continued

---

```

# Look for start of video.

outputList = []

videoFlag = False
lineSubset = lines[1:len(lines)]
#print (lineSubset)

keyIndex = 0

for line in lineSubset:
    lsplit = line.rstrip('r\n').split(',')
    if videoFlag == False:
        #pdb.set_trace()
        #print(float(lsplit[gpsTime]))
        if math.floor(float(lsplit[gpsTime])) == videoStartTime:
            startTime = float(lsplit[gpsTime])
            videoFlag = True
            previousDelta = 10000000
            #gpsline = lsplit[latIndex] + ',' + lsplit[lonIndex] + ',' + lsplit[altIndex] + '\n'
            #outputList.append(gpsline)
    else:
        # Now compute and compare delta values. Have start time.
        currentTime = math.floor(float(lsplit[gpsTime])) - startTime
        # Step through image timestamps.
        #print (currentTime)

        try:
            currentDelta = currentTime - keyTimestamps[keyIndex]
        except:
            print ("Out of data?")
            break
        #print (currentDelta)
        #pdb.set_trace()
        if abs(currentDelta) <= abs(previousDelta):
            previousDelta = currentDelta
        else:
            # We have our closest entry, hopefully.
            # Convert altitude from feet to meters.
            try:
                alt = float(lsplit[altIndex])
            except:
                alt = 0
            try:
                depth = float(lsplit[depthIndex])
            except:
                depth = 0

```

---

Figure E.1 ROV Frame Extraction and Timestamp Script Continued

```

        #outputList.append(gpsline)
    else:
        # Now compute and compare delta values. Have start time.
        currentTime = math.floor(float(lsplit[gpsTime])) - startTime
        # Step through image timestamps.
        #print (currentTime)

        try:
            currentDelta = currentTime - keyTimestamps[keyIndex]
        except:
            print ("Out of data?")
            break
        #print (currentDelta)
        #pdb.set_trace()
        if abs(currentDelta) <= abs(previousDelta):
            previousDelta = currentDelta
        else:
            # We have our closest entry, hopefully.
            # Convert altitude from feet to meters.
            try:
                alt = float(lsplit[altIndex])
            except:
                alt = 0
            try:
                depth = float(lsplit[depthIndex])
            except:
                depth = 0
            #pitch = float(lsplit[pitchIndex]) + 90.0
            #gpsline = frameList[keyIndex] + ',' + lsplit[latIndex] + ',' + lsplit[lonIndex] + ',' + str(alt) \
            # + ',' + lsplit[rollIndex] + ',' + str(pitch) + ',' + lsplit[yawIndex] + '\n'
            gpsline = frameList[keyIndex] + ',' + lsplit[latIndex] + ',' + lsplit[lonIndex] + ',' + str(alt) \
                + ',' + str(depth) + '\n'
            outputList.append(gpsline)
            keyIndex = keyIndex + 1
            previousDelta = 10000000

        #pdb.set_trace()
    print (outputList)
    print (len(outputList))

    #if not os.path.exists(sys.argv[1].split('.')[0] + '_gps.txt'):
    #    f = open(sys.argv[1].split('.')[0] + '_gps.txt', 'w')

    f = open(sys.argv[1].split('.')[0] + '_gps.txt', 'w')

    for line in outputList:
        f.write(line)
    f.close()

```

Figure E.1 ROV Frame Extraction and Timestamp Script Continued

---

```

import os, sys
import subprocess
import glob
import pdb
import time

# Extract key frames from MOV file.

exestring = 'ffmpeg -i ' + sys.argv[1] + ' -vf "select=eq(pict_type,I)" -vsync vfr ' \
+ sys.argv[1].split('.')[0] + '_frame-%04d.png'
print (exestring)
subprocess.run(exestring, shell=True)

frameList = glob.glob('*.png')
print (frameList)

# Generate timestamps from MOV file.

exestring = 'ffprobe -select_streams v:0 -show_entries packet=pts_time,flags -of csv=print_section=0 "' \
+ sys.argv[1] + '" > "' + sys.argv[1].split('.')[0] + '_timestamps.txt"'
print (exestring)
subprocess.run(exestring, shell=True)

f = open(sys.argv[1].split('.')[0] + "_timestamps.txt", 'r')
lines = f.readlines()
f.close()

#print (lines)

keyTimestamps = []
for line in lines:
    linesplit = line.split(',')
    print (linesplit)
    if linesplit[1] == 'K\n':
        keyTimestamps.append(float(linesplit[0]))

#print (keyTimestamps)

# Find the start of the video file.

exestring = 'ffprobe -v quiet -select_streams v:0 -show_entries stream_tags=creation_time -of default=noprint_wrappers=1:nokey=1 ' + sys.argv[1]
result = subprocess.run(exestring, shell=True, stdout=subprocess.PIPE, text=True)
videoStartTime = result.stdout.split('.')[0]

# Need to convert this to the proper time zone.

```

---

Figure E.2 North Farm sUAS Frame Extraction and Timestamp Script



---

```

# Need to convert this to the proper time zone.

# 2019-05-22T17:40:12Z

#tstruct = time.strptime(videoStartTime, "%Y-%m-%dT%H:%M:%S")
#ttemp = time.mktime(tstruct)
#tgmt = time.gmtime(ttemp)
#videoStartTime = time.strftime("%Y-%m-%dT%H:%M:%S", tgmt) + 'Z'

print (videoStartTime)
#pdb.set_trace()

# Now open the CSV file.

f = open(sys.argv[2], 'r')
lines = f.readlines()
f.close()

# Get line indices.

header = lines[0].split(',')
#pdb.set_trace()
#startVideo = header.index(" CAMERA.isVideo")
gpsTime = header.index("GPS:dateTimeStamp")
latIndex = header.index("GPS:Lat")
lonIndex = header.index("GPS:Long")
altIndex = header.index("GPS:heightMSL")
flyTime = header.index("Clock:offsetTime")
rollIndex = header.index("IMU_ATT(0):roll:C")
pitchIndex = header.index("IMU_ATT(0):pitch:C")
yawIndex = header.index("IMU_ATT(0):yaw:C")

# Look for start of video.

outputList = []

videoFlag = False
lineSubset = lines[1:len(lines)]
#print (lineSubset)

keyIndex = 0

for line in lineSubset:
    lsplit = line.split(',')
    if videoFlag == False:
        #pdb.set_trace()

```

Figure E.2 North Farm sUAS Frame Extraction and Timestamp Script Continued

```

keyIndex = 0
for line in lineSubset:
    lsplit = line.split(',')
    if videoFlag == False:
        #pdb.set_trace()
        if lsplit[gpsTime] == videoStartTime:
            startTime = float(lsplit[flyTime])
            videoFlag = True
            previousDelta = 10000000
            #gpsline = lsplit[latIndex] + ',' + lsplit[lonIndex] + ',' + lsplit[altIndex] + '\n'
            #outputList.append(gpsline)
        else:
            # Now compute and compare delta values. Have start time.
            currentTime = float(lsplit[flyTime]) - startTime
            # Step through image timestamps.
            #print (currentTime)

            try:
                currentDelta = currentTime - keyTimestamps[keyIndex]
            except:
                print ("Out of data?")
                break
            #print (currentDelta)
            #pdb.set_trace()
            if abs(currentDelta) <= abs(previousDelta):
                previousDelta = currentDelta
            else:

                # Convert altitude from feet to meters.
                alt = float(lsplit[altIndex])
                pitch = float(lsplit[pitchIndex]) + 90.0
                gpsline = frameList[keyIndex] + ',' + lsplit[latIndex] + ',' + lsplit[lonIndex] + ',' + str(alt) \
                    + ',' + lsplit[rollIndex] + ',' + str(pitch) + ',' + lsplit[yawIndex] + '\n'
                outputList.append(gpsline)
                keyIndex = keyIndex + 1
                previousDelta = 10000000

            #pdb.set_trace()
print (outputList)
print (len(outputList))

#if not os.path.exists(sys.argv[1].split('.')[0] + '_gps.txt'):
#    f = open(sys.argv[1].split('.')[0] + '_gps.txt', 'w')

f = open(sys.argv[1].split('.')[0] + '_gps.txt', 'w')

```

Figure E.2 North Farm sUAS Frame Extraction and Timestamp Script Continued

---

```

import os, sys
import subprocess
import glob
import pdb
import time

# Extract key frames from MOV file.

# Fixed interval version.

exestring = 'ffmpeg -i ' + sys.argv[1] + ' -vf fps=3/1,showinfo ' \
    + sys.argv[1].split('.')[0] + '_frame-%04d.png > templog.txt 2>&1'
print (exestring)
subprocess.run(exestring, shell=True)

frameList = glob.glob('*.png')
print (frameList)

# Generate timestamps from MOV file.

# Fixed interval version.
f = open('templog.txt', 'r')
lines=f.readlines()
f.close()

f = open(sys.argv[1].split('.')[0] + "_timestamps.txt", 'w')

for line in lines:
    list = line.split(' ')
    for entry in list:
        try:
            ptsindex = entry.index('pts_time')
            ptstime = entry.split(':')[1]
            #print (ptstime)
            f.write(ptstime + '\n')
        except:
            pass

f.close()

f = open(sys.argv[1].split('.')[0] + "_timestamps.txt", 'r')
lines = f.readlines()
f.close()

#print (lines)

```

---

Figure E.3 Mississippi Gulf Coast sUAS Frame Extraction and Timestamp Script

```

#print (lines)

keyTimestamps = []
for line in lines:
    linesplit = line.split(',')
    print (linesplit)
    #if linesplit[1] == 'K \n':
    keyTimestamps.append(float(linesplit[0]))

#print (keyTimestamps)

# Find the start of the video file.

exestring = 'ffprobe -v quiet -select_streams v:0 -show_entries stream_tags=creation_time -of default=noprint_wrappers=1:nokey=1 ' + sys.argv[1]
result = subprocess.run(exestring, shell=True, stdout=subprocess.PIPE, text=True)
videoStartTime = result.stdout.split('.')[0]

#Converts to the proper time zone.

# 2019-05-22T17:40:12Z

tstruct = time.strptime(videoStartTime, "%Y-%m-%dT%H:%M:%S")
ttemp = time.mktime(tstruct)
tgmt = time.gmtime(ttemp)
videoStartTime = time.strftime("%Y-%m-%dT%H:%M:%S", tgmt) + 'Z'

print (videoStartTime)
#pdb.set_trace()

# Now open the CSV file.

f = open(sys.argv[2], 'r')
lines = f.readlines()
f.close()

# Get line indices.

header = lines[0].split(',')
#pdb.set_trace()
#startVideo = header.index(" CAMERA.isVideo")
gpsTime = header.index("GPS:dateTimeStamp")
latIndex = header.index("GPS:Lat")
lonIndex = header.index("GPS:Long")
altIndex = header.index("GPS:heightMSL")
flyTime = header.index("Clock:offsetTime")
rollIndex = header.index("IMU ATT(0):roll:C")

```

---

Figure E.3 Mississippi Gulf Coast sUAS Frame Extraction and Timestamp Script Continued

---

```

# Look for start of video.

outputList = []

videoFlag = False
lineSubset = lines[1:len(lines)]
#print (lineSubset)

keyIndex = 0

for line in lineSubset:
    lsplit = line.split(',')
    if videoFlag == False:
        #pdb.set_trace()
        if lsplit[gpsTime] == videoStartTime:
            startTime = float(lsplit[flyTime])
            videoFlag = True
            previousDelta = 10000000
            #gpsline = lsplit[latIndex] + ',' + lsplit[lonIndex] + ',' + lsplit[altIndex] + '\n'
            #outputList.append(gpsline)
    else:
        # Now compute and compare delta values. Have start time.
        currentTime = float(lsplit[flyTime]) - startTime
        # Step through image timestamps.
        #print (currentTime)

        try:
            currentDelta = currentTime - keyTimestamps[keyIndex]
        except:
            print ("Out of data?")
            break
        #print (currentDelta)
        #pdb.set_trace()
        if abs(currentDelta) <= abs(previousDelta):
            previousDelta = currentDelta
        else:
            # Convert altitude from feet to meters.
            alt = float(lsplit[altIndex])
            pitch = float(lsplit[pitchIndex]) + 90.0
            gpsline = frameList[keyIndex] + ',' + lsplit[latIndex] + ',' + lsplit[lonIndex] + ',' + str(alt) \
                + ',' + lsplit[rollIndex] + ',' + str(pitch) + ',' + lsplit[yawIndex] + '\n'
            outputList.append(gpsline)
            keyIndex = keyIndex + 1
            previousDelta = 10000000

        #pdb.set_trace()
print (outputList)

```

---

Figure E.3      Mississippi Gulf Coast sUAS Frame Extraction and Timestamp Script Continued