UNIVERSITY OF CALIFORNIA SAN DIEGO

3D Capture for Remote Environments

A dissertation submitted in partial satisfaction of the
requirements for the degree
Doctor of Philosophy

in

Computer Science (Computer Engineering)

by

Peter Tueller

2022

University of California San Diego

2022

TABLE OF CONTENTS

Dissertation Approval Page . . . iii

Table of Contents . . . iv

List of Figures . . . vi

List of Tables . . . viii

Acknowledgements . . . ix

Vita . . . x

Abstract of the Dissertation . . . xii

Chapter 1 Introduction . . . 1

Chapter 2 3D Data Capture . . . 4
- 2.1 Introduction . . . 5
- 2.2 Synthetic Aperture Sonar . . . 5
- 2.3 LiDAR . . . 8
- 2.4 RGBD Cameras . . . 9
  - 2.4.1 Time of Flight . . . 9
  - 2.4.2 Stereo . . . 10
  - 2.4.3 Structured Light . . . 11
- 2.5 Photogrammetry and SfM . . . 12

Chapter 3 Target Detection using Features for Sonar Images . . . 13
- 3.1 Introduction . . . 13
- 3.2 Related Work . . . 18
- 3.3 Feature Based Target Detection . . . 19
  - 3.3.1 System Model and Key Idea . . . 19
  - 3.3.2 Feature Extraction . . . 20
  - 3.3.3 Feature Classification . . . 23
  - 3.3.4 Identifying Seabed Structure . . . 25
  - 3.3.5 Target Detection . . . 27
- 3.4 Design Space Exploration . . . 28
  - 3.4.1 Exploring the Space of Parameters . . . 28
  - 3.4.2 Utilization Function for Parameter Exploration . . . 30
- 3.5 Results . . . 32
  - 3.5.1 Simulations . . . 32
  - 3.5.2 Classification Results . . . 33
  - 3.5.3 Comparing Feature Extraction Algorithms . . . 34

3.5.4 Real Sonar Images 38
3.6 Conclusion 39
3.7 Acknowledgments 41

Chapter 4 3D Reconstructions of Ancient Mayan Archaeological Excavations 42
4.1 Introduction 43
4.2 Sensing for Archaeology 44
4.2.1 LiDAR 46
4.2.2 Photogrammetry 46
4.2.3 Simultaneous Localization And Mapping with RGBD Sensors 47
4.3 Multi-Sensor Registration 49
4.3.1 Acquisition 50
4.3.2 Software 51
4.3.3 Game Engines 53
4.3.4 VR Devices 54
4.4 Results 54
4.4.1 Arroyo Tapiado Mud Caves 54
4.4.2 El Zotz Archaeological Site 61
4.4.3 Virtual Reality Model 64
4.5 SLAM Results Analysis 65
4.5.1 Sensors 65
4.5.2 Algorithms 65
4.5.3 Scanning Procedure 66
4.6 Conclusion 67
4.7 Acknowledgments 68

Chapter 5 FishSense: RGBD Imaging for Underwater Fish Detection and Assessment 69
5.1 Introduction 70
5.2 Platform 72
5.2.1 Mechanical 73
5.2.2 Electrical 73
5.2.3 Intel RealSense D455 74
5.3 Fish Detection and Measurement 75
5.4 Analysis 76
5.5 Conclusion 79
5.6 Acknowledgments 80

Chapter 6 Conclusion 81

Bibliography 83

LIST OF FIGURES

Figure 2.1: Generic workflow for utilizing 3D data capture techniques for remote environment observation. 5
Figure 2.2: Absorption coefficient of pure seawater for different transmission wavelengths. 6
Figure 2.3: Side scan sonar vs synthetic aperture sonar. 7
Figure 2.4: SAS image of the sunken USS Murphy. 7
Figure 2.5: Leica BLK360 LiDAR scan. 8
Figure 2.6: RGB (left) and depth (right) images. 9
Figure 2.7: Intel RealSense D455 RGBD camera. 11
Figure 2.8: Operation of structured light. 12

Figure 3.1: Sonar image with heterogenous seabed structures – sand, sand ripples, and Posidonea plant life. 14
Figure 3.2: Synthetic Aperture Sonar (SAS) image of the seafloor with an object appearing prominently in the center right. 15
Figure 3.3: General framework for our feature-based detection algorithm. 21
Figure 3.4: Feature classification. The first flow chart represents training an SVM model, and the second flow chart represents performing classification using that model. 23
Figure 3.5: Placement of centroids on simulated sonar images. Features are displayed as cyan points, and the two centroids are the green plus symbols. Features associated with an object (left) will on average be closer to a centroid than features not associated with an object (right). 25
Figure 3.6: Simulated sonar images with grass (left), ripple (middle), and sand (right) seabeds. 26
Figure 3.7: Example for a Design Space Exploration. The red line shows the true set of optimal designs, and the black line shows what was estimated utilizing only the magenta points. 30
Figure 3.8: Three detectors' ROCs for a single design 31

Figure 4.1: Interior of m7-1. 44
Figure 4.2: Exterior of m7-1 45
Figure 4.3: Using Fine Registration to extend the virtual excavation with point cloud data from 2015, 2017, and 2019 field expeditions. 46
Figure 4.4: Left-Center: CSF Filter Point Cloud Results from m7-1 exterior perimeter. Right: Meshed Game Asset in Unity 47
Figure 4.5: Spider Monkey Bowl found in m7-1. From left: photographs taken back at camp. Center: Agisoft Photoscan reconstruction output, Right: Final Game Asset inside Unreal Engine. 48
Figure 4.6: The 7 UDIM Sub-objects making up the model were further segmented for a total 21 parts represented by each color. 54

Figure 4.7: The highlighted areas require the most geometric detail and were separated from the overall mesh to preserve them during the decimation process. . . . 55
Figure 4.8: From top to bottom: The LiDAR scan of the mud cave, the SLAM scan using RTAB-MAP and Kinect v2, and the same SLAM scan with point-to-point error with the LiDAR scan highlighted (darker shade of red = larger error). The scale indicates the error in meters. 56
Figure 4.9: RMSE of the cloud-to-cloud difference between SLAM algorithm results and the LiDAR cloud, for the mud caves site. 56
Figure 4.10: Trajectory comparison between RTAB-MAP with Kinect v1 and the recovered ground truth. The position error is plotted on the SLAM trajectory in meters. 58
Figure 4.11: Comparison of trajectories with and without loop closure using RTAB-MAP with Kinect v1. The trajectory is plotted in meters for each axis (the z axis is aligned to gravity), against the camera frame index. 60
Figure 4.12: Point-to-point difference between the point cloud from ORB-SLAM with Kinect v1 and the LiDAR point cloud. Darker red corresponds to a larger error, which happens in areas with concentrated sensor noise, and areas where the two models differ in geometry. 62
Figure 4.13: RMSE of the cloud-to-cloud difference between SLAM algorithm results and the LiDAR cloud, for the M7-1 excavation site. 62
Figure 4.14: Top view of the trajectory comparison between SLAM results and the recovered ground truth. The position error is plotted on the SLAM trajectory in meters. 63
Figure 4.15: The Final m7-1 model composed of several sub-objects 64
Figure 4.16: Resulting Game Asset for the Maya Archaeology VR Application utilizing Physical-based materials and Optimized Geometry in Unreal Engine. . . . 64

Figure 5.1: FishSense imaging platform 71
Figure 5.2: FishSense mechanical prototype cross-section 72
Figure 5.3: FishSense wiring diagram 74
Figure 5.4: Process of data interpretation 75
Figure 5.5: ROC curves for three models over two pool tests 77
Figure 5.6: Sample images from the Pool 1 (top) and Pool 2 (bottom) datasets 77

LIST OF TABLES

Table 3.1: Comparison of feature detectors 22

Table 3.2: In the majority of cases, all optimized curves perform better than the default curves (in black). The curves optimized for a particular background perform at least as well, but typically significantly better than curves optimized for other backgrounds. 35

Table 3.3: ROC Curves Continued 36

Table 3.4: ADRS values for each feature detector. The dataset with the designs closest to each optimal front is in bold. This is a numerical representation of the plots in Table 3.2, and support the conclusions drawn from visual analysis. 37

Table 3.5: Target Detection Methods on Real Sonar Images with the SURF detector. Methods 2 and 3 reduce the number of false positives. 40

Table 4.1: Absolute Trajectory Error (ATE) and Relative Position Error (RPE) on different combinations of sensors and algorithms. *ATE* compares the trajectories aligned on the first pose only. *ATE (aligned)* compares trajectories globally aligned. *RPE* measures the relative error per 1 meter segments. 58

Table 4.2: ATE and RPE in meters, on the M7-1 excavation. 63

ACKNOWLEDGEMENTS

I would like to acknowledge Professor Ryan Kastner for his support as the chair of my committee. His guidance has shaped this dissertation, the way I approach research, and my career as an engineer.

Chapter 3, in full, is a reprint of the material as it appears in IET Radar, Sonar, & Navigation 2020. Tueller, Peter; Kastner, Ryan; Diamant, Roee. The dissertation author was the primary investigator and author of this paper.

Chapter 4, in part is currently being prepared for submission for publication of the material. Tueller, Peter; Vindiola, Giovanni; Garrison, Thomas; Schurgers, Curt; Kastner, Ryan. The dissertation/thesis author was the primary investigator and author of this material. Chapter 4, also in part, is a reprint of the material as it appears the Journal of Cultural Heritage Management and Sustainable Development 2020. Gautier, Quentin; Garrison, Thomas; Rushton, Ferrill; Bouck, Nicholas; Lo, Eric; Tueller, Peter; Schurgers, Curt; Kastner, Ryan.

Chapter 5, in part, is a reprint of the material as it appears in IEEE/MTS OCEANS 2021. Tueller, Peter; Maddukuri, Raghav; Paxson, Patrick; Suresh, Vivaswat; Ashok, Arjun; Bland, Madison; Wallace, Ronana; Guerrero, Julia; Semmens, Brice; Kastner, Ryan. The dissertation/thesis author was the primary investigator and author of this material. Chapter 5, also, in part is currently being prepared for submission for publication of the material. Tueller, Peter; Maddukuri, Raghav; Paxson, Patrick; Suresh, Vivaswat; Miao, Albert; Drews, Donovan; Paxson, Charles; Semmens, Brice; Kastner, Ryan. “FishSense: 3D Capture and Analysis for Fish Detection and Stock Assessment”.

VITA

2016	Bachelor of Science in Engineering, Computer Systems Engineering Arizona State University
2018	Master of Science, Computer Science (Computer Engineering) University of California San Diego
2019-2022	SMART Scholar Sponsoring Facility: Naval Information Warfare Center Pacific
2022	Doctor of Philosophy, Computer Science (Computer Engineering) University of California San Diego

PUBLICATIONS

Perry Naughton, Tahiya Salam, Peter Tueller, Philippe Roux, Curt Schurgers, Ryan Kastner, Jules Jaffe, Paul Roberts, "Self-synchronization of multiple vehicles using ambient impulsive noise", *2018 Fourth Underwater Communications and Networking Conference (UComms)*, 1-5, 2018.

Peter Tueller, Ryan Kastner, Roee Diamant, "A Comparison of Feature Detectors for Underwater Sonar Imagery", *OCEANS 2018 MTS/IEEE Charleston*, 1-6, 2018.

Michael Barrow, Nelson Ho, Alric Althoff, Peter Tueller, Ryan Kastner, "Benchmarking Video With The Surgical Image Registration Generator (SIRGn) Baseline", *ISVC 2019*, 2019.

Quentin Kevin Gautier, Thomas G Garrison, Ferrill Rushton, Nicholas Bouck, Eric Lo, Peter Tueller, Curt Schurgers, Ryan Kastner, "Low-cost 3D scanning systems for cultural heritage documentation", *Journal of Cultural Heritage Management and Sustainable Development*, 2020.

Peter Tueller, Ryan Kastner, Roee Diamant, "Target detection using features for sonar images", *IET Radar, Sonar & Navigation*, 1940-1949, 2020.

Alex Yen, Bryse Flowers, Wenshan Luo, Nitish Nagesh, Peter Tueller, Ryan Kastner, Pat Pannuto, "A UCSD view on replication and reproducibility for CPS & IoT", *Proceedings of the Workshop on Benchmarking Cyber-Physical Systems and Internet of Things*, 20-25, 2021.

Peter Tueller, Raghav Maddukuri, Patrick Paxson, Vivaswat Suresh, Arjun Ashok, Madison Bland, Ronan Wallace, Julia Guerrero, Brice Semmens, Ryan Kastner, "FishSense: Underwater RGBD Imaging for Fish Measurement", *OCEANS 2021 MTS/IEEE San Diego*, 1-5, 2021.

Peter Tueller, Giovanni Vindiola, Thomas Garrison, Curt Schurgers, Ryan Kastner,"3D Capture and Virtual Reality for Cultural Heritage Documentation", *Journal of Cultural Heritage Management and Sustainable Development*, preparing for submission.

Peter Tueller, Raghav Maddukuri, Patrick Paxson, Vivaswat Suresh, Albert Miao, Donovan Drews, Charles Paxson, Brice Semmens, Ryan Kastner, “FishSense: Underwater 3D Capture for Fish Measurement and Stock Assessment”, *IEEE Journal of Oceanic Engineering*, preparing for submission.

ABSTRACT OF THE DISSERTATION

3D Capture for Remote Environments

by

Peter Tueller

Doctor of Philosophy in Computer Science (Computer Engineering)

University of California San Diego, 2022

Professor Ryan Kastner, Chair

There are many environments on Earth that are so remote that they are inhospitable to humans and conventional sensing equipment. Yet, these environments can hold information of ecological and cultural significance that cannot be gathered anywhere else. Current methods of gathering information in these environments give an important window, but utilizing modern sensors to capture 3D information can allow us to interpret existing data and understand the environments in new and unique ways. This thesis will demonstrate how 3D capture can improve data collection and interpretation in three separate remote environments. First, I will show how Synthetic Aperture Sonar on autonomous underwater vehicles paired with optimized feature detectors can improve target detection and seafloor recognition. Next, I will show how RGBD cameras, photogrammetry, and LiDAR can be used in isolated Guatemalan archaeological excavations to visualize and contextualize ancient sites in relation to each other and to our broader understanding of Mayan history. Finally, I will demonstrate the effectiveness and potential of RGBD cameras for fish stock assessment through detection and length and biomass measurement in open waters and in aquaculture.

Chapter 1

Introduction

There are corners of this Earth that hold secrets and information vital to improving our understanding and appreciation and ability to preserve the planet. By observing and measuring these remote environments, we can estimate the health of the planet and the sustainability of current practices. For example, being able to track and measure the health of fish populations in the ocean or within aquaculture farms is important for understanding how much of that population can be harvested so overfishing does not occur. Also, these remote environments can yield culturally significant information, such as in the case of uncovering and understanding ancient structures or shipwrecks, and can yield important information regarding safety, such as detecting and neutralizing mines on the seafloor.

These corners need to be observed and information needs to be gathered somehow, but traditional techniques and equipment are often insufficient. Large equipment cannot be easily brought into jungles or maneuvered around tight enclosed spaces such as caves, and underwater environments prohibit the use of much of the electromagnetic spectrum, in addition to the ever present water pressure, particulate matter, and lack of light.

There is a vast amount of research on techniques for gathering information in these inhospitable environments. Some research seeks to improve the physical sensors themselves;

from improving the resolution and scope of acquired data, to the usability and functionality in the field. Additional research improves the data processing methods for gleaning the most information from the acquired data, whether this be through procedures for cleaning and registering samples, or machine learning-based detection.

This paper focuses on the use of 3D sensors and capture techniques for remote environments, as well as the methods to appropriately process and analyze 3D data that will complement and improve upon current state-of-the-art techniques. We focus on 3D capture in three remote environments. First, we consider target detection on the seafloor using autonomous underwater vehicles (AUVs) and synthetic aperture sonar (SAS). Next, we consider imaging and 3D reconstruction of ancient Mayan temple excavations in northern Guatemala for archaeological analysis and accessible presentation through virtual reality (VR). Finally, we consider using RGBD cameras for fish detection, length and biomass measurement, and population assessment in open ocean environments as well as aquaculture farms.

There are a number of contributions of this work regarding the use of 3D sensors in these environments. We list the contributions here in order:

- A method for feature-based target detection on synthetic aperture sonar imagery that takes into account the spatial identification of a target, as well as the type of background;
- An implementation of an active learning design space exploration method to optimize feature detector parameters on sonar imagery;
- A comparison of the relative performances of feature detectors on a simulated synthetic aperture sonar dataset;
- A comparison of 3D capture techniques for archaeological tunnel excavations to evaluate most accurate and most usable techniques for virtual representation;
- A workflow for representing archaeological tunnel excavations using data from 3D sensors

that is more complete and accurate than workflows using single sensors;

- An exploration of virtual reality as a means for communicating and visualizing archaeological excavations;
- Development of a low-profile diver-operated platform for free-swimming fish imaging;
- An implementation of a machine learning detector for identifying fish in RGBD images;
- An implementation of a fish length measuring algorithm for RGBD images.

The remainder of this paper is organized as follows. Chapter 2 presents an overview of specific 3D sensors and their benefits and drawbacks in remote environments. It also presents a generic workflow for utilizing and making the most out of 3D capture techniques. Chapter 3 presents the use of SAS imagery as it pertains to target detection on the seafloor. This chapter additionally presents an analysis of the use of feature detectors and an active learning design space exploration method for optimizing this detection. Chapter 4 presents the use of LiDAR (both ground-based and aerial), photogrammetry and structure for motion (SfM), and RGBD cameras for reconstructing archaeological excavations and presenting the reconstructions in a modern, intuitive, and accessible manner. Chapter 5 outlines a RGBD camera-based underwater imaging platform and processing pipeline for detecting and measuring fish in open ocean environments and in aquaculture farms. Finally, Chapter 6 concludes the paper and summarizes the results.

Chapter 2

3D Data Capture

When capturing remote environments, we use a generic workflow as presented in Figure 2.1. The first step represents the quantification of physical attributes of the environment through the use of 3D sensors. The second step takes each return from a particular sensor and combines them into a coherent view of the scene as a whole. The third step incorporates additional information from other sensors about the environment to modify the resulting views from step 2. This information could be additional views of the scene, 3D or not, as well as information about location, geography, or color, for example. Finally, the fourth step takes the view of the environment with all the required context and interprets it and performs all the requisite measurements and analyses.

Remote environments can be challenging in unique ways, but this generic workflow allows us the flexibility to use the sensor best suited for the environment, while still resulting in a useful and meaningful final representation of the data. It also allows for the use of multiple sensors, where the best qualities of each are incorporated, and a view of the environment at a scope and resolution greater than each of the sensors is possible. This particular aspect will be portrayed in more detail in Chapter 4.

Each of the three environments that are investigated by this paper will follow this pipeline,

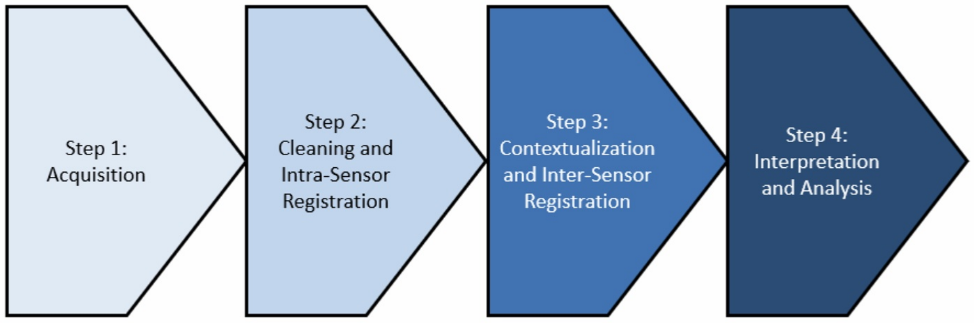


Figure 2.1: Generic workflow for utilizing 3D data capture techniques for remote environment observation.

and each of the contributions improve one particular aspect of the pipeline. In this chapter, we will outline the 3D sensors that can be used in Step 1 of 2.1.

2.1 Introduction

3D sensors can be a boon to capturing an environment, as there is a whole extra dimension associated with the data when compared to traditional 2D sensors, such as monocular cameras. However, the variety of methods for acquiring this third dimension lead to a wide array of sensors that can be classified as ”3D”, and each one has unique properties that make them useful in different contexts.

2.2 Synthetic Aperture Sonar

A primary goal of an autonomous underwater vehicle (AUV) is to observe large swaths of the seafloor. This is to detect mines, pipelines, shipwrecks or plane wrecks, or to perform bathymetric measurements. Because of the sheer scale of the seafloor, any sensor must have significant range to observe as much as possible. However, effectively sensing underwater is extraordinarily difficult when compared to operating a sensor through air. If a terrestrial or aerial vehicle were to perform similar observation of land, typically its primary sensor would employ optics or radar to collect data. Neither of these sensors are feasible in an underwater environment due to extreme absorption and scattering in the electromagnetic spectrum. Transmissability of microwaves, such as those employed by radar, is particularly low underwater, with an absorption coefficient in the range of 100 cm^{-1}, as seen in Figure 2.2. The absorption coefficient of light in the visible spectrum, which is the range in which cameras would traditionally operate, is significantly better, but the effective range remains on the order of tens of meters in the best case scenario with blue-green light and minimal particulate matter [Hug90].

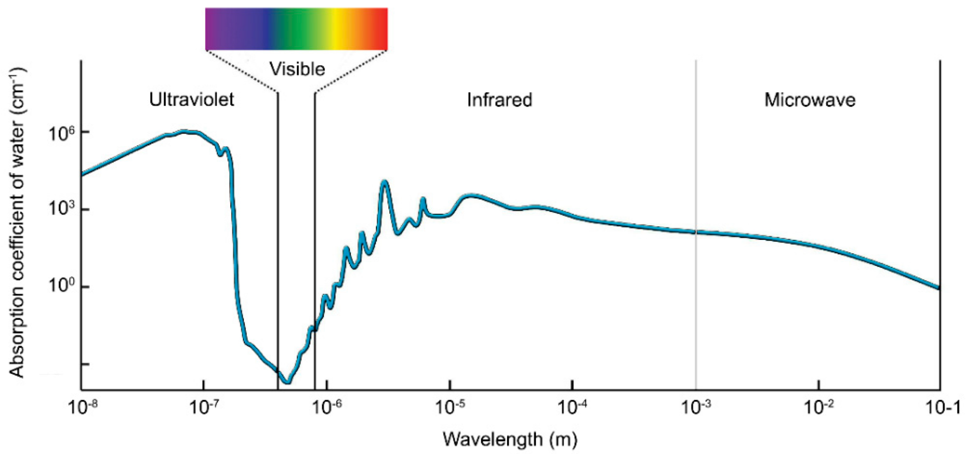


Figure 2.2: Absorption coefficient of pure seawater for different transmission wavelengths. [SSCL20]

On the other hand, sonar technology has the capability to achieve ranges that are more effective for seafloor observation and imaging. By employing sound, which is highly transmissible underwater, some side scan imaging sonars are able to achieve ranges from hundreds to tens of thousands of meters [MHVL11]. The images that are returned by side scan sonars effectively capture a 3D view of the seafloor, as the time it takes for the sound to return correlates to distance from the vehicle. The primary problem with side scan sonar, however, is the azimuth resolution that decreases as the range increases, and as such the resulting image is ”muddy” and it is difficult to effectively interpret and use.

Synthetic aperture sonar (SAS) corrects this problem with range-independent azimuth resolution [HG09]. Instead of sending out a single narrow beam and waiting for its return before continuing on, SAS transmits wider beams continuously and processes all returns simultaneously, as seen in Figure 2.3. This results in a higher resolution return, with the image approaching optical-like quality. An example SAS image can be seen in Figure 2.4.

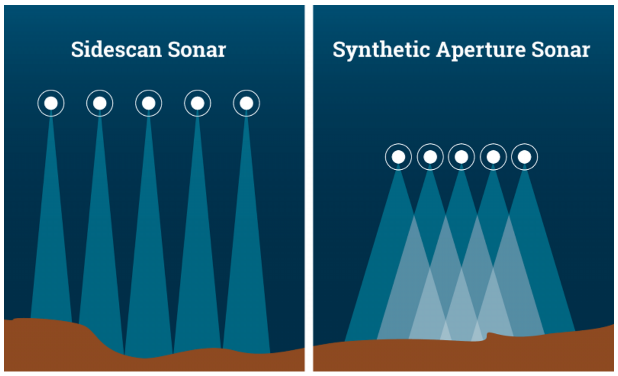


Figure 2.3: Side scan sonar vs synthetic aperture sonar.

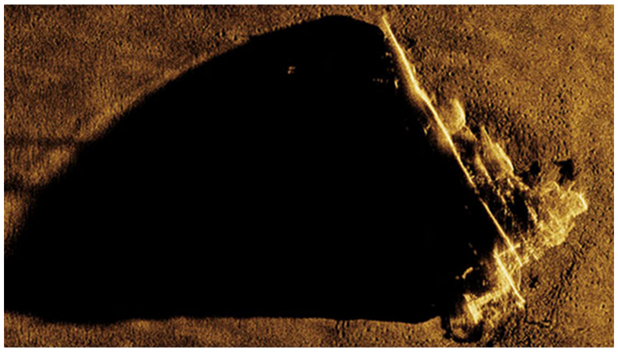


Figure 2.4: SAS image of the sunken USS Murphy.

Although SAS operation is less flexible and more expensive than traditional side scan sonar, the benefits it offers in terms of resolution, and particularly resolution at long range, make it ideal for seafloor imaging and detection of targets, which will be explored in Chapter 3.

2.3 LiDAR

Similar to the operation of sonar, where pings of sound are emitted and the intensity of the return is measured to estimate 3D geometry of the environment, LiDAR emits energy, typically in the infrared portion of the electromagnetic spectrum, and measures the time it takes for pulses of energy to return. LiDARs have various sizes and usages, with range and resolution correlating to their size and power.

For example, the Leica BLK360 Imaging Laser Scanner is easily hand carried and operated, though it must remain stationary while scanning. It has a range from 0.6 to 60m, a horizontal field of view of 360°, a vertical field of view of 300°, and a 3D point accuracy of 6mm at 10m. It collects 360,000 points per second, and a full scan of the environment takes about 3 minutes. It also has an RGB imaging sensor and can align and project images from that sensor onto the points within the point cloud that the LiDAR component generated. As such, it is particularly useful creating high resolution 3D representations of environments from the ground. We use this sensor in Chapter 4 to create scans of the tunnels within the excavations, as well as certain scans on the exterior of the excavation sites. An example of a point cloud gathered by the Leica BLK360 is shown in Figure 2.5.

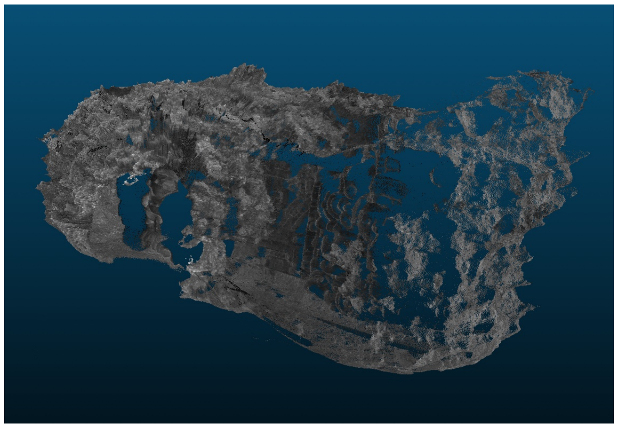


Figure 2.5: Leica BLK360 LiDAR scan from the El Diablo excavation in El Zotz, Petén, Guatemala. Captured June 2021.

2.4 RGBD Cameras

RGBD cameras, another, more modern class of sensor, have seen increasing usage in remote environments. As their name would indicate, they capture RGB images like a traditional optical sensor, but combine it with some sensor that captures depth information, as seen in Figure 2.6. These sensors are used in a similar manner to cameras, and RGB and depth frames together are captured and aligned typically between 10 and 30 frames per second. This is in stark contrast to the previous two sensors described, as SAS imaging and LiDAR scanning collects a single scan on the order of minutes. The RGBD sensors that are described here are also smaller and more mobile and maneuverable. However, the sensors only return frames with 3D information, as opposed to full point clouds from the Leica BLK360, and as such there is a larger burden on the intra-sensor registration (step 2 in Figure WORKFLOW) to create a full 3D representation of the environment.

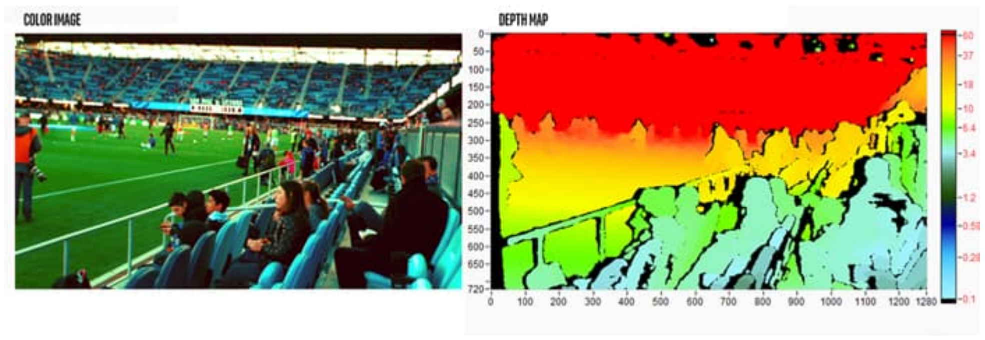


Figure 2.6: RGB (left) and depth (right) images.

2.4.1 Time of Flight

A common type of RGBD camera uses a Time of Flight (ToF) depth sensor. Similar to the operation of a SAS or LiDAR, the sensor emits energy in the infrared spectrum in a particular direction and measures the phase of the returning signal to estimate the time it took to return, and thus the distance along that vector. By doing this emittance and measurement process across the entire field of view, the sensor constructs a depth ”image” of the environment, where each pixel has a corresponding distance associated with it. An example depth image can be seen in Figure 2.6. This image is aligned with the RGB image of the same view of the environment. The depth image then has the color from the RGB image ”painted” onto it to get a full 3D color picture of the scene.

Arguably the most prominent RGBD camera that utilizes ToF technology is the Microsoft Kinect Sensor V2. This sensor has a maximum depth distance of 4.5m, a horizontal field of view of 70°, a vertical field of view of 60°, and returns 1080p RGB images paired with 424p depth images at 30 frames per second.

Another RGBD camera that utilizes ToF technology is the Intel RealSense L515. This camera has an active LiDAR as its depth sensor, with a range of 9m, a horizontal field of view of 70°, a vertical field of view of 55°, and returns 1080p RGB images paired with 768p depth images at 30 frames per second. One of the primary benefits of this particular camera is its small size. At 61mm in diameter and 26mm deep, it is easily handheld, and the use of Intel’s Vision D4 chip for on-board processing means that fully aligned RGB and depth image pairs can be transmitted to the host computer without a need for additional processing. Thus, mobile devices can be used as the host computer with a real-time reconstruction software like Dot3D aligning the RGBD frames together, which further increases the flexibility and portability of this device for sensing in remote environments, such as archaeological excavations, which we will further explore in Chapter 4.

2.4.2 Stereo

Another RGBD camera, the Intel RealSense D455, seen in Figure 2.7, utilizes two passive infrared cameras as a stereo pair to create a depth image. Each image pair from the two infrared cameras are compared and features are matched between them on-board the camera, and with the distance between the cameras being fixed and known, the camera is able to compute a disparity image. This process can be done at 90 frames per second with 720p resolution and with a maximum range of 6m. This depth stream is then combined with the 800p resolution, 30 frames per second RGB stream to achieve an RGBD 3D image with a horizontal field of view of 90°and a vertical field of view of 65°. There is an infrared emitter within the camera, but its only function is to illuminate the scene in the infrared band to make stronger and more accurate feature matching in the depth stream; there is no sensor measuring ToF to establish depth.

Figure 2.7: Intel RealSense D455 RGBD camera.

The passive sensing nature and low-profile of this camera makes it ideal for underwater short-range applications. This aspect will be discussed further in Chapter 5.

2.4.3 Structured Light

One other common method of depth sensing for RGBD cameras is with structured light. Structured light is where a known pattern of light, typically infrared, is projected onto the environment over the field of view of the camera. Then, the imaging sensors measures the deformation of this pattern of light, and thus the 3D geometry of the environment, as seen in Figure 2.8. This 3D information is quantified into a depth image, which is then aligned with the RGB stream, similarly to the previously mentioned RGBD cameras.

One prominent structured light RGBD camera is the first generation of the Microsoft Kinect Sensor. Another such camera is the Intel RealSense SR305 RGBD camera. These cameras were state of the art a decade ago and are still usable, but improvements in computation have allowed stereo and ToF RGBD cameras to surpass structured light RGBD cameras in terms of resolution and accuracy in recent years.

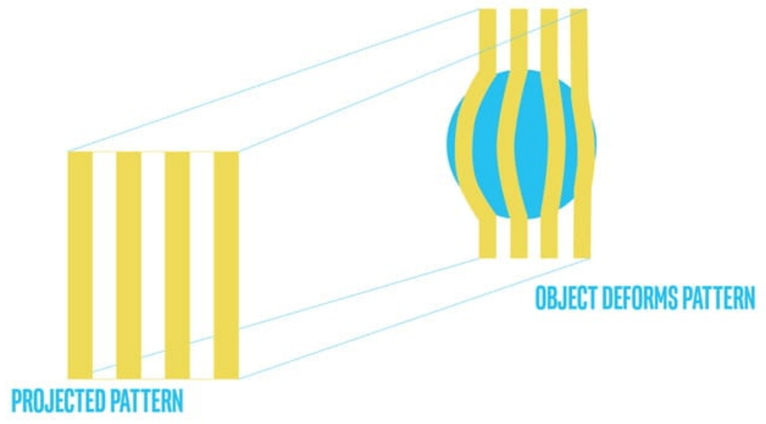


Figure 2.8: Operation of structured light.

2.5 Photogrammetry and SfM

One final method of gathering 3D information about a scene uses the most common sensors of all: monocular RGB cameras. Just as stereo cameras match features between image pairs to create a single disparity image, with monocular cameras we are able to match features between sets of single images to create an estimate of the 3D geometry of the scene. This is known as Structure from Motion (SfM), which is a type of photogrammetry. As the number of images of the scene increase, particularly images that have significant overlap, the geometry of the scene and thus the position of the camera for each image can be estimated. However, in contrast to stereo imaging, the distance between images is not known. This means that even though the geometry may be accurate, it is impossible to know the scale of the reconstruction unless there is are known points of reference within the scene. This is not difficult to do in environments that are easily controlled, as you can place and measure markers within the scene, but in more complex environments, the results from SfM must be correlated with other sensors to achieve proper scaling and allow measurements to be taken. We explore this further in Chapter 4.

Chapter 3

Target Detection using Features for Sonar Images

In this chapter we present and explore a workflow for utilizing 3D data capture to image and detect targets on the seafloor with synthetic aperture sonar (SAS). This workflow follows along the generic workflow presented in Figure 2.1 and we primarily explore the final step in this workflow with the optimization of a target detection algorithm. Steps 1 through 3 in this particular application are handled by the SAS and the computer of the autonomous underwater vehicle (AUV) that controls the SAS. SAS emits and listens for the acoustic beams (step 1), constructs the return of each beam into an image (step 2), and adds in longitude, latitude, and depth information as determined by the navigational sensors on the AUV (step 3).

3.1 Introduction

Sonar imaging is a valuable tool for analyzing the seafloor. Detecting the presence of sunken ships and archaeological sites on the seafloor, mapping structures and objects, or performing pipeline inspection are all extremely useful tasks that are difficult for divers to accomplish, particularly in areas of significant depth. Current synthetic aperture sonar (SAS) imaging systems provide cm scale resolution at ranges of $\approx 100m$, and can be equipped on Autonomous Underwater Vehicles (AUVs) or vessels that remain on the surface, such as ships or Unmanned Surface Vehicles (USVs). Yet, sonar images are heterogeneous by nature, and the design of a fully automated system for object detection that is robust to various seafloor environments and that functions well for different types of targets is challenging. Fig. 3.1 shows a sonar image that contains different seabed structures where Poseidonea, sand, and sand ripples exist in the same image. Each of these have different features and characteristics that make detection challenging.

There are a number of approaches for the task of detection objects in a sonar image. One possibility is using a supervised learning approach that compares regions in the image to previously known patterns that correspond to objects. Barngrover [Bar14] trains a Haar-like feature classifier on a large dataset of objects for mine detection. These approaches are prone to false negatives when there is mismatch between the template and the actual sonar image, e.g., due to changes with altitude and angle of the sonar system with respect to the target. Further, this supervised learning may only correspond to the case when the shape of the target is known a priori.

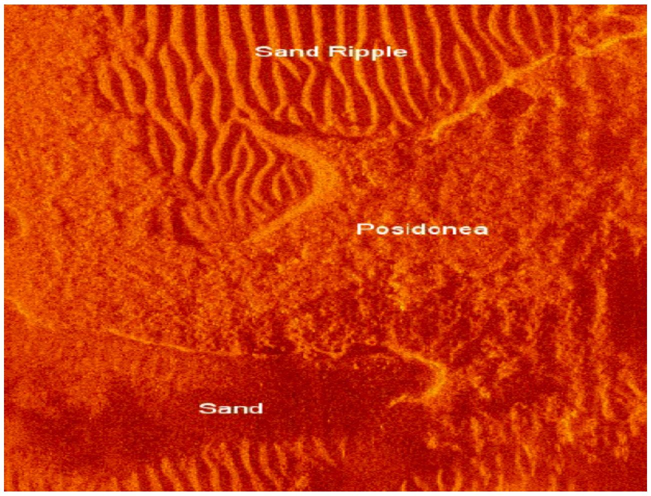


Figure 3.1: Sonar image with heterogenous seabed structures – sand, sand ripples, and Posidonea plant life.

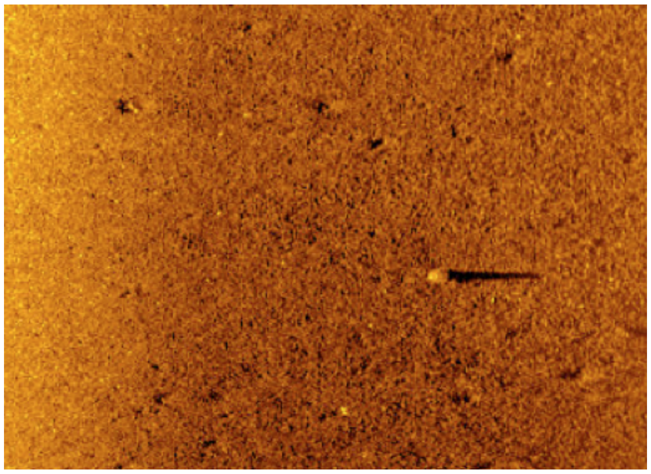


Figure 3.2: Synthetic Aperture Sonar (SAS) image of the seafloor with an object appearing prominently in the center right.

Myers *et al.* [MF10] suggests an alternative which segments a sonar image into five classes based on simple thresholding and compares them to previously established templates. These templates are generated from a database of targets at a variety of orientations. Recently, Abu *et al.* [AR19] abandons supervised learning altogether and proposes a statistical detection approach. Their classifier uses the sonar parameters and on the expected distribution of the highlight, shadow, and background. The result is a detected region-of-interest based on segmentation of highlights and shadow, set by a likelihood ratio.

While the above approaches perform well for their application, they all inherently assume some knowledge about the target, e.g., the shape of the object in template matching or an assumed distribution for the highlight or shadow as in statistical detection methods. Instead, in this work we focus on the case where such information is not available. Our method only relies on the assumption that the object is sufficiently different from its environment. That is, we assume that there exist some features related only to the object to allow us to separate it from its environment. As shown in the example in Fig. 3.2, these objects can manifest as a bright circle against a sandy background or a shadow with complex geometry.

Feature detectors lie at the heart of all indirect image processing techniques. They do not require prior knowledge of the target; instead they aim to identify geometry that is highly distinguishable from the background. As a result, feature detectors fulfill the robustness requirement for handling multiple sea environments and sonar images. Yet, although feature detectors have been successfully applied for the detection of objects in optical images or videos [HCC$^+$15],[LWZ11],[DABP14],[ZN13], there has been little done to explore their effectiveness in the sonar realm.

The two main challenges of feature-based target detection are 1) separating the target from the background and 2) setting up of the feature parameters. Regarding the former, the features extracted from the target and from target-like objects like rocks or ripples are hard to distinguish especially in a diverse seabed environment. Regarding the latter, the number of parameters range from 3 to 6 depending on the method used (see Table 3.1). Yet, the number of values for each parameter has thousands of possibilities. While the default parameters have been set to give good performance with simple images, these default parameters are far from being suitable in the general case [MTM10] and we show that custom parameters are necessary for sonar images. Unfortunately, the brute force exploration of all possible configurations is not feasible. This is exacerbated by the fact that the performance of the feature extraction is highly affected by the unique patterns on the seafloor, such as sand, sand ripples, or grass, as seen in Fig. 3.1.

We introduce a new methodology for feature-based target detection that addresses these two challenges. Our method accounts for the expected spatial separation of the target and the type of background in the sonar image. The method also efficiently explores the parameter space across the various environment backgrounds to find the optimal or near optimal parameter configuration for a given sonar image background type. Our method does not guarantee optimal detection, but rather approaches optimality with low complexity.

Our method fits target identification applications like simultaneous localization and mapping (SLAM) or identification of targets that stand out from their environment, e.g., pipeline exploration, archaeological surveys, and mine detection. The algorithm works by first gathering features across a dataset of sonar images and training a classifier to label each feature as object or non-object.We then modify the parameters of the feature detectors and find the set of best parameters to fit the evaluated seabed type using design space exploration. This solves the problem of needing to hand tune the parameters of the detector, and manually test tens of thousands of configurations to optimize performance. Finally, given a set of features generated by the optimal parameters, we perform target detection by prioritizing detection of distinct features in space to identify the coordinates of single or multiple targets.

Our contributions are:

1. A method for feature-based target detection on sonar imagery that takes into account the spatial identification of a target, as well as the type of background;
2. An implementation of an active learning design space exploration method to optimize feature detector parameters on sonar imagery;
3. A comparison of the relative performances of feature detectors on a simulated synthetic aperture sonar dataset.

We explore the performance of our algorithm via a designated simulator for sonar images, and over a set of sonar images we obtained using our own AUV platform. This SAS image simulator [AD18] has been used previously to train methods that have been experimentally verified on real SAS images, and thus we can expect similar reliability. The results show that identifying the background and utilizing a feature detector with parameters optimized for that background yields the best result. Yet we also observe that in the general case where the background is not known or estimated, utilizing parameters trained for all backgrounds together produces the most reliable performance. Additionally, the design space exploration yields much higher performance than the default values for every feature detector. The optimized parameters for each background also yield different performances, dramatically so for some feature detectors.

This paper is organized as follows. In Section 3.2, we describe the state-of-the-art in target detection on sonar images. Section 3.3 includes the details of our target detection method. In Section 3.4 we show how the feature exploration's parameters are set. Section 3.5 includes results from synthetic images and from real sonar images. We conclude in Section 3.6.

3.2 Related Work

There has been a great deal of interest of non-model based object detection in sonar imagery for segmentation and mapping unknown environments. However, the literature is sparse when it comes to analyzing and optimizing feature detection, which is the very first step of the process.

Feature-based detectors for sonar images have been mostly used for applications related to subsea navigation. In particular, the main approaches use feature extraction to identify similar objects within a set of sonar images. Wililams *et al.* [WDDW01] perform two steps for such identification: first, they identify areas of constant depth as those may correspond to real objects. Then they classify pixels which are of consistent depth and not larger than a specified threshold as point features and feed them into subsequent feature matching for navigation. Similarly, the work by Johannsson *et al.* [JKE$^+$10] utilizes a simple computer vision pipeline of smoothing, gradient thresholding, and clustering, which is similar to simple feature detectors that the work by Harris *et al.* [HS88] evaluates. They use these features to perform registration on overlapping frames, which in turn generates motion estimates between frames for SLAM.

The use of feature extraction for SLAM applications is common. Ribas *et al.* [RRNT06], [RRTN08] use a mechanical forward looking scanning sonar (FLS) that produces consecutive sonar images while the underwater vehicle is moving. The movement creates a warped image since different parts of the image will be observed from different locations. To remove these distortions, they use a motion estimate to re-align each pixel to a globally consistent frame.

After this pre-processing step, they use a simple line extraction detector for target identification. Similarly, Fallon *et al.* [FFML13] propose a feature-based navigation system to incorporate features from high resolution maps of the seafloor and relocate objects using a lower resolution FLS. To detect features from the FLS, they first adaptively thresholded each image based on altitude to remove background noise. Then, they segment the image and use a gradient based feature detector with a threshold based on the average background image level to identify points of highlight and shadow. They then register these features back against the map and use them in a SLAM framework. Yet, the results show that performance are highly dependent on the seabed type.

While the above methods are tested for realistic sonar images, we identify some major gaps. In particular, the algorithms do not consider the spatial variation in sonar images. Even more importantly, the available methods optimize their parameters based on a small scale of cases, which may not fit the aim of robustness to various target types and different seabed structures. Specifically, these methods are missing an analysis of how to optimize the performance of these detectors in a broader set of environments. In this paper, we consider these two observations and offer a method for feature-based target detection that uses spatial information in target identification, as well as a rigorous methodology of how to set parameters to match a self-identified seabed type.

3.3 Feature Based Target Detection

3.3.1 System Model and Key Idea

Our system model includes a single sonar image of any type, e.g., SAS, multibeam, or FLS. The sonar image we consider is two-dimensional (2D), although 3D extension is straight forward. The sonar image may or may not include a target, but we assume that if a target is present, it is distinguishable from the seabed environment. That is, it is either isolated compared to similar objects (e.g., a cluster of rocks) or it has a unique pattern. We do not assume knowledge about the shape of the target. Our goal is to identify the existence and location of a target in the given sonar image. In our scheme, we use knowledge of the seabed environment type: specifically sand ripples, grass, and a mix of these backgrounds.

Figure 3.3 outlines our general approach. We start with an image (block 1) and setting the parameters (block 2) to a feature detection algorithm to detect an object in the presence of any typical seafloor pattern. These default parameters will be the ones that give the best results across a mix of all backgrounds. The detection algorithm then returns keypoints (block 3), which are the pixel coordinates and calculated intensities associated with a feature. We rank the intensity of the keypoints and apply a threshold (block 4)[1] to return the best keypoints and their coordinates. Once we identify the subset of features and locations (block 5), we extract the additional feature descriptors (blocks 6-7) and we make a decision about the seabed type between four different clusters (block 8): sand ripple; grass; sand; and mix. Then we execute the feature extraction algorithm again for the set of parameters that best fits the evaluated seabed environment (blocks 2-7). Finally, we perform target detection (block 9) based on the joint spatial spread of the detected features to identify a single or a set of targets that stand out from their environment either in terms of their pattern or by being spatially separate.

3.3.2 Feature Extraction

We base our detector on an underlying feature extraction algorithm. In this work, we do not develop our own feature extracting algorithm. Rather, we explore a variety of such filters, and provide the framework to include these algorithms for target detection in sonar images. Non-model based feature detectors are all fundamentally based on analyzing intensity gradients in an image, but differ in determining uniqueness based on patterns at a point (i.e. corner) or within a region surrounding a point (i.e. blob) or some combination of the two.

[1]Note the threshold is determined as a system parameter and is not user defined.

We compare seven feature extraction algorithms: Harris [HS88], Shi-Tomasi [S$^+$94], STAR [Ope18], FAST [RD06], SIFT [Low04], SURF [BTVG06], and ORB [RRKB11]. Each of these are widely-used, are considered general purpose, and are similar to detectors used by state-of-the-art feature based navigation methods [FFML13],[JKE$^+$10]. Further, these algorithms are all available in the open-source library OpenCV [Bra00]. These algorithms act as the Detection Filter block in Fig. 3.3. Given a set of parameters, the feature detector gives coordinates for a feature along with a single intensity value. Some detectors, namely STAR, SIFT, SURF, and ORB, return additional information beyond the single intensity value, such as scale and orientation of each feature.

The key idea, pros and cons, average running time per feature, and the number of parameters to tune for each of these detectors are summarized in Table 3.1. The average running time per feature is computed using each image in our dataset (explained in Section 5.1) and every possible parameter configuration on a 3.2GHz Intel Core i5 processor with 16-GB 2400-MHz

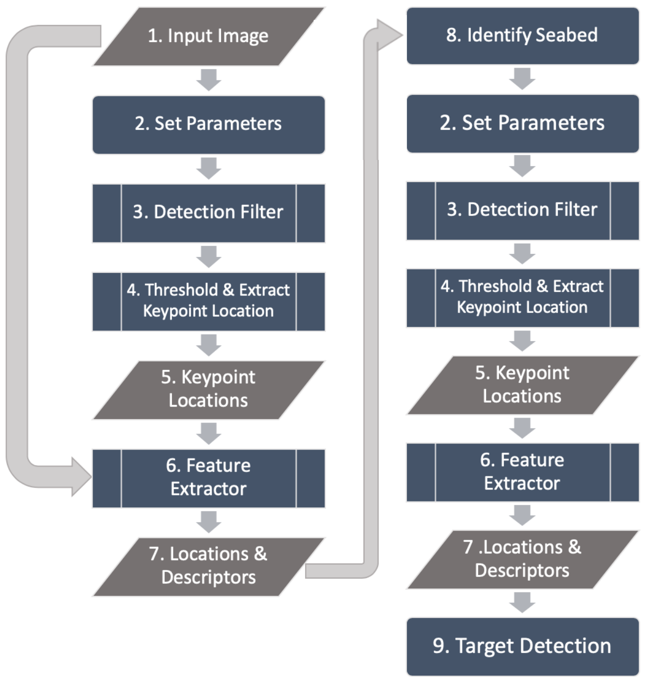


Figure 3.3: General framework for our feature-based detection algorithm.

Table 3.1: Comparison of feature detectors

Feature Detector	Description	Number of Parameters, Number of Configurations	Average Time Per Feature	Pros	Cons
Harris [HS88]	Computes gradient with respect to each direction.	5 parameters, 10000 configurations.	637.01 μs	Distinguishes corners and edges well	Susceptible to scale variance
Shi-Tomasi [S$^+$94]	Similar to Harris but uses a simpler thresholding method.	3 parameters, 10000 configurations.	682.77 μs	Distinguishes corners and edges well	Susceptible to scale variance
STAR [Ope18]	Multi-scale detector with no subsampling base on CenSure [AKB08].	5 parameters, 1000 configurations.	1257.68 μs	Robust to viewpoint changes	Susceptible to brightness changes
FAST [RD06]	Computes difference in brightness of neighbors in Bresenham circle.	3 parameters, 500 configurations.	16.42 μs	Efficient	Poor noise robustness, susceptible to scale and illumination variance
SIFT [Low04]	Computes oriented gradient histograms for patches around a point.	4 parameters, 5000 configurations.	793.94 μs	Rotation and scale invariant	Computationally expensive, susceptible to blur
SURF [BTVG06]	A more efficient approximation of SIFT.	3 parameters, 1000 configurations.	73.35 μs	Faster than SIFT	Susceptible to viewpoint and illumination change
ORB [RRKB11]	Efficient replacement for SIFT that builds off of the FAST detector.	6 parameters, 15000 configurations.	34.84 μs	Scale and rotation invariant, real-time, noise resilience	Generally fewer features

DDR4 RAM. The parameters used by each of these detectors affect the performance in terms of feature detection. For example, SURF's three parameters affect the threshold at which points are accepted or rejected and the size and number of octave layers, which are fundamental components to its operation. These, in turn, affect the size of features that SURF detects and returns.

3.3.3 Feature Classification

The output of the feature extraction process is a set of pixel coordinates with associated intensity values; i.e. a set of features. To decide on the important features out of those that passed the detection filter by means of classification, we label each feature with '1' if it is related to a target and '0' otherwise. As illustrated in Fig. 3.4, for classification we use this set of labeled features to train a support vector machine (SVM) model via a 4-fold validation process over a radial basis function (RBF) kernel. SVM was chosen due to its ability to perform well also for small sets of training data. The model takes the detectors' outputs (feature intensity, and, if relevant, scale and orientation), as well as the location of the identified features in the image.

Recall we assume that features corresponding to an object in a sonar image are isolated from other features. Thus, to classify an object, we use the relative position information to serve as an input to our SVM model. We avoid setting the absolute position information as

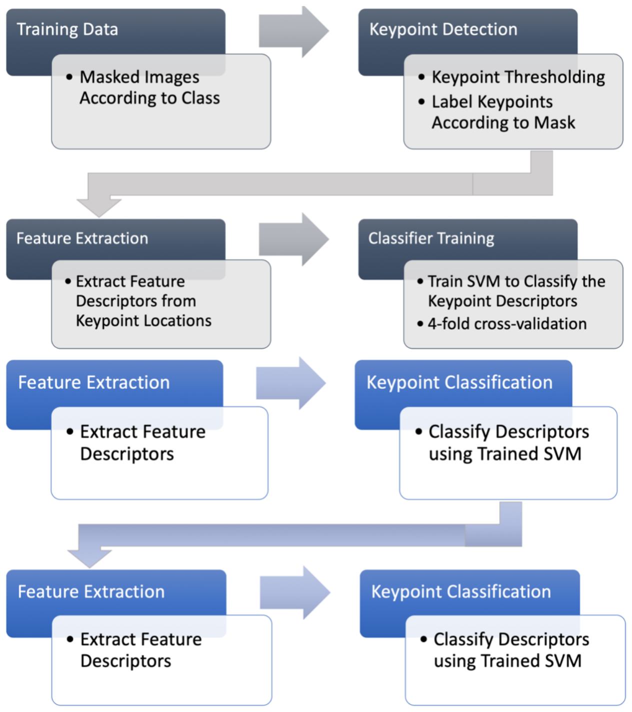

Training Data
• Masked Images According to Class
Keypoint Detection
• Keypoint Thresholding
• Label Keypoints According to Mask
Feature Extraction
• Extract Feature Descriptors from Keypoint Locations
Classifier Training
• Train SVM to Classify the Keypoint Descriptors
• 4-fold cross-validation
Feature Extraction
• Extract Feature Descriptors
Keypoint Classification
• Classify Descriptors using Trained SVM
Feature Extraction
• Extract Feature Descriptors
Keypoint Classification
• Classify Descriptors using Trained SVM

Figure 3.4: Feature classification. The first flow chart represents training an SVM model, and the second flow chart represents performing classification using that model.

an input, though, as an object may appear at any position in the image. Instead, we calculate two characteristics that relate to the feature's density. The first parameter describes any given feature's distance to the closest of two centroids, which we calculate given the total distribution of features in a single image. In images with a single object, numerous features around the object will create a centroid that is very close to the position of the object. Therefore, theoretically, features that have smaller distances to their nearest centroids would be more likely associated with an object. The pseudocode to setup the distance characteristic is outlined in Algorithm 1, and a visual representation is depicted in Fig. 3.5.

ALGORITHM 1
Distance to Nearest Centroid

```
$nClasses \leftarrow 2$
$kMeansIterations \leftarrow 15$
$c \leftarrow k\ Means\ Init\ Centroids(features, nClasses)$
**for** $i = 1 : kMeansIterations$ **do**
  $idx \leftarrow find\ Closest\ Centroids(features, c)$
  $c \leftarrow compute\ Centroids(features, idx, nClasses)$
**end for**
**for** $p \in features$ **do**
  $distCentroid1 \leftarrow norm(p - c(1))$
  $distCentroid2 \leftarrow norm(p - c(2))$
  **if** $distCentroid1 < distCentroid2$ **then**
    $distance(p) \leftarrow distCentroid1$
  **else**
    $distance(p) \leftarrow distCentroid2$
  **end if**
**end for**
```

The second parameter is a density index, as related by Equation (3.1), that quantifies the relative distance from a single point to all other points in the image. This way, features that are close to many other features will have a lower value than features that are isolated in the image.

These two feature density parameters and the parameters inherently calculated by the feature detector all make up the "keypoint descriptors" in Fig. 3.4 that are input to the SVM model. After training, the SVM, for each keypoint descriptor, will output either a '1' for 'object'

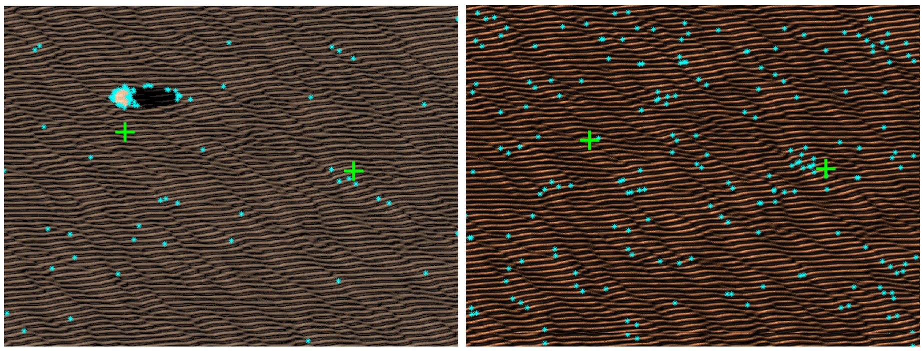


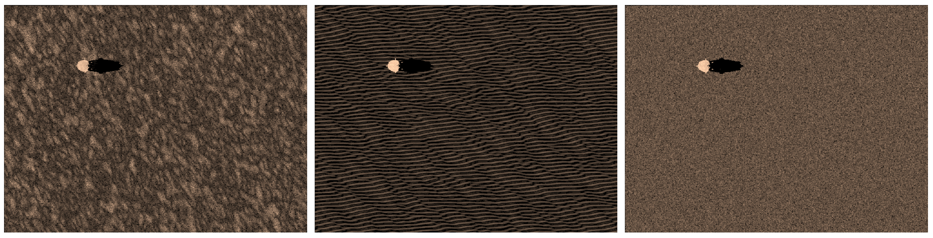
Figure 3.5: Placement of centroids on simulated sonar images. Features are displayed as cyan points, and the two centroids are the green plus symbols. Features associated with an object (left) will on average be closer to a centroid than features not associated with an object (right).

or '0' for 'non-object'. By this method we perform feature classification.

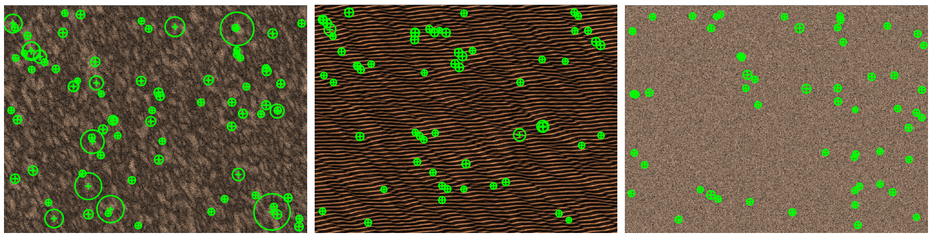
3.3.4 Identifying Seabed Structure

Once we extract the set of features, $\mathcal{F}$, we make a decision on the type of seabed. Information on the seabed type allows us to better determine the feature extraction algorithm's parameters. We take a set of images of an unknown seabed and compute the number of features, the distribution of features within images, and feature descriptors. These criteria are compared to averages of the same criteria computed from a representative dataset of three environments. If the criteria from $\mathcal{F}$ match one of these environments within a predetermined threshold, we make a decision on the type of seabed and apply the optimal parameters for that seabed. Referring to Fig. 3.6, we distinguish between *sea grass*, *sand*, and *sand ripples*. We use *mix* and its corresponding optimal parameters for a seabed environment that is not well distinguished by the above criteria. Then, once we properly set these parameters, we execute the feature extraction algorithm again so that we can glean the most accurate set of features possible.

We base our decision regarding the type of seabed on fixed thresholds on the distribution, number of the features extracted, and descriptors from the feature detector. We learn these thresholds from a simulator built by the definitions from Abu *et al.* [AD18] for the different three



(a) Example Simulated sonar images.



(b) Distribution of SURF features on the simulated seabeds. The location of a feature is represented by a green plus symbol, and the scale associated with that feature is represented by a green circle around it.

Figure 3.6: Simulated sonar images with grass (left), ripple (middle), and sand (right) seabeds.

seabeds in Fig. 3.6, while normalizing by the number of pixels in the sonar image.

To quantify the spatial spread of the features, we project the 2D location space of each feature f_i into a *closeness* index. Let $\mathbf{r}_i$ be the 2D location of the ith feature, and $d_{i,j}$ be the Euclidean distance between features i and j. We choose the closeness index to be

$$h_i = \sum_{k,k\neq i} \sum_{j,j\neq i,k} \left(1 - \frac{(\mathbf{r}_i - \mathbf{r}_k)\,(\mathbf{r}_j - \mathbf{r}_k)}{\|d_{i,k}\|\|d_{i,j}\|} \right) , \tag{3.1}$$

which, for a point i iterates through each feature k and compares it to every pair of feature i, j. If i is far from most other features, and those features are relatively close to each other compared to i, then the equation will evaluate to a relatively large value. Thusly, h_i measures how isolated feature i is relative to the other features, such that the higher h_i is, the more isolated feature i is. Then, we compare the mean of $h_i\ \forall i$ together with the number of extracted features by the above logic to determine if the current seabed is composed of a few/large number of isolated/dense features, and then identify the seabed environment.

3.3.5 Target Detection

Once we identify the seabed, the feature classification step is run with the detector parameters tuned for that specific seabed type. This yields a set of features that we can now analyze to identify the coordinates of a single or a set number of targets within an image. We use three methods for identifying a target. The first method sorts the features by intensity and applies a simple threshold to label the n features with the highest intensities as n targets in the image. This is the method that is typically used, as by design a higher intensity indicates a 'stronger' feature.

The second method calculates weights for each feature using (3.1). The weights are then normalized by calculating

$$\hat{w}_i = \frac{h_i}{\sum\limits_j h_j}, \tag{3.2}$$

and then the weight $\hat{w}_i$ of each feature is multiplied by its intensity v_i:

$$\bar{v}_i = \hat{w}_i v_i. \tag{3.3}$$

The intuition here is that intensity is not the only quality of a feature indicating it can correspond to an object. The spatial relationship between one feature and all other features can also hint at the existence of an object. Combining this information will theoretically lead to a more robust target detection.

The third method utilizes the same weights $\hat{w}$ and multiplies by the intensity of a feature normalized with a fixed window:

$$\tilde{v}_i = \hat{w}_i \frac{v_i}{\frac{1}{\|S_j\|} \sum\limits_{j \in S_j} v_j}, \tag{3.4}$$

where S_j is the set of features within a 5 by 5 pixel window surrounding feature j. Features corresponding to objects will theoretically be more different than their immediately surrounding features in comparison to the difference between features corresponding to noise or backgrounds. However, this method relies on the window size being larger than the objects in the image so that the intensity differences are more significant.

Finally, similar to the first method, a threshold is set for the sorted values of $\bar{v}$ and $\tilde{v}$ to identify the n targets within the image.

3.4 Design Space Exploration

One of the main challenges in our feature-based target detection is how to determine the optimal set of parameters for each feature detection algorithm to obtain a favorable trade off between the true positive rate and the false positive rate. While each of the algorithms we consider may only have 4 or 5 parameters, the number of possible combinations quickly builds up as the granularity of the sampling for each parameter increases.

To illustrate this, consider an algorithm with only 5 parameters each having only 4 possible values. Then, the total number of designs is 1024. Now consider an evaluation process for a single design that lasts for 10 minutes. The result would be approximately 1 week of continuous computing to fully explore the design space. Naturally, this duration compounds when several seabed environments and target types are explored. Clearly a method to reduce the computing time while still achieving a good approximation of the set of optimal designs is required. While for optic images the default parameters supplied with the feature extractors are designed to be suitable, to the best of our knowledge, how to set the parameters for sonar images has not been explored.

3.4.1 Exploring the Space of Parameters

We propose a parameter-tuning methodology based on active learning. While there exists several options for parameter exploration via active learning, e.g., [LC13],[IMS+08],[BNZ+16], we adopt previous design space exploration work known as ATNE [MAGK16] that we have modified for our application, which works by sampling a design based on whether it guesses that the results will either further explore the design space or get closer to the optimal edge. Instead of brute-force exploring the whole sample domain, ATNE covers a subset sufficient to estimate the Pareto-optimal front for even very complicated non-linear design spaces. It marks an improvement on previous active learning papers by requiring fewer initial random samples: 6 compared to 3000 [BNZ$^+$16]. Additionally, it can handle complex design spaces of a variety of sizes and retain similar prediction quality. By running ATNE on each detector for each dataset of seabed environments (200 images for grass, ripple, and sand, and all 600 images for the mix dataset), we can generate a set of feature detector parameters. The result is a decrease by 70% in the exploration time of the best set of parameters. Each design can take up to 8 minutes to evaluate on an Intel Core i5-6500 Skylake Quad 3.2GHz with 16GB of memory and a Sapphire Radeon Nitro R7 370 4GB GPU. Thus, with a feature detector with 500 possible designs, it saves about 47 hours of processing time.

To demonstrate the operation of ATNE, Figure 3.7 shows that a 30% sampling of the design space with the predicted best configurations in black, compared to the true optimal (brute forced) Pareto front in red. Additionally, the sampled designs are shown with a magenta dot, and the full space of designs are shown simply as blue circles. It is clear that ATNE, though sampling only a portion of the designs, essentially reconstructs the space and estimates a Pareto-optimal set of designs that are close to the ground truth optimal set. We measure the closeness to ground truth using average minimum distance from reference set (ADRS) [MAGK16],[PSZ09] and ATNE. ADRS is defined as

$$\mathrm{ADRS}(P_{\mathrm{gt}}, P_{\mathrm{pred}}) = \frac{1}{|P_{\mathrm{gt}}|} \sum_{\mathbf{p}_{\mathrm{gt}} \in P_{\mathrm{gt}}} \min_{\mathbf{p}_{\mathrm{pred}} \in P_{\mathrm{pred}}} d(\mathbf{p}_{\mathrm{gt}}, \mathbf{p}_{\mathrm{pred}}), \tag{3.5}$$

where P_{pred} and P_{gt} are the sets of points in the predicted Pareto front and the ground truth Pareto

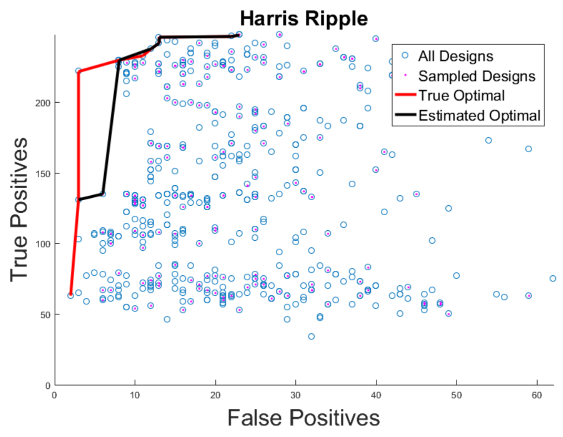


Figure 3.7: Example for a Design Space Exploration. The red line shows the true set of optimal designs, and the black line shows what was estimated utilizing only the magenta points.

front, and the function $d(\mathbf{p}_{\text{gt}}, \mathbf{p}_{\text{pred}})$ represents the distance between a point in the ground truth set and a point in the predicted set.

3.4.2 Utilization Function for Parameter Exploration

As any reinforcement learning technique, ATNE requires an objective function or space to converge to. Since our aim is target detection, we choose the receiver operator characteristic (ROC) which compares the number of correctly classified object features against the number of incorrectly classified non-object features. One alternative is the precision-recall curve, which is used in cases where there is an abundance of one class over another. Though there is an imbalance in our case (there are far fewer object pixels than background pixels), the ROC is a better choice because it is more important to us that the object as a whole is identified robustly in comparison with the background, rather than every single object pixel be identified. The ROC is a common way of choosing parameters for detections; yet, ROCs can also be used to compare detectors. In that context, we recall that a desired detector would yield a better false positive-true positive trade off in most of the cases compared to other options. If we use a graphical explanation (see for example Fig. 3.8), the best detector would be the one obtaining the far left curve of the ROC. Utilizing this observation, as a utility function, we feed ATNE with approximations of the ROC by sampling three true positive-false positive pairs on the curve.

Our procedure works as follows. For each feature detector design and parameter configuration, we execute our detection and classification workflow (see Section III) to obtain a list of features labeled as object. This list is then compared with ground truth information to yield a precision-recall pair for each parameter configuration. Once ATNE tests one design and configuration, it decides which design to sample next, based on its updated estimate of the design space. After sampling 30% of the space, we can then reasonably say that ATNE has recreated the Pareto-optimal front of the design space. The Pareto-optimal front will consist of the points that are on the edge of the scatter plot with the fewest false positives and most true positives.

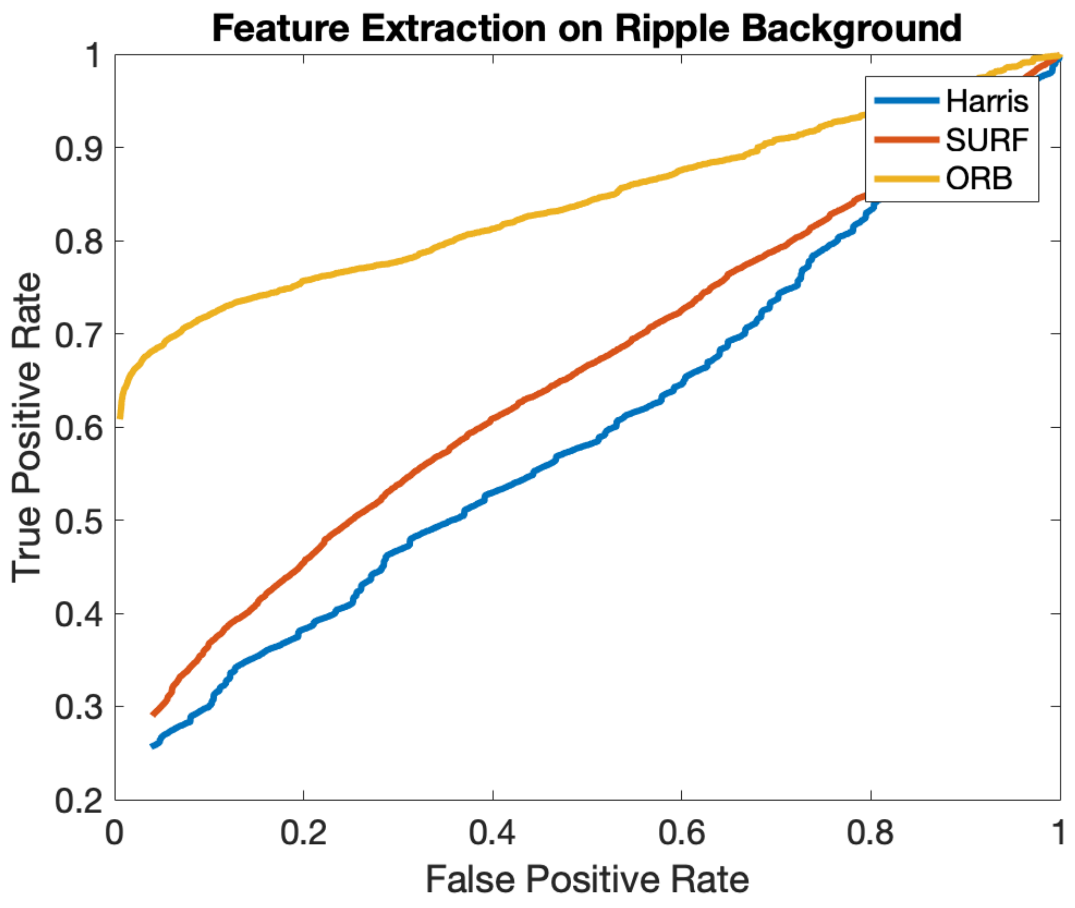


Figure 3.8: Three detectors' ROCs for a single design

3.5 Results

In this section, we explore the performance of our feature-based target detection scheme. Our dataset comprises 600 simulated sonar images to determine the parameter space of each feature extraction algorithm. In addition, we explore performance for some genuine sonar images that we gathered in multiple sea experiments. This dataset includes sonar images from a synthetic aperture sonar (SAS) mounted on our Eca-robotics Alister A18 AUV and we gather multibeam images using a surface ship-mounted multibeam. We start by exploring the set of parameters based on our simulator, and then provide sea trial results for target detection.

3.5.1 Simulations

The simulated SAS images depict a single target on one of three backgrounds: grass, sand ripple, or sand. An example of all three environments is shown in Fig. 3.6. The backgrounds have equal representation in the dataset, such that there are 200 grass-, 200 ripple-, and 200 sand-based sonar images. We obtain this dataset from a distribution function tailored to each of these seabed environments – see [AD18] for further details.

Within each background, there are many different target, shadow, and background intensities, so that there is sufficient variety represented in the dataset. Because the dataset is split into three backgrounds, there are four experiments that we can run: one for each background and one for a mix of all backgrounds. This allows us to test the suitability of various feature detectors for particular backgrounds in comparison to each other. This is done by measuring the number of true positives and false positives that each detector generates with a specific design, where the number of true positives corresponds to all the features that are within the bounding box of the object, and the number of false positives corresponds to all other features which are outside that bounding box. We relate these with Equations (3.6) and (3.7) for true positives and false positives, respectively. We measure the true positive performance by dividing the number of true positives,

N_{tp}, by the number of pixels that should have been detected but were not (i.e. false negatives), N_{fn}, plus the number of true positives. This denominator corresponds to the number of pixels that correspond to the object in the image. We measure the false positive performance by dividing the number of false positives, N_{fp}, by the number of pixels that were correctly not identified (i.e true negatives), N_{tn}, plus the number of false positives. This denominator corresponds to the number of pixels that correspond to the background of the image.

$$P_{\mathrm{tp}} = \frac{N_{\mathrm{tp}}}{N_{\mathrm{tp}} + N_{\mathrm{fn}}}, \tag{3.6}$$

$$P_{\mathrm{fp}} = \frac{N_{\neg\mathrm{fp}}}{N_{\mathrm{fp}} + N_{\mathrm{tn}}}, \tag{3.7}$$

3.5.2 Classification Results

Table 3.2 illustrates the performance of each of the seven feature detectors on each of the four datasets. We plot the optimal set of designs for each dataset for a given feature detector for comparison, with the true positive rate on the y-axis calculated by Equation (3.6) and the false positive rate on the x-axis calculated by Equation (3.7). With Table 3.2 we show the performance of the *grass*, *ripple*, *sand*, and *mix* optimal designs for each of the four datasets. In each plot, there is a line of best fit for each of the optimal design sets to obtain an estimate of the Pareto-optimal curve. For example, in the Harris Grass plot, there are five sets of designs that we test on the *grass* dataset. As expected, the designs that are optimized for the *grass* dataset perform the best, as the red dotted line is the closest to the top left corner. The designs optimized for the other datasets are shown as well: *ripple* in magenta, *sand* in green, *mix* in blue, and the default designs with no optimizations in black. In this particular instance, the *sand* designs appear closest to the most optimal set of designs.

We calculate the distance between each curve on a single plot using average distance to reference set (ADRS) as defined in Equation (3.5). Table 3.4 shows the ADRS values for each plot. ADRS gives a measure of how far the results are from the optimal results, so lower is better. An ADRS of 0 indicates that the sets exactly match. We calculate ADRS based off of the number of true positives and false positives for each point, rather than the performance metrics in Equations (3.6) and (3.7). Since all the images in the dataset contain the same number of pixels, this will represent the same relative performance as the plots in Table 3.2. For example, the Harris Grass plot in Table 3.2 corresponds to the Optimal Grass row in the Harris subtable in Table 3.4. In this particular case, as we already supposed from the plot, designs trained on the Sand dataset and tested on the Grass dataset are the closest to the grass optimal front, with an ADRS value of 1.47. This means that if we substitute the parameters for Harris which have been optimized for the *sand* dataset, we will get a similar performance as if we used *grass* parameters, which are optimized for the actual dataset that is being tested.

In all cases, as the plots illustrate, all optimized designs (even the ones trained and tested on mismatched datasets) perform better than the designs with default parameters. Additionally, in most of the combinations, using the designs optimized for the particular background that is being viewed will dramatically improve performance compared to using a mismatched training dataset.

We would expect that there be a consensus of which set of optimal designs would be best to use without any knowledge of the background that is being tested. In other words, that a single column in each of the subtables in Table 3.4 will contain a majority of bold values, such as is the case for the Shi-Tomasi table, where the designs optimized for the *mix* dataset are always second best in each of the individual datasets.

3.5.3 Comparing Feature Extraction Algorithms

While is not our main aim, our method provides a efficient way to compare the performance of the seven feature extraction algorithms used. We base our comparison on the ROC performance of the seven methods in Tables 3.2 and 3.4.

Table 3.2: In the majority of cases, all optimized curves perform better than the default curves (in black). The curves optimized for a particular background perform at least as well, but typically significantly better than curves optimized for other backgrounds.

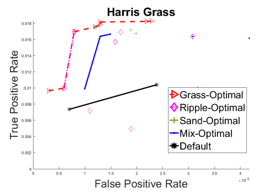
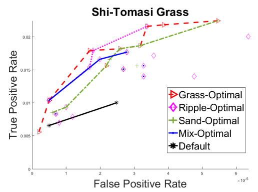
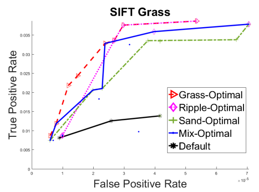
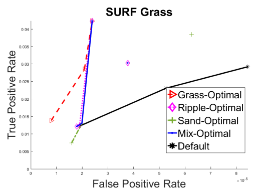
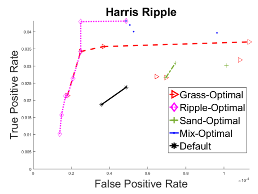
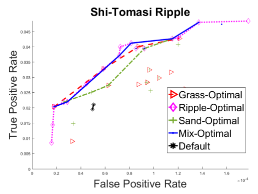
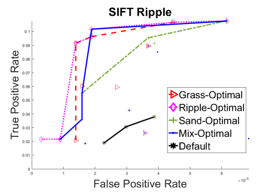
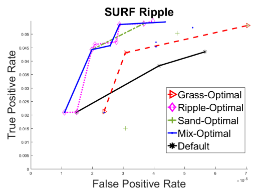
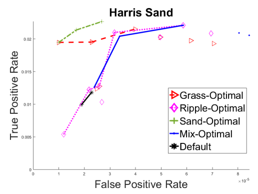
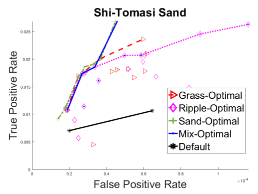
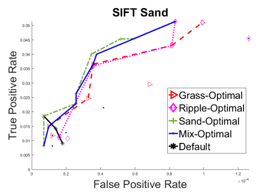
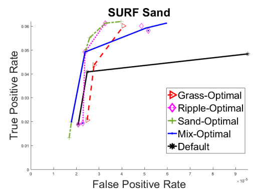
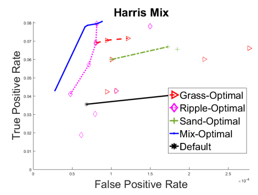
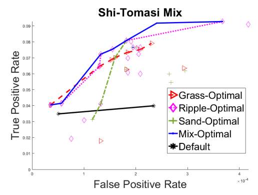
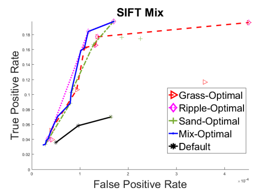
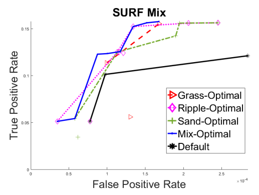
Harris	**Shi-Tomasi**	**SIFT**	**SURF**
			
			
			
			

Table 3.3: ROC Curves Continued

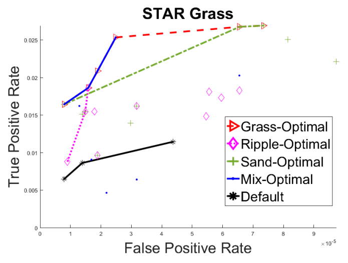
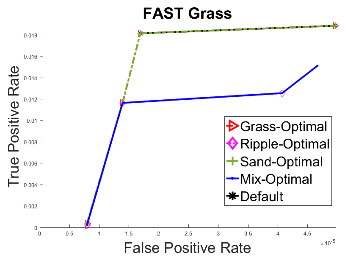
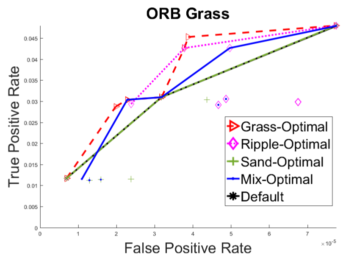
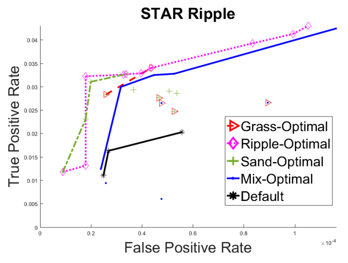
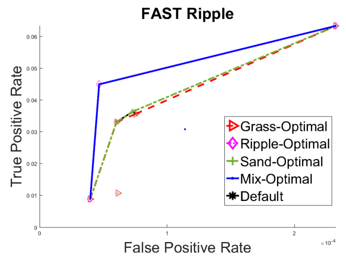
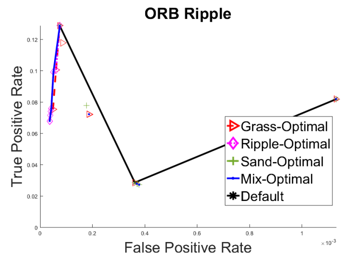
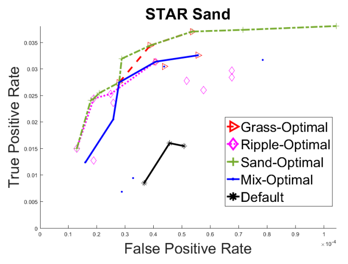
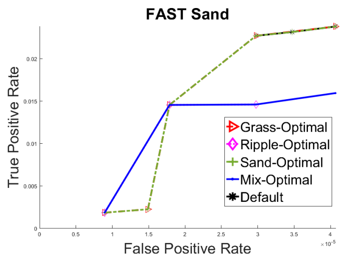
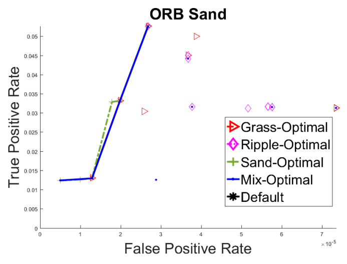
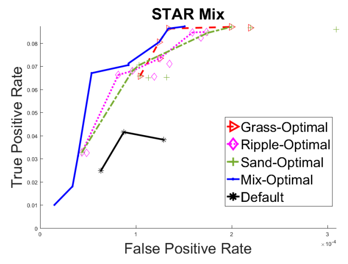
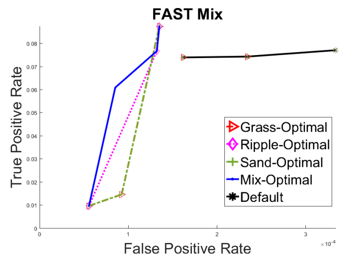
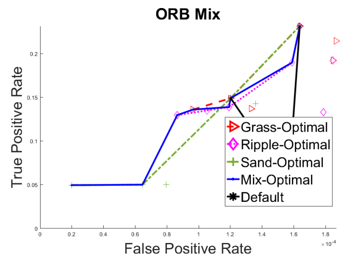
STAR	**FAST**	**ORB**
		
		
		
		

Table 3.4: ADRS values for each feature detector. The dataset with the designs closest to each optimal front is in bold. This is a numerical representation of the plots in Table 3.2, and support the conclusions drawn from visual analysis.

Harris	**Grass**	**Ripple**	**Sand**	**Mix**	**Shi-Tomasi**	**Grass**	**Ripple**	**Sand**	**Mix**
Optimal Grass	0	9.14	**1.47**	5.12	**Optimal Grass**	0	9.01	**5.12**	7.78
Optimal Ripple	**9.24**	0	24.15	16.19	**Optimal Ripple**	5.02	0	7.44	**0.50**
Optimal Sand	13.35	34.23	0	**12.41**	**Optimal Sand**	7.37	6.24	0	**2.36**
Optimal Mix	86.16	**56.76**	142.80	0	**Optimal Mix**	28.17	**19.31**	28.59	0
SIFT	**Grass**	**Ripple**	**Sand**	**Mix**	**SURF**	**Grass**	**Ripple**	**Sand**	**Mix**
Optimal Grass	0	**2.30**	6.94	6.62	**Optimal Grass**	0	**10.06**	20.50	10.17
Optimal Ripple	79.96	0	93.25	**50.41**	**Optimal Ripple**	5.23	0	17.32	**3.91**
Optimal Sand	28.65	19.50	0	**18.04**	**Optimal Sand**	18.22	9.45	0	**7.79**
Optimal Mix	22.32	**11.90**	25.16	0	**Optimal Mix**	32.59	**1.82**	39.08	0
STAR	**Grass**	**Ripple**	**Sand**	**Mix**	**FAST**	**Grass**	**Ripple**	**Sand**	**Mix**
Optimal Grass	0	20.00	**13.36**	33.35	**Optimal Grass**	0	2.95	**0**	9.77
Optimal Ripple	45.27	0	35.73	**25.60**	**Optimal Ripple**	46.33	0	52.87	**35.41**
Optimal Sand	**4.56**	7.60	0	25.35	**Optimal Sand**	**0**	0.24	0	3.68
Optimal Mix	**7.11**	10.77	9.46	0	**Optimal Mix**	20.22	**0**	17.49	0
ORB	**Grass**	**Ripple**	**Sand**	**Mix**					
Optimal Grass	0	3.65	2.13	**1.71**					
Optimal Ripple	**60.22**	0	289.55	140.56					
Optimal Sand	19.24	26.60	0	**12.20**					
Optimal Mix	25.52	9.58	**5.70**	0					

From the plots, and verifying with the ADRS values, the Shi-Tomasi, SURF, and ORB designs optimized for the dataset containing a mix of all backgrounds performs consistently well in comparison to the other feature detectors in all categories. These are the most robust of the 7 feature detectors that were tested, and would function better than the others in environments with a variety of seabeds if there was no ability to detect the type of seabed.

Harris is interesting in that each set of designs trained on the individual datasets perform very close to optimally when tested on the datasets all together, but the designs trained on the datasets all together only perform averagely when tested on individual datasets. FAST is also unique in that it has a smaller design space and a number of the designs have identical performance. This yields a situation where the designs trained on *grass* and *sand* are identically optimal when testing on *grass* and *sand* datasets, but have different ADRS values when testing on the *ripple* or *mix* datasets.

3.5.4 Real Sonar Images

Table 3.5 shows the performance of the target detection step on real sonar images. The first two images are from a surface ship mounted multibeam sonar. The first image contains multiple prominent rocks on a nonhomogenous seafloor. The second image contains three targets deliberately placed on the seafloor, but are not as prominent against the background. The third and fourth images are from our AUV mounted SAS, and each contain a single target: a simple pipe and a complex shipwreck, respectively. For the first two images, which contain multiple targets, we expect that Method 1, a simple threshold filter, will identify the single most prominent target. We also expect that Methods 2 and 3, which take the distribution of features into account, will identify other targets in the image as well. Additionally, for all images we expect that features with high intensities that correspond to the seafloor will be eliminated by Methods 2 and 3.

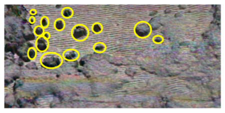
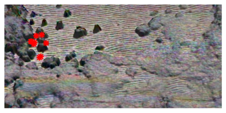
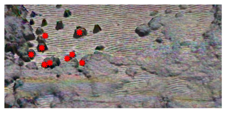
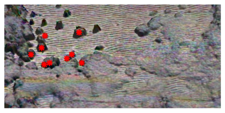
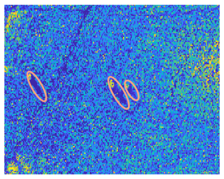
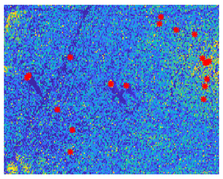
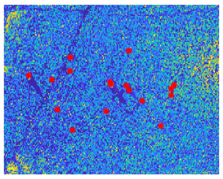
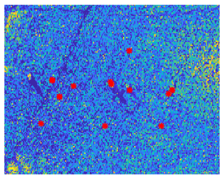
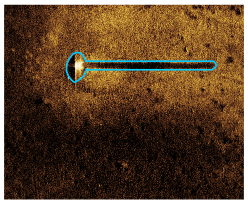
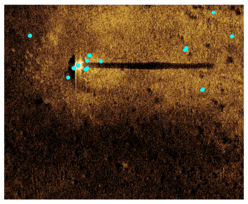
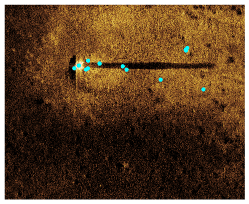
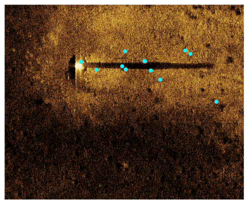
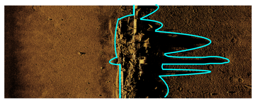
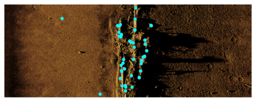
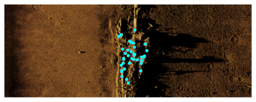
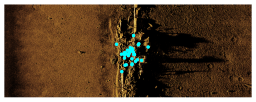
We show results for the SURF feature extraction algorithm, and note that the results are similar in trends for the other feature extraction algorithms. The images each have at least one target, outlined in the first column. The methods we test are outlined in Section III.E: Method 1 corresponds to the simple feature intensity thresholding, Method 2 corresponds to (3.3), and Method 3 corresponds to (3.4).

We observe that Methods 2 and 3 present a clear improvement over Method 1. In particular, separate targets are identified more strongly in the first and second images, and in the first and third images the vast majority of false detections on the background are eliminated. The fourth image depicts a single, large, complex feature, and as such it is easily identifiable by all methods.

3.6 Conclusion

In this paper, we presented a feature extraction-based method to detect targets in sonar imagery. Our method does not require prior knowledge of the shape or distribution of the target, and is capable of target detection for multiple seabed environments and sonar types. We described the framework of our detector, and showed how it exploits the spatial diversity of the extracted features. Further, we described our methodology for exploring the parameter space across the various environment backgrounds to determine the parameters of the feature extraction algorithms. We demonstrated the applicability of our approach over a simulated dataset and examined it on genuine sonar imagery of different sonar types. The results showed that optimized parameters yields much higher performances than default parameters. Further work will combine the positives of all feature extraction algorithms to yield a better and more robust detection rate. Additionally, the use of neural networks may illuminate additional optimizations that the support vector machines we used could not, and presents an interesting opportunity for future research. This could be paired with additional research into a more robust sonar image simulator. The one used in this paper has limited dimensionality, and as such generating additional images to use a neural network runs the risk of overfitting.

Table 3.5: Target Detection Methods on Real Sonar Images with the SURF detector. Methods 2 and 3 reduce the number of false positives.

Original	**Method 1**	**Method 2**	**Method 3**
	 Object: 5 — Non-Object: 4	 Object: 8 — Non-Object: 1	 Object: 8 — Non-Object: 1
	 Object: 4 — Non-Object: 16	 Object: 8 — Non-Object: 12	 Object: 9 — Non-Object: 11
	 Object: 6 — Non-Object: 14	 Object: 15 — Non-Object: 5	 Object: 16 — Non-Object: 4
	 Object: 48 — Non-Object: 2	 Object: 50 — Non-Object: 0	 Object: 50 — Non-Object: 0

3.7 Acknowledgments

Chapter 3, in full, is a reprint of the material as it appears in IET Radar, Sonar, & Navigation 2020. Tueller, Peter; Kastner, Ryan; Diamant, Roee. The dissertation author was the primary investigator and author of this paper.

This work was sponsored in part by the NATO Science for Peace and Security Programme under grant G5293. A portion of this work with preliminary results was presented at OCEANS ’18 in Charleston, SC, USA on October 24, 2018 [TKD18].

Chapter 4

3D Reconstructions of Ancient Mayan Archaeological Excavations

In this chapter we present a workflow for performing 3D data capture for archaeologists, specifically with regards to the Mayan civilization at El Zotz in the Petén region of modern Guatemala. This workflow follows along the generic workflow presented in Figure 2.1 and each step is explored. For step 1, we use multiple different sensors: LiDAR, photogrammetry, RGBD cameras, and satellite data. The excavation sites are so remote and fragile that each sensor is extremely limited in what it can observe safely, and so we aim to combine the best aspects of each sensor to achieve a full reconstruction of the environment. Steps 2 and 3 focus on cleaning and combining the scans for each sensor individually and all together, respectively. Finally, step 4 encapsulates the entire reconstruction and how we communicate measurements that are important to archaeologists and contextualize the scale and geometry of the sites in ways that are intuitive and clear to people who have not physically visited the sites.

4.1 Introduction

Digital documentation techniques are becoming increasingly commonplace for visualization and measurement of archaeological excavations where high-resolution, remotely sensed data provide a more accurate record than traditional analog recording of excavation contexts, architecture and ancient monuments [GRN$^+$16]. However, each technique may only be viable for documenting certain aspects of the excavation, such as overall geometric structure or detailed interior color and texture.

Digital documentation of excavation M7-1 at El Zotz, Guatemala has been done over several field seasons using Terrestrial LiDAR. In 2014 the Faro Focus 3D 120 was used to generate a precise 3D model to aid Archaeologist in tunnel mapping. In 2016, structure M7-1 was excavated even further, and a new Point Cloud was acquired to merge with previous data. In 2019, the Leica BLK360 imaging laser scanner was used to append newly excavated tunnels to the Point Cloud previously acquired.

Starting in the 2017 expedition, we began to incorporate additional techniques for reconstruction called Simultaneous Localization and Mapping (SLAM). While the accuracy of the reconstructions created by SLAM is lower than the reconstructions from properly executed LiDAR scans, the usability is much higher. In other words, the ability to properly execute a scan in the confined quarters of the excavated tunnels is easier. Incorporating this approach has allowed us to abandon a ’one-size-fits-all’ approach, and instead utilize the best aspects of each sensor and combine their results in a way that is greater than the sum of their parts. We aim to find the combination in which usability is manageable and accuracy remains high.

Additionally, during the 2017 and 2021 field expeditions, SLAM based on RGBD scans from an Intel RealSense L515 camera were used to also gather Point Clouds in an effort to compare their accuracy to the LiDAR scans and to augment shortcomings of the LiDAR scans. Regardless of how the desired Point Clouds are obtained the steps that follow can be used to prepare the scan data for real-time applications using game engines like Unity or Unreal with a Desktop (e.g Non-Gaming, Gaming), Mobile-device (e.g Smartphone, Tablet), VR/AR HMD (e.g Oculus Quest, Microsoft Hololens), or Online Viewer (e.g Potree, SketchFab).

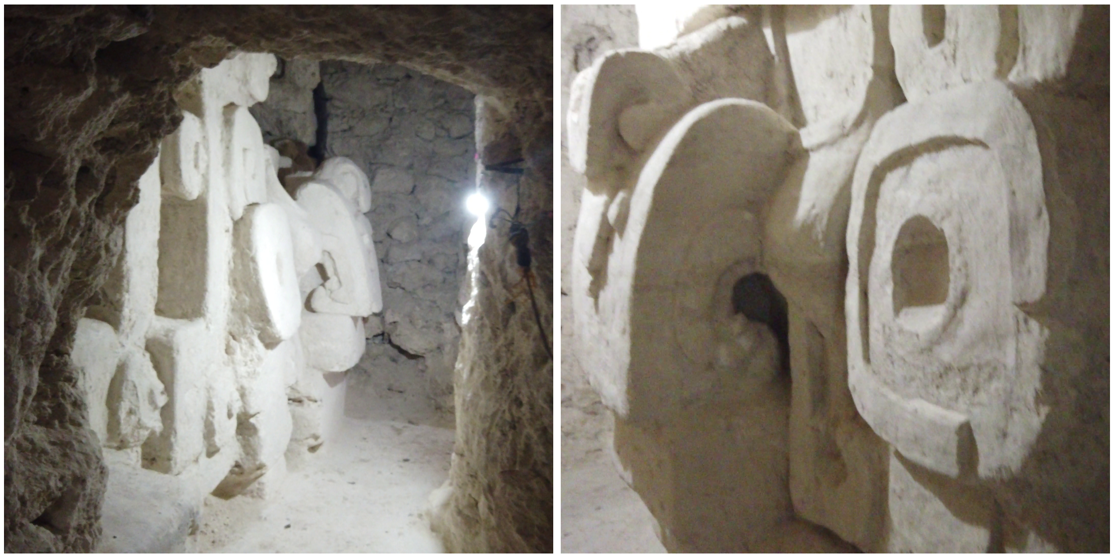


Figure 4.1: Interior of m7-1.

The contributions of this chapter are:

1. A comparison of 3D capture techniques for archaeological tunnel excavations to evaluate most accurate and most usable techniques for virtual representation;

2. A workflow for representing archaeological tunnel excavations using data from 3D sensors that is more complete and accurate than workflows using single sensors;

3. An exploration of virtual reality as a means for communicating and visualizing archaeological excavations;

4.2 Sensing for Archaeology

For Mesoamerican structures, tunneling is the primary method for archaeological excavation, as opposed to trenching. This is primarily because ancient Maya civilizations tended

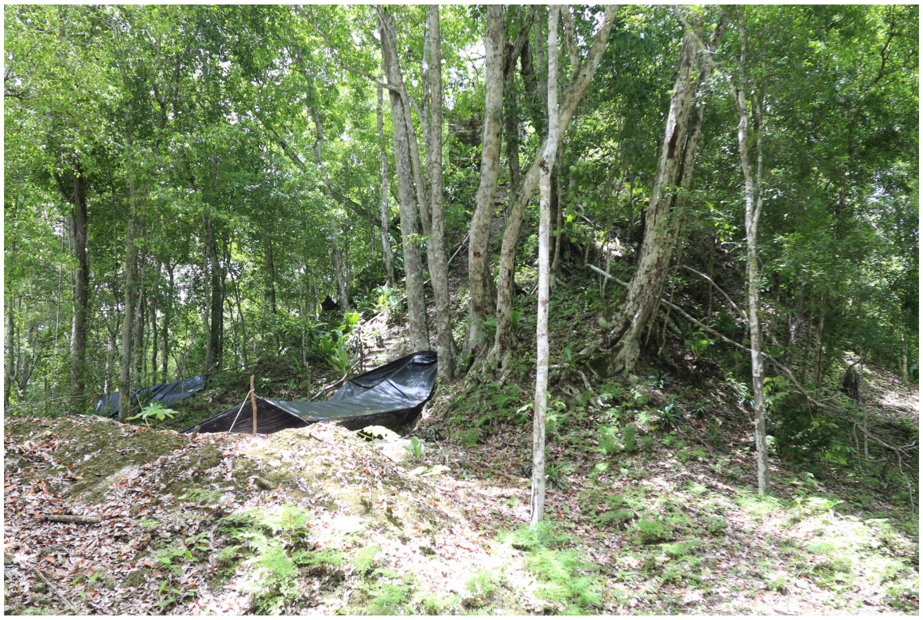


Figure 4.2: Exterior of m7-1

to build structures over and around other structures that were no longer usable. In most cases, these previous structures were deliberately preserved by the Maya people, and as such, as tunnels penetrate deeper into the site, archeologists travel back in time. Each layer must be kept intact, as it is all valuable from an archaeological perspective, and thus the tunnels that are created become labyrinthine and convoluted. Still, archaeologists must keep careful measurements of the dimensions of the tunnels so that geometric data of the structures and locations of artifacts can be accurately recorded, but due to the complicated nature of the tunnels, collecting measurements to create drawings is not easily usable. Additionally, because the excavations are not exposed, visually assessing the sites can be a difficult task. Being able to represent the sites with high resolution and accurate reconstructions is valuable not only for archaeologists to evaluate the sites from a geometric perspective, but also for a wider audience to evaluate and understand the sites from a historic and cultural perspective in a way that photographs cannot capture.

4.2.1 LiDAR

Terrestrial LiDAR scanners are high precision devices that can capture the geometry of their surrounding environment. LiDARs emit lasers in many directions, and calculate the distance to the device by measuring the time delay to the reflected signal. These devices are often used for cultural heritage documentation due to their high accuracy. This type of documentation is popular in Maya archaeology [GRN$^+$16, CLM14], which means that there is good, high-resolution baseline 3D data to compare against other sensors and methods. While individual 3D scans generally get millimeter accuracy, the process of collecting the data over extensive excavation sections is very long. Furthermore, the final result is obtained after a lengthy post-processing step that cannot be performed directly in the field, preventing a quick visualization of the current scan. Finally, the cost of high-quality laser scanners is often a barrier to large-scale deployment, or deployment in difficult environments (e.g., narrow, dusty tunnels) where the equipment is at risk of being damaged.

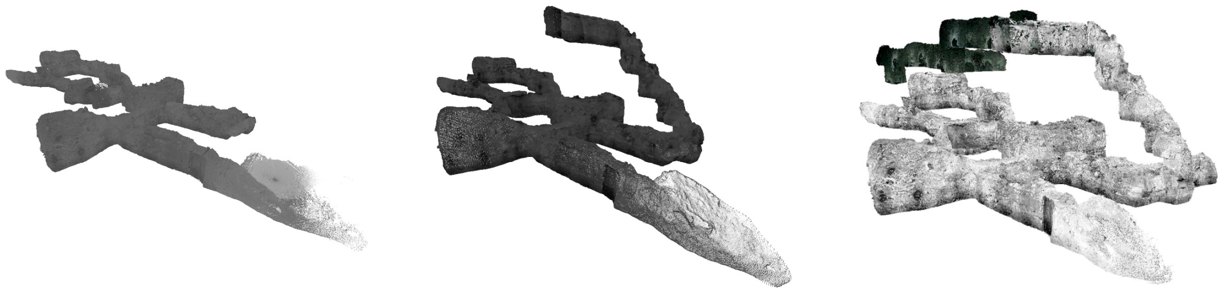


Figure 4.3: Using Fine Registration to extend the virtual excavation with point cloud data from 2015, 2017, and 2019 field expeditions.

4.2.2 Photogrammetry

"Structure-from-motion" photogrammetry is a general technique that uses computer vision algorithms to reconstruct a 3D shape from a set of 2D pictures. It is a very accessible

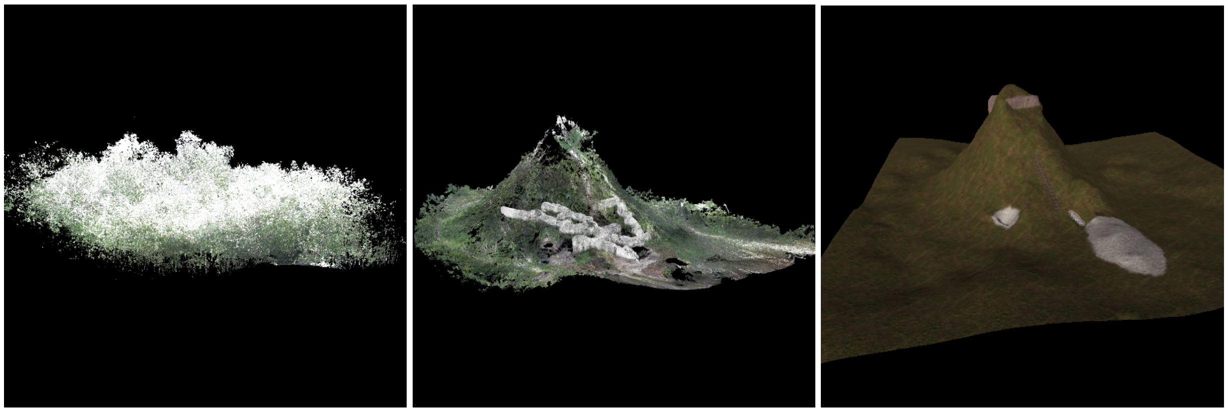


Figure 4.4: Left-Center: CSF Filter Point Cloud Results from m7-1 exterior perimeter. Right: Meshed Game Asset in Unity

solution to scan different types of environments; the user only needs a digital camera, and potentially a good light source. This technique also provides good results for cultural heritage documentation [Yas07, FFADT13]. The largest drawback is the computational time required to process the data. Due to this limitation, it can be difficult to evaluate the results in the field. Additionally, the computation time grows very rapidly with the number of photos, and that number grows quickly with the size of the area to scan, which renders the scan of an entire excavation very difficult. The sometimes-cramped conditions of archaeological contexts with fragile cultural heritage monuments can also be a deterrent to photogrammetric methods that require the photographer to be in close proximity to sensitive features for extended periods of time.

4.2.3 Simultaneous Localization And Mapping with RGBD Sensors

Simultaneous Localization And Mapping (SLAM) algorithms are a class of algorithms – first defined in 1986 [SC86] – focused on localizing an agent within an unknown environment while simultaneously building a map of this environment [DB06]. The goal of SLAM is to provide an update of the position and/or a map in real-time. The definition of real-time varies with the application, but in our case, we target an update around the same speed as camera sensors,

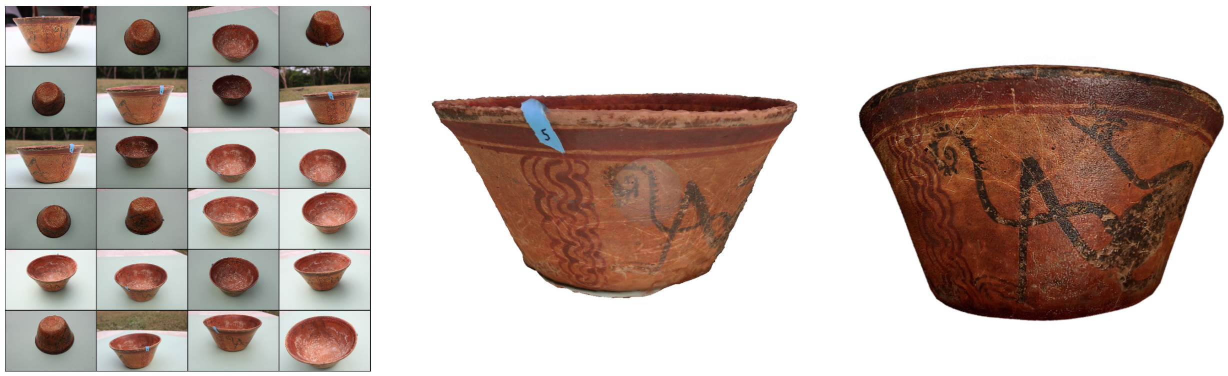


Figure 4.5: Spider Monkey Bowl found in m7-1. From left: photographs taken back at camp. Center: Agisoft Photoscan reconstruction output, Right: Final Game Asset inside Unreal Engine.

i.e., 30 Hz. We are particularly interested in the use of visual SLAM algorithms, utilizing visual sensors, but SLAM works with any number and combinations of sensors ([KVSMK11, MSK+11, LLZH14, FZG+17, IA17, MAT17]).

In parallel with the development of SLAM algorithms, many low-cost visual and depth sensors have been commercialized in the past few years. The Microsoft Kinect and the Intel Realsense are two examples of such sensors, which combine an RGB camera with a depth sensor that provides geometrical information about the scene.

The combination of SLAM algorithms and low-cost sensors yields a good solution to the problem of obtaining quick 3D scans of a scene. However, the quality of results is unknown and can be difficult to assess for multiple reasons. First, there is a wide variety of SLAM algorithms available, all using different methods to achieve the best quality of results. Second, each sensor has its own specificity, and specifically all sensors present a certain degree of noise that needs to be mitigated by the algorithms. Finally, many algorithms are developed in a similar, well-lit environment, but few are tested in the field where the condition are often less than ideal to maintain a good quality of tracking. We tested our prototype in Guatemala where archaeological excavations at a Classic Maya (250–1000 CE) site present a challenging environment due to the lack of good lighting, and narrow tunnels without connecting loops.

Several studies actually compare and evaluate SLAM algorithms. Zennaro and colleagues [ZMM+15] compared two low-cost depth cameras in terms of noise, and Kraft's team [KNS+17] evaluated these cameras on trajectory estimation. Huletski and colleagues [HKK15] compared multiple SLAM algorithms on a standard dataset. The common point of these studies is the use of an indoor, office-like environment. Though there has also been extensive evaluation of SLAM in outdoor environments for autonomous driving [BAYG17]. Most of the evaluations are driven by the availability of data, typically through standard benchmarks with provided ground truth (e.g., [SEE+12, GLU12]).

Other works study the usage of various handheld or robot-mounted sensors dedicated to the collection of 3D data in cultural heritage sites. Various studies present the evaluation of a handheld LiDAR scanning system in different field environments ([ZBG+14, Far16, DFSAB+18, NMR+17]). This system constitutes an improvement over larger terrestrial LiDARs in terms of usability, but remains a costly piece of equipment along with closed-source, commercial software. [NMR+17] and [MFG+17] both evaluated a commercial backpack mapping system, providing a good mobile sensing solution, but also containing costly sensors. We present a study of multiple SLAM algorithms with multiple consumer-grade depth cameras, (see [LMFS16] for a similar study), in a more constrained environment, and comparing more recent and complex SLAM algorithms.

4.3 Multi-Sensor Registration

Previous expeditions tried to utilize just one sensor to gather the most useful data for a single site. As the need for covering more detail of the site increased, though, each sensor began to give diminishing returns in terms of usability and accuracy. In order to capture every nook and cranny, the number of LiDAR scans that need to be set up and executed increase exponentially. Photogrammetry is good for small areas, but large geometric reconstructions are infeasible. Finally, reconstructions from SLAM techniques are easy to gather in a tunnel environment, but the sensors lose tracking over large areas and cannot capture low level detail. We aim to utilize the best of each of the sensors to create a full, accurate, immersive 3D model of the excavations, while greatly reducing the time and difficulty of data gathering.

While the benefits of multi-sensor registration may seem clear, it is not often done because the types of data each sensor provides can be drastically different not just in terms of format, but also in terms of differing types of error and scale. In this chapter we present a method for combining data from disparate sensors that are typically used in archaeological scanning that unlocks the ability to perform this multi-sensor registration. A full tutorial and descriptions of the usage of each software component to perform this multi-sensor registration can be found online at our wiki here: `https://gitlab.nrp-nautilus.io/maya-archeology/maya-archeology-unity/-/wikis/home`.

4.3.1 Acquisition

LiDAR

The process for Terrestrial LiDAR Scanning (TLS) generally involves setting the desired scan density, which in affect increases the number of points collected and time needed per scan. In the case of the Leica BLK360 TLS, medium quality was sufficient for our purposes and took 3 minutes per scan. Each scan occurred more than 1 meter apart depending on if all the environment's geometry could be reached from the previous scans location. The set of individual point clouds obtained was then registered using visual alignment in Leica's propriety software Cyclone Register 360. The output of all this is several standard file format .ptx files all in alignment with each other, which can then be brought to external software for further 3D point cloud and mesh processing.

Structure from Motion (SfM)

SfM point cloud derived models are able to store accurate RGB data at a higher resolution, but with lower geometric accuracy. Since RGB information collected during SfM have greater accuracy, they can be useful in projecting color information back onto the Mesh generated by LiDAR. We can do Fine registration with the Iterative Closet Point (ICP) algorithm on the two models to share the same coordinate space. Performing SfM on larger environments can be time consuming in collecting and processing data. In addition to this are the extra constraints of consistency in scene lighting, and obtaining enough coverage and overlap between photos of the entire scene. Thus we conceptually decompose an environment into it's most abundant components and perform SfM on these components under controlled lighting conditions for the purpose of high-detail surface scanning applicable toward environmental texturing. The m7-1 excavation was classified into two materials, it's general structure component of limestone stucco and the natural elements comprising the walls and ceiling of the excavation. This surface information can then be input into newer high-level development tools that utilize machine learning to upscale and synthesis material variations.

SLAM

SLAM point cloud models are similar in format to SfM point cloud models, but contain even less geometric accuracy. Generally, these models are used to fill out texturing and coloring in the portions of the scene that do not have associated SfM point clouds.

4.3.2 Software

CloudCompare

CloudCompare is an open source 3D point cloud and mesh processing software. Within the application are various tools to clean, project, align, subsample, measure, and perform statistical analysis. Once the software is ready you can drag and drop your set of standard 3d files (.obj, .fbx, .las, .xyz, .ply, etc) into the main window. This can take several minutes depending on your hardware specifications. After the loading process has been completed the sequence for preparing the point cloud for meshing can take on many variations. The Statistical Outlier Removal (SOR) filter is used to remove points that deviate from the average distances of its nearest neighbors. Overlapping points and areas of unnecessary high density can be addressed by performing a subsample operation on the data set. M7-1 had approximately 30 Meters of newly excavated features scanned using the BLK360 that needed to be appended onto the original model. To do so 'Fine Registration' using ICP (Iterative Closest Point) algorithm was done in CloudCompare over a subset of the two point clouds. This process also returns a transformation matrix which is then applied to the whole point cloud. Merging the exterior of M7-1 with the interior point cloud cluster requires overlap with the two point cloud clusters, which can be seen in Figure 4.6. Additional TLS work was done around the radius of the pyramid in an attempt to capture a digital elevation model (DEM) of the nearby terrain. A Cloth simulation filter (CSF) technique can be used to extract ground points from point clouds. Additionally, the size of the final cloud can be reduced so that it can fit on a VR platform through a process called decimation, which can be observed in Figure 4.7.

Meshroom

Meshroom is an open-source 3D Reconstruction software based on the AliceVision Framework that provides Natural Feature Extraction, Image Matching, Features Matching, Structure from Motion, Depth maps estimation, and Meshing. Additionally, the AliceVision Plugin for Houdini provides this functionality inside the 3D Procedural Software. Using Meshroom can be as simple as dragging your set of overlapping images into the application and pressing Play to reconstruct a 3D Mesh from photographs. We used Meshroom to produce our photogrammetry assets that are then sent to Houdini for optimization.

Houdini

Houdini is 3D procedural software that provides a node-based workflow to define a recipe of transformations. These workflows can then be easily shared across different projects. In our workflow are nodes that provide an iterative interface to tweak and send photogrammetry meshes through common modeling tasks such as Retopology, UV Mapping, Baking, and Texturing. The software is also used to generate a landscape based off LiDAR data and provide splat maps to detail the terrain with appropriate materials.

4.3.3 Game Engines

Unity

Unity is used to support El Zotz Maya Archeology Quest VR. The application was designed for mobile VR headsets accessed through a museum exhibition or digital download.

Unreal 4

Unreal 4 is used to support our Model Viewer publication tool that archaeologists can use to render out meshed reconstructions of field data collected and provide visuals for outreach.

Unreal 5

Unreal 5 coming soon provides a novel solution for handling seemingly infinite geometry through a ”virtualized micropolygon geometry system” called Nanite. It will also provide a new World Partitioning tool to handle automatic optimization of massive terrains. Future iterations of this project will attempt to push this technology to it’s limits by importing the raw scans from the archaeological site and procedural generating a huge surrounding forest environment from the aerial LiDAR survey.

4.3.4 VR Devices

The Oculus Quest was chosen as an all-in-one solution for providing access to these experiences anywhere. The console includes room-scale tracking and two wireless motion controllers with 6-DOF or Hand-tracking capability to interact with the virtual world.

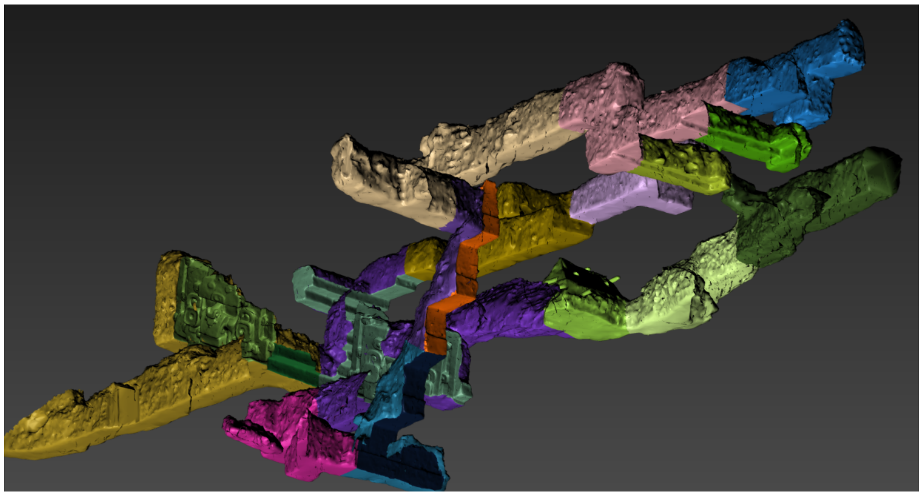


Figure 4.6: The 7 UDIM Sub-objects making up the model were further segmented for a total 21 parts represented by each color.

4.4 Results

This section presents the results collected to compare and contrast the various sensors and SLAM algorithms used in both a test enviornment (Arroyo Tapidao Mud Caves) and the deployment environment of M7-1. Additionally, the final VR models of the Maya excavations are presented.

4.4.1 Arroyo Tapiado Mud Caves

We performed a controlled set of experiments at the Arroyo Tapiado Mud Caves site near the University of California San Diego. This location was the best option for replicating the

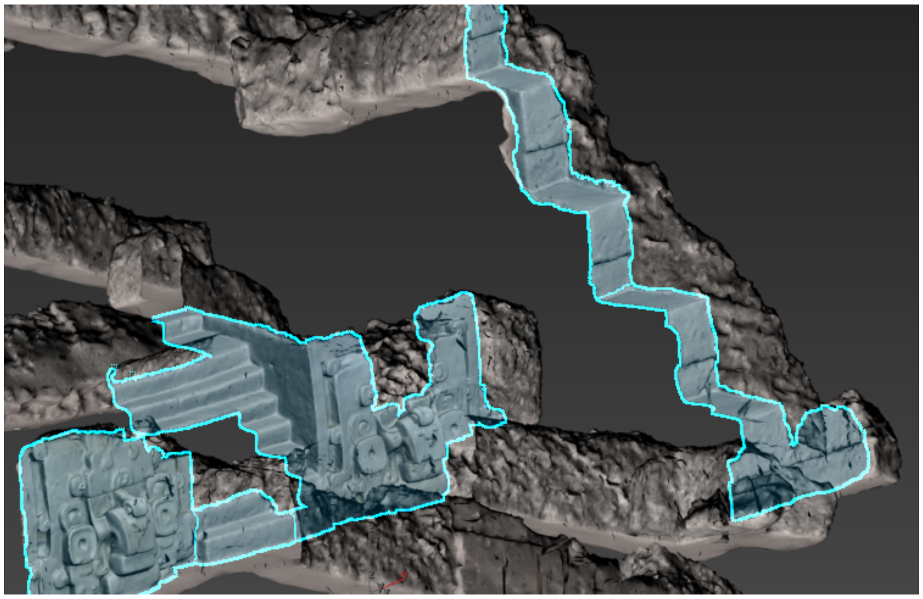


Figure 4.7: The highlighted areas require the most geometric detail and were separated from the overall mesh to preserve them during the decimation process.

archaeological conditions in Guatemala in a local setting where all of the laboratory resources would be available during prototype development. First, we defined an area of approximately 50 m in length and scanned this area with the same FARO LiDAR system that was used to scan the El Zotz excavations. Then, we ran a series of scans using our SLAM scanning solution with three different sensors: Kinect v1, Kinect v2, Realsense D435, and two different algorithms: ORB-SLAM, RTAB-MAP. All the scans were performed by the same operator, and followed a similar trajectory and scanning pattern. Additionally, we recorded all sensor data, and replayed them through the InfiniTAM v2 algorithm afterward.

Here we present the results of comparing the output data from the SLAM algorithms to the LiDAR 3D model.

Cloud-to-cloud difference

Figure 4.8 shows an example of the cloud-to-cloud difference between a SLAM scan and a LiDAR scan. On top is the LiDAR scan of the mud caves site where we have focused our

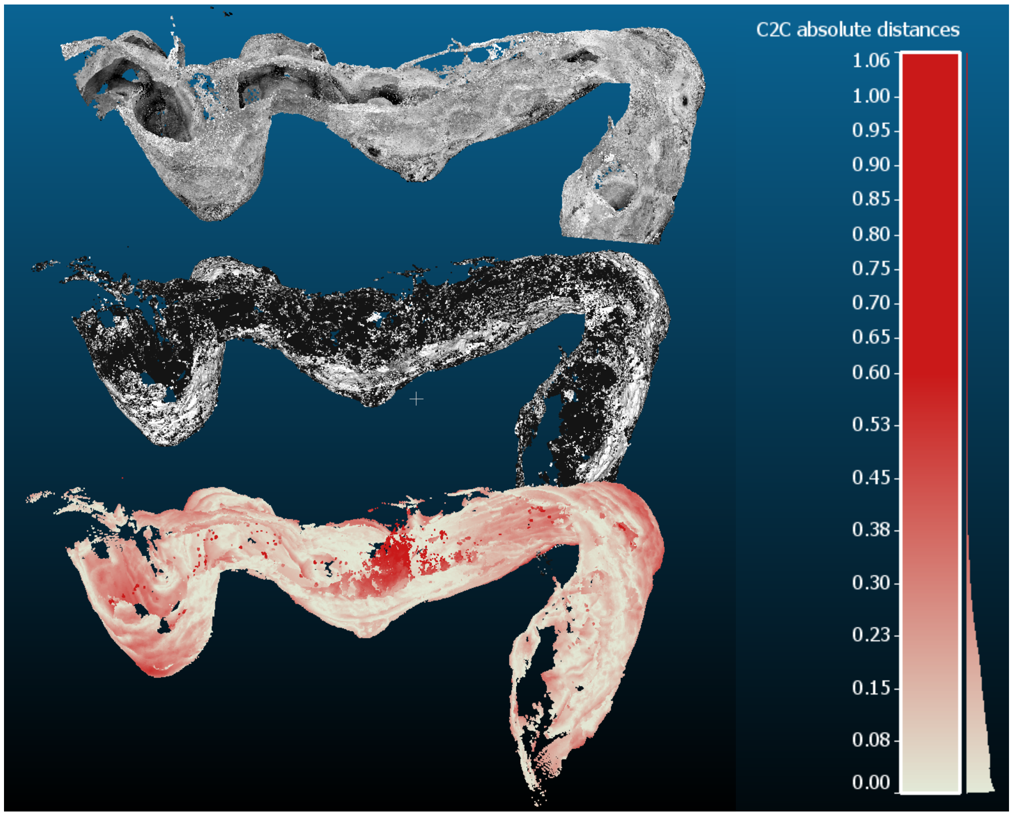


Figure 4.8: From top to bottom: The LiDAR scan of the mud cave, the SLAM scan using RTAB-MAP and Kinect v2, and the same SLAM scan with point-to-point error with the LiDAR scan highlighted (darker shade of red = larger error). The scale indicates the error in meters.

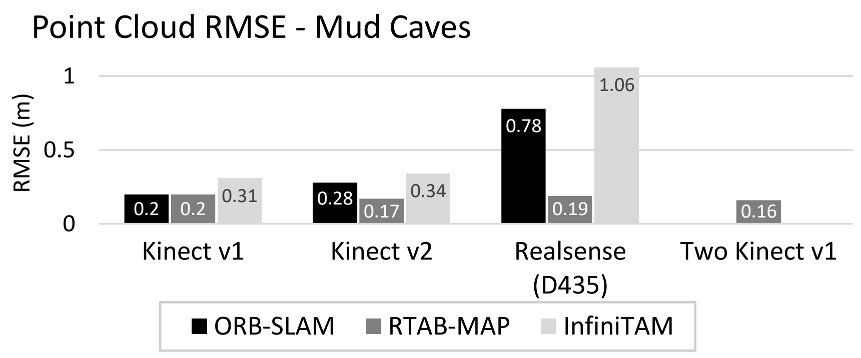


Figure 4.9: RMSE of the cloud-to-cloud difference between SLAM algorithm results and the LiDAR cloud, for the mud caves site.

experiments. In the middle is the point cloud obtained from scanning the cave with the Kinect v2 and the RTAB-MAP algorithm. First, we aligned these models globally using ICP. Then, for each point in the SLAM point cloud, we computed the distance to the nearest neighbor in the reference LiDAR scan. The bottom model shows these point-to- point distances colored on a color scale from white (small distance) to red (large distance). In general, we notice more errors accumulated where the cave presents a turn. This is mainly due to two reasons. First, more drift is accumulated in areas with less visibility because the sensor has a more limited view of environment features. The other reason is that turns are places where sensor noise tends to accumulate. The depth quality of RGB-D cameras decreases with the distance, therefore distant obstacles can accumulate the noise from multiple scans.

We summarize the cloud-to-cloud differences for each combination of algorithm and sensor by computing the RMSE over the entire point cloud. The results are presented in Figure 4.9. We can observe a similar error in multiple models using Kinect v1 and v2 on ORB-SLAM and RTAB-MAP. InfiniTAM v2 tends to give a larger error, as expected from an algorithm without loop closure detection. The Realsense camera tends to show a less consistent, larger error. This is due to the technology employed to generate depth data. Stereo matching can be noisier, especially as distance from the sensor increases. Intel provides a number of software filtering options for this camera, and we can expect better results when fine-tuning the process for this specific application. We also compare the results to a setup using two Kinect v1 cameras with RTAB-MAP. The model in this case presents a better RMSE as the software can utilize more sensor data to refine its trajectory calculation.

Trajectory difference

For each trajectory output from the SLAM algorithms, we recovered the corresponding ground-truth trajectory using the LiDAR data. We measured the Absolute Trajectory Error (ATE) between the two outputs in two different ways: 1) by aligning the first pose of each

Table 4.1: Absolute Trajectory Error (ATE) and Relative Position Error (RPE) on different combinations of sensors and algorithms. *ATE* compares the trajectories aligned on the first pose only. *ATE (aligned)* compares trajectories globally aligned. *RPE* measures the relative error per 1 meter segments.

	ORB-SLAM		**RTAB-MAP**		**InfiniTAM v2**	
	ATE (aligned)	RPE	ATE (aligned)	RPE	ATE (aligned)	RPE
Kinect v1	0.64 (0.26)	0.04	0.98 (0.33)	0.05	2.10 (0.59)	0.05
Kinect v2	1.08 (0.36)	0.08	0.70 (0.31)	0.06	1.21 (0.56)	0.10
Realsense (D435)	5.24 (2.20)	0.25	1.49 (0.32)	0.11	-	-
Two Kinect v1	-	-	0.87 (0.25)	0.07	-	-

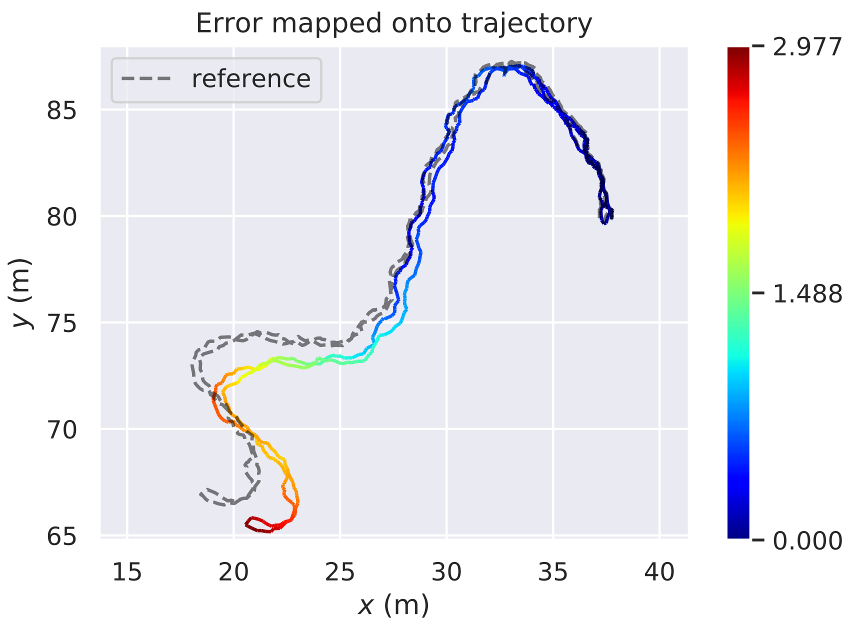


Figure 4.10: Trajectory comparison between RTAB-MAP with Kinect v1 and the recovered ground truth. The position error is plotted on the SLAM trajectory in meters.

trajectory—this gives an idea of how much error the algorithm accumulates with respect to the starting position, and 2) by aligning the whole trajectory to minimize the average error—this informs about the average error over the entire scan.

Figure 4.10 shows an example of the ATE between a trajectory from RTAB-MAP with Kinect v1 and the reference output, with respect to the starting position. We summarize the results in Table 4.1. The InfiniTAM results with the Realsense camera are not included, as they are too noisy to provide a good reconstruction. Generally, the results from the Kinect cameras are similar. Despite possessing a global shutter camera, the Kinect v2 does not show a consistent improvement, however we believe that it is possible to improve the results by using a higher resolution than the default used in most algorithms. The Realsense camera creates more noise, and this translates to a less accurate trajectory. The InfiniTAM algorithm accumulates a lot of error over time as its tracking is based on depth data only. This is clearly reflected by a much worse ATE (large global error), but still keeping a similar RPE as other algorithms (good local consistency).

Loop Closure Analysis

While loop closure is a crucial step of most SLAM algorithms in order to increase the accuracy and reduce the drift, loops are very rare in tunnels and excavations. In this type of environment, one possible way to trigger a loop closure is to turn around, travel back to a previous point in the trajectory, and turn around again. This last turn around allows the algorithm to recognize a visited place by matching similar visual features. We analyze the difference between closing the loop and not closing it. We run the sensor data through the ORB-SLAM algorithm, once keeping the loop closure, and once stopping before backtracking. Figure 4.11 shows the profiles of the trajectories with and without loop closure, and the recovered ground truth. Both SLAM trajectories present a small drift over time with respect to the ground truth, but the drift is clearly higher without loop closure.

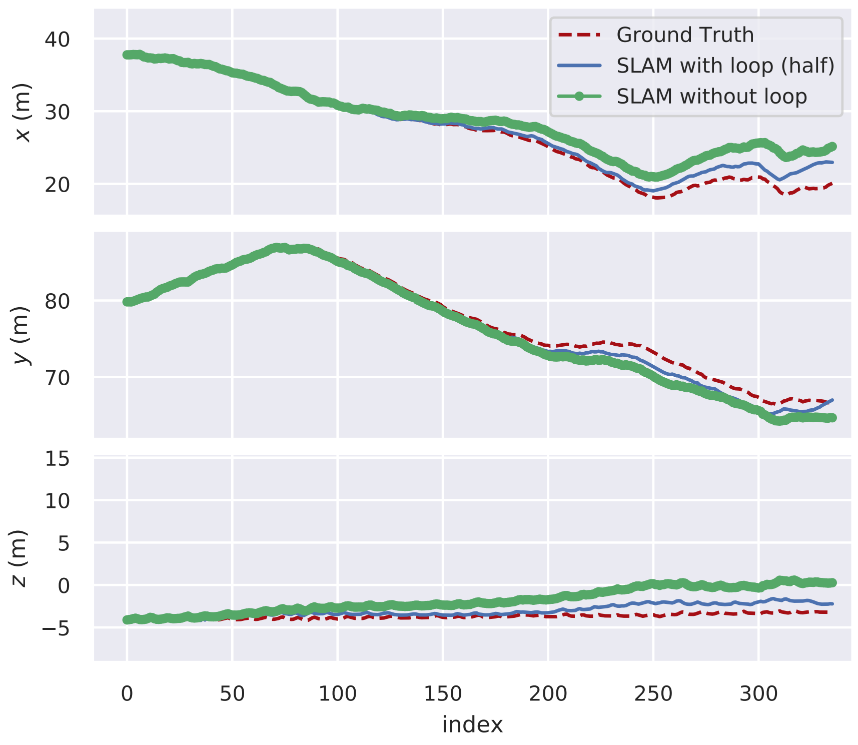


Figure 4.11: Comparison of trajectories with and without loop closure using RTAB-MAP with Kinect v1. The trajectory is plotted in meters for each axis (the z axis is aligned to gravity), against the camera frame index.

4.4.2 El Zotz Archaeological Site

We tested our scanning setup in a cultural heritage context at the archaeological site of El Zotz, Guatemala, and report our results here. Due to various constraints of the field, not all of our experiments were executed in the exact same conditions. For this reason, we replayed the saved camera data through the different algorithms to increase the consistency. We collected data with the Kinect v1 and v2 cameras. Due to algorithm failure, the Kinect v2 data do not perform any loop closure. We also collected data with a Realsense ZR300 camera, however the results are too noisy to be properly analyzed here.

We also compared the SLAM models to a LiDAR scan taken at a previous time, and therefore the models can differ at certain specific areas since excavation conditions changed as archaeologists expanded new tunnels and backfilled other branches. The cloud-to-cloud difference is higher for this reason, and can only be used as a comparison metric between SLAM algorithms and sensors. The trajectory reconstruction has been manually adjusted to ignore the areas with differences.

Cloud-to-cloud difference

Figure 4.12 shows an error plot of a SLAM model against the LiDAR model. The areas of high errors mostly correspond to the differences between the scans. Otherwise we notice a similar trend with the mud caves models, where certain walls accumulate noise from sensor. This is more noticeable with ORB-SLAM as the sensor data are simply merged together without any kind of filtering.

Figure 4.13 shows the summary of RMSE for the two cameras. ORB-SLAM with Kinect v2 fails to reconstruct, but otherwise presents better results for Kinect v1. In this case, the Kinect v2 gives better results even though there is no loop closure. This environment is more difficult and scans are very shaky, which can be handled better by the global shutter camera.

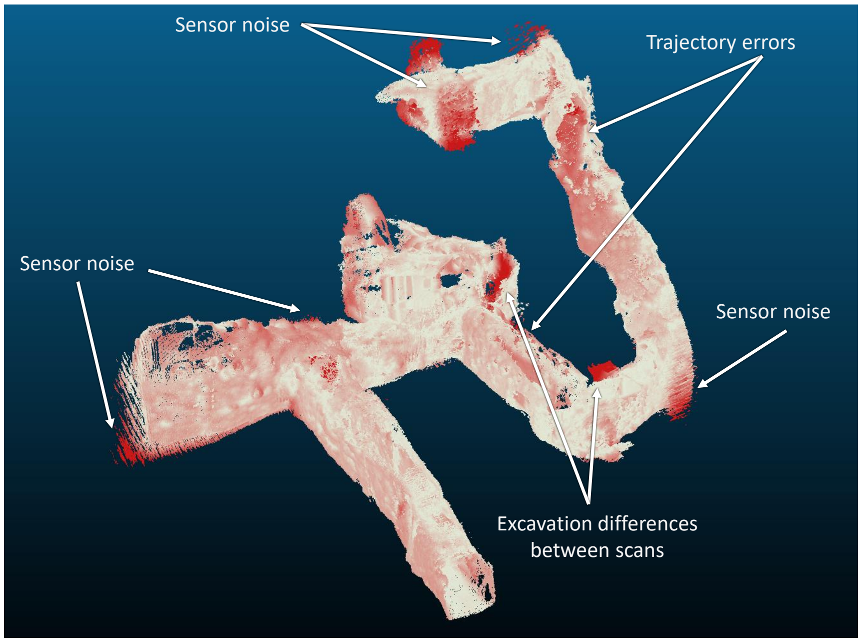


Figure 4.12: Point-to-point difference between the point cloud from ORB-SLAM with Kinect v1 and the LiDAR point cloud. Darker red corresponds to a larger error, which happens in areas with concentrated sensor noise, and areas where the two models differ in geometry.

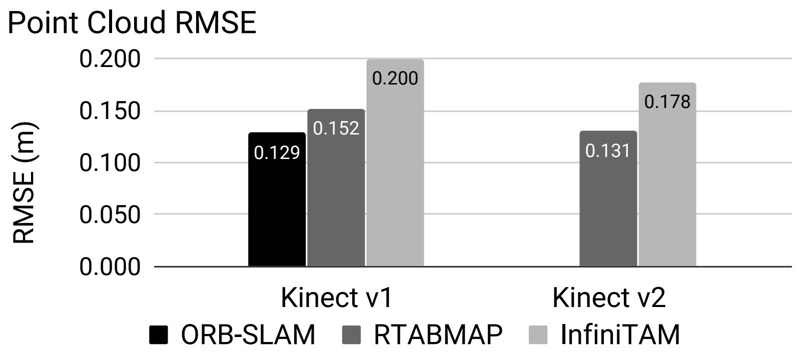
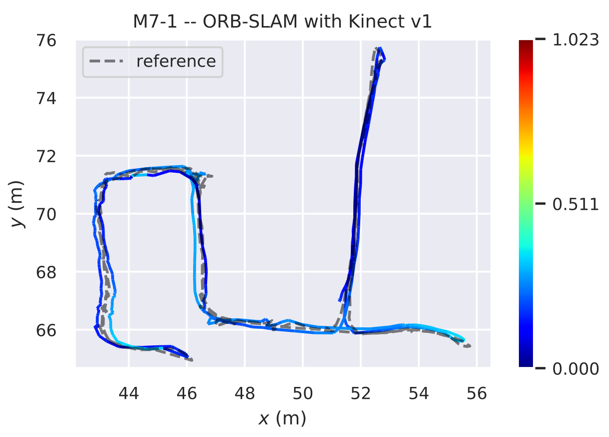
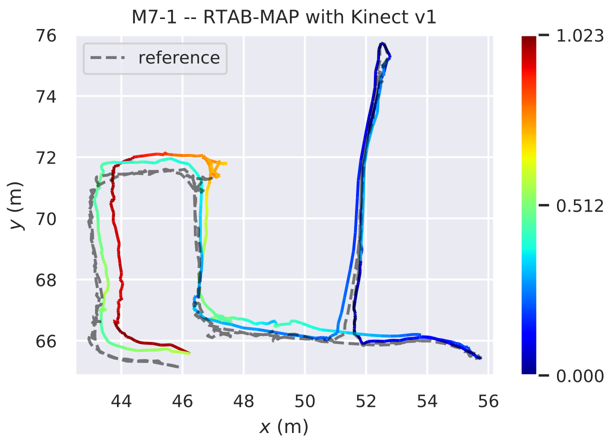


Figure 4.13: RMSE of the cloud-to-cloud difference between SLAM algorithm results and the LiDAR cloud, for the M7-1 excavation site.



(a) Trajectory error using ORB-SLAM with Kinect v1.



(b) Trajectory error using RTAB-MAP with Kinect v1.

Figure 4.14: Top view of the trajectory comparison between SLAM results and the recovered ground truth. The position error is plotted on the SLAM trajectory in meters.

Table 4.2: ATE and RPE in meters, on the M7-1 excavation.

	ORB-SLAM		**RTAB-MAP**		**InfiniTAM v2**	
	ATE (aligned)	RPE	ATE (aligned)	RPE	ATE (aligned)	RPE
Kinect v1	0.21 (0.19)	3.32*	0.47 (0.29)	0.06	2.27 (2.49)	0.08
Kinect v2	-	-	0.93 (0.17)	0.08	1.10 (0.47)	0.10

* The trajectory has a large discontinuity causing a very large value for RPE.

Trajectory difference

We recovered a reference trajectory based on the LiDAR data for our camera scans. Figure 4.14 shows the trajectory errors from ORB-SLAM and RTAB-MAP with Kinect v1. In this case, the RTAB-MAP algorithm has accumulated more drift than ORB-SLAM. We summarize the trajectory metrics in Table 4.2. The results are similar to what we measured in the mud caves environment, with InfiniTAM results being worse in global consistency compared to the image-based tracking algorithms.

4.4.3 Virtual Reality Model

The combination of LiDAR, SfM, and reconstructions from SLAM algorithms results in a Virtual Reality model, according to the combination instructions presented in the previous section and in the aforementioned wiki. Screenshots of the VR world can be seen in Figures 4.15 and 4.16, and the entire models, including video demonstrations and raw Unreal Engine files can be viewed at the following link: `https://sites.google.com/ucsd.edu/maya-archeology-xr/vr-development/vr-demostrations`.

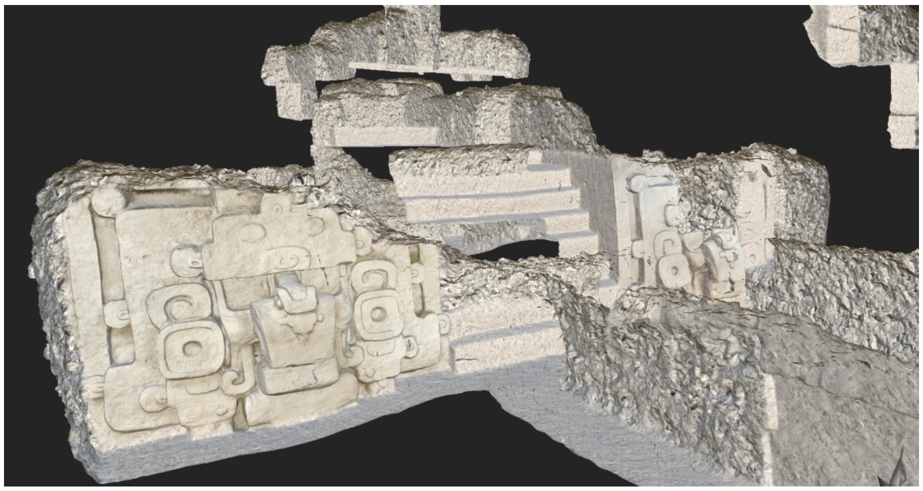


Figure 4.15: The Final m7-1 model composed of several sub-objects

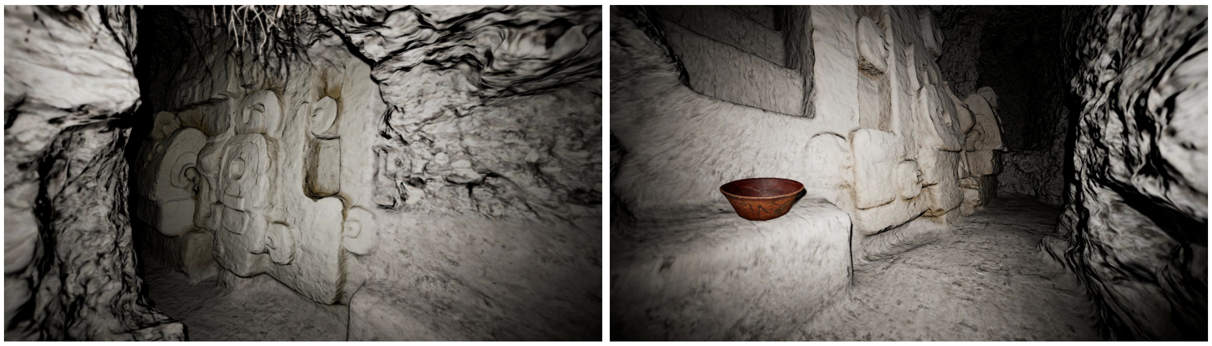


Figure 4.16: Resulting Game Asset for the Maya Archaeology VR Application utilizing Physical-based materials and Optimized Geometry in Unreal Engine.

4.5 SLAM Results Analysis

In this Section, we discuss high-level trends and directions to take to maximize the accuracy of SLAM-based scanning, based on the results presented in Section 4.4 and our observations in the field.

4.5.1 Sensors

The RBG-D cameras that we used are broadly divided into two categories: active depth sensing (Kinect v1 and v2), and passive depth sensing (Realsense). Both our results and our observations confirm that active sensing produce the best results in terms of 3D scans. The difference in technology for active sensing (structured light with Kinect v1 and time-of-flight with Kinect v2) has a less significant impact on the result, although there tends to be a slight global improvement with the time-of-flight sensor. Stereo-based sensors produce depth maps that tend to be spatially and temporally noisy, which increases the drift in SLAM algorithms, and also increases the frequency of tracking loss during data collection. The ZR300 camera produced results too noisy to be analyzed in Section 4.4.2. However, it is important to note that the active sensors can only operate indoors, and would not be suitable to scan the exteriors of sites. Additionally, the stereo sensors can be improved in software through the fine-tuning of filters, although this increases the complexity of the scanning system.

4.5.2 Algorithms

The algorithms can also be divided in two categories: feature-based tracking (ORB-SLAM and RTAB-MAP), and registration-based tracking (InfiniTAM). Feature-based algorithms mostly rely on image features, and therefore can more easily recover from tracking failure. However, these algorithms cannot recognize the same tunnel viewed from a different point-of-view (e.g., after turning around), and must rely on loop closure to create a consistent model in that case.

Registration-based algorithms rely on depth data only, and as such can recognize an environment based on its geometry only. These algorithms tend to be less stable in terms of tracking as they rely primarily on frame-to-frame alignment. InfiniTAM v3 is a good compromise, but our experiments failed to properly scan on long distances. While InfiniTAM v2 can be used for scanning without much tweaking, InfiniTAM v3 would require more tuning to be useable in these conditions. Feature-based algorithms worked better in our experiments, but they did require particular attention to the lighting of the scene. Our scanning system is equipped with a light panel emitting diffuse light, which minimizes the negative effects on the SLAM algorithm, however a strong, static light is recommended whenever possible.

4.5.3 Scanning Procedure

In Section 4.4, we highlight the importance of creating a loop closure, even when the environment presents no obvious such loop. This procedure applies to feature-based algorithms, or algorithms capable of handling loop closure, but is still useful in all cases simply to add new points-of-view of the scene and increase the number of details. However, due to the nature of these real-time algorithms, finding a loop closure can sometimes lead to a tracking failure, which forces the user to start over. Techniques for recovering the tracking in real-time are improving and the next generation of algorithms might solve this problem. There are also several good practices that we have noted during our experiments. First, each depth sensor has its own specifications in terms of sensing range and field of view. These limitations are very important for the user to keep in mind, as they generally make the difference between a failed scan and a good scan. Typically, there should always be a recognizably distinctive portion of the scene (containing visual features and geometric features) within the nominal range of the sensor (especially above the minimum range). Second, the scanning speed is important. While generally faster than other techniques, SLAM-based scanning is sensitive to fast motion which introduces errors from: motion blur, lens distortion from rolling shutter cameras (minimized by using a global shutter camera such as

Kinect v2), more distance between frames (minimized by using higher frequency cameras such as the Realsense D435), or simply breaking the small motion assumption made by registration-based algorithms.

4.6 Conclusion

We have analyzed multiple hardware and software solutions to help with the documentation of cultural heritage within restricted spaces. Through multiple experiments in the field, we have established an accuracy value for different combinations of sensors and algorithms in these low-light conditions. In terms of software, image-based tracking algorithms tend to perform better for large-scale area scanning, but may offer a sparser 3D model without post-processing. Dense SLAM algorithms relying on depth tracking offer a good solution for small areas but do not scale well, unless implemented with loop-closure solutions. In terms of sensors, active depth sensing solutions targeted at indoor use perform better than stereo-based cameras in an underground environment. We have found that in average, the best performing combination of sensor and algorithm for our SLAM prototype scanning system is a Kinect v2 with RTAB-MAP. This result can be generalized to new generations of cameras with similar technology (global shutter with a time-of-flight depth sensor), which, when running a feature-based SLAM algorithm, are likely to perform better.

SLAM algorithms provide a low-cost alternative to LiDAR, and a fast alternative to photogrammetry methods. They are generally limited in the quality of results: in the presented use-case of documenting archaeological underground excavations, the best system can produce about 20 cm of error in average after alignment, and up to 50 cm in some cases. The proposed system is nevertheless a good solution for the rapid documentation of areas with extensive damage from looting where having an expensive instrument in difficult field conditions is impractical. Likewise, such a system has cultural heritage documentation applications in high-risk areas where archaeological sites are damaged or threatened in order to make rapid assessments of impacts and resource requirements.

We have also presented a method of multi-sensor registration optimized for presenting archaeological tunnel excavations in virtual reality. By combining data from high-accuracy low-usability sensors and techniques such as LiDAR and SfM with middling-accuracy and high-usability sensors and techniques such as RGBD cameras and SLAM, we can create representations of these sites that are not only useful for archaeologists, but also for the general public to understand the sites' historical and cultural implications.

Cutting edge advancements in SLAM such as Dot3D and the Intel RealSense L515 handheld LiDAR may pave the way to further incorporating pointclouds from highly-usable sensors and not interfere with accurate reconstructions from larger LiDARs and SfM. Preliminary data collection efforts to test the effectiveness of the L515 as a platform and Dot3D as a method for SLAM have been conducted, and future expeditions will evaluate their utility.

4.7 Acknowledgments

Chapter 4, in part is currently being prepared for submission for publication of the material. Tueller, Peter; Vindiola, Giovanni; Garrison, Thomas; Schurgers, Curt; Kastner, Ryan. The dissertation/thesis author was the primary investigator and author of this material. Chapter 4, also in part, is a reprint of the material as it appears the Journal of Cultural Heritage Management and Sustainable Development 2020. Gautier, Quentin; Garrison, Thomas; Rushton, Ferrill; Bouck, Nicholas; Lo, Eric; Tueller, Peter; Schurgers, Curt; Kastner, Ryan.

Chapter 5

FishSense: RGBD Imaging for Underwater Fish Detection and Assessment

In this chapter we present the usage of 3D data capture with regards to underwater imaging of fish for detection and health assessment. Once again, we follow the generic workflow presented in Figure 2.1. We explore step 1, which in this case comprises the usage of an RGBD camera underwater, by developing and validating an underwater imaging platform. In this case, step 2 is handled by the software of the platform to align RGB and depth images individually, but as we are interested in the fish themselves, and not necessarily with the environment as a whole or the temporal and spatial relationship of the fish, we are satisfied with collecting and analyzing the RGBD images individually. Similarly, for step 3, the only context required is the deployment location of data collection, which for the moment is manually entered, but the platform can be extended in the future to incorporate additional sensors that may aid the detection and measurement process. Finally, step 4 is explored extensively with the development of machine learning algorithms to detect the fish and calculate the length and geometry of the fish.

5.1 Introduction

Our ocean ecosystems are facing a problem of enormous proportions: overfishing. Despite estimates that 3.5 trillion fish are swimming throughout the ocean, ecologists estimate that the planet will be devoid of fish as early as 2048 [WBB$^+$06]. This chilling prediction is backed by fishing data as well: approximately 50% of the global fish population is harvested for human consumption *every year* [MB10]. These figures do not even include the illegal fishing market that comprises an estimated 18% of the yearly catch [APP$^+$09]. Such a drastic loss in biodiversity will not only impact the health and quality of our marine ecosystems, but will also starve nearly three billion people who depend on fish for survival.

In order to combat this daunting problem, scalable, yet simple, sensing systems are desperately needed to measure fish populations; specifically their biomass and health. Measurements are predominantly gathered using direct human observation: divers conduct manual censuses of fish populations and observers survey populations from above. Other solutions, such as catch & release, are invasive, often requiring fish to be removed from the sea for measurement. [Str93][WSS06]. These methods greatly affect fish health and stress levels, which ultimately negatively impact their growth rate[Ash07]. Research of non-invasive methods for measuring free-swimming fish has yielded solutions that require underwater imaging and machine learning techniques.

Effective methods for underwater imaging are absolutely vital in retrieving valuable data. Typical methods include stereo vision [PFA$^+$18], laser-based systems [RWT17] [ARO15], or acoustic-based systems [LZKW16]. These methods can be computationally complex and require intricate computer vision algorithms. There are also methods which have been presented which correct the effects of light scattering using light stripe range scanning and photometric stereo [NN05]. Alternatively, the Sea-thru method attempted to reconstruct the color in images through a process of removing the water from underwater images [AT19]. Other methods have used light emitters in conjunction with cameras to estimate depth in images. One such method proposed a scanning with a digital camera and laser that operated along a railed system [ARO15]. Another method proposed included the use of a structured light depth camera used for 3D underwater capture, which also employed a calibration method that accounted for underwater refraction [DCT+16], though this system did not have sufficient range for capturing fish. Many of these systems are unwieldy, large, or too difficult for divers to operate, making them problematic for sustainable use.

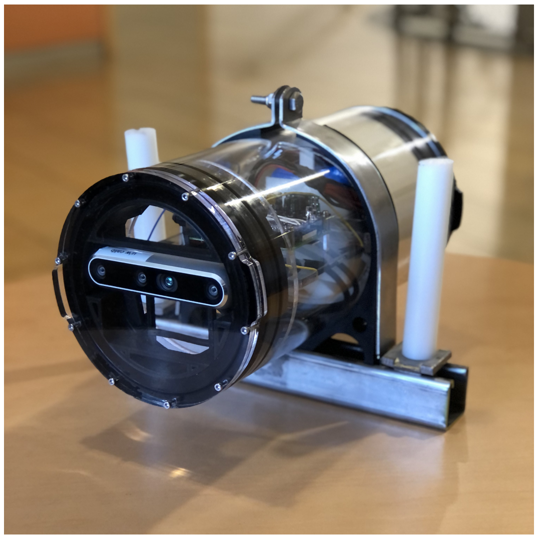


Figure 5.1: FishSense imaging platform

In this work, we propose FishSense: a new handheld underwater RGBD imaging platform that measures fish length. This system can be used to derive fish biomass [Ful04] and health. The FishSense core technologies include depth cameras, compute platforms, software, and a sleek mechanical design. FishSense utilizes scientific advancements in underwater imaging and machine learning to collect and process data. This integrated technology provides scientists with meaningful data to advance scientific predictions about fish health and populations. Our main contributions are:

1. A low-profile diver-operated platform for free-swimming fish imaging, as seen in Fig. 5.1;

2. An implementation of a machine learning detector for identifying fish in RGBD images;

3. An implementation of a fish length measuring algorithm for RGBD images.

The remainder of this paper is organized as follows. Section II provides an overview of the FishSense platform detailing the mechanical and electrical interfaces, as well as the Intel RealSense imaging device. Section III delves into the machine learning application with relation to fish detection and measurement. Section IV addresses the collected data. Finally, Section V provides the conclusion.

5.2 Platform

The FishSense design is a 12 inch long, 6 inch diameter cylindrical enclosure which stores our depth camera, compute platform, and batteries. The Intel RealSense D455 Depth Camera is mounted on one end, the four battery packs are on the other, and the Raspberry Pi 4 and switch system are located in between (see Figure 5.2).

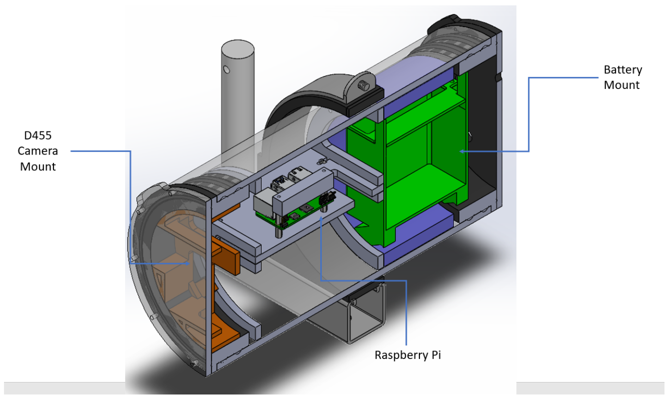


Figure 5.2: FishSense mechanical prototype cross-section

5.2.1 Mechanical

A small, handheld system allows for easy portability, storage, and modularity. Our design allows for multiple configurations without changes to the housing. For example, the system can be easily converted into a camera trap, underwater vehicle payload, aquaculture cage monitor, or simply as an attachment to another piece of equipment. Mounting our device for any application is made simple and straightforward due to the multiple directions of screw holes.

Our device is fully optimized for underwater functionality. The system is well equipped to handle underwater deployments, with a depth rating of 65 meters, handles for divers, and magnetic switches for power and recording. Even with its sealed, waterproof design, it has four blank ports on the back plate, which enable wired connections into the system. With these ports, users can stream the camera feed to a topside computer, add an extended power bank for longer deployments, or simply charge the batteries without needing to open the housing.

5.2.2 Electrical

Our initial prototype uses a Raspberry Pi 4 with the Intel RealSense D455 and a 1TB SSD both connected via USB 3.0. It is powered by four 4S 3500 mAh lithium ion battery packs. With a fully charged pack, RealSense depth and color videos can run at 1-5 fps for approximately 12 h. As seen in Fig. 5.3, the battery is connected to a power module that distributes power to the reed switches, LEDs, and Pi.

To power on FishSense, a magnet needs to be applied to a reed switch. The walls of the enclosure are thin enough to allow for even a small magnet to activate the switch, which enables the user to power on the FishSense system without the need to open the watertight enclosure. It also allows for faster setup time and makes it more practical for amateur use. Another reed switch is used to start and stop the data collection, and a simple LED array alerts the user to the status of the system.

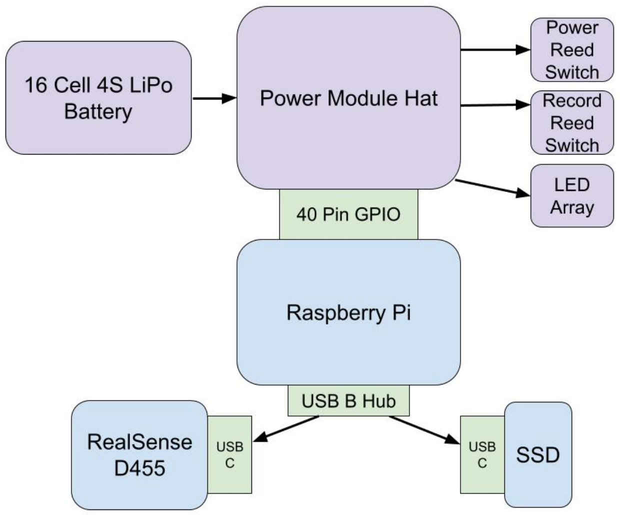


Figure 5.3: FishSense wiring diagram

5.2.3 Intel RealSense D455

This project utilizes the Intel RealSense D455 Depth Camera. RealSense camera captures RGB images with a global shutter and a resolution up to 1280 x 800 at frame rate of 30 fps. The stereoscopic depth images have a resolution up to 1280 x 700 and a frame rate up to 90 fps. The RealSense camera uses an infrared emitter to project a pattern so that the stereoscopic depth cameras can easily reconstruct a depth image. The RealSense camera also contains the Intel RealSense Vision Processor D4 for vision processing to align the RGB and depth images together and capture a 3D view of the world. The Intel RealSense D455 camera was selected because of it's ability to perform with dynamic movement and for it's simplicity. Our camera has been calibrated for underwater use, which will allow us to capture and construct the entire scene with the D4 processor. This is important because our system is dependant on the accuracy of the camera, and it needs to construct the 3D view at the correct scale. The RealSense camera is positioned at one end of the FishSense system to capture images of fish in a desired location. The camera was used in conjunction with the Intel RealSense SDK 2.0.

5.3 Fish Detection and Measurement

Complementary to underwater imaging, techniques must be utilized to evaluate the collected data for extended use. A binary image classifier was used in [MFS$^+$18] where Regions of Interest (RoI) were identified in data. Another source attempts to classify fish via an Artificial Neural Network, a supervised learning algorithm [PWS19]. The machine learning method in [VRSM19] utilized the Speeded-Up Robust Features (SURF) and Support Vector Machines (SVM) algorithms along with a bag of features for simplicity. Specific to RGBD imaging, there are certain algorithms that utilize both the RGB and D components of the image simultaneously to perform detection, such as in [WN18][ZFC$^+$21][TJUM20]. While all algorithms have benefits, the YOLOv4 model has been tested and time-proven to be the best when accounting for both detection speed and accuracy. Also, because YOLOv4 operates solely on the RGB component of the image, we can leverage existing image datasets of fish. The relative newness of RGBD imaging to underwater systems means that there are practically no open source datasets of fish, and thusly, YOLOv4 is the model we chose for our system.

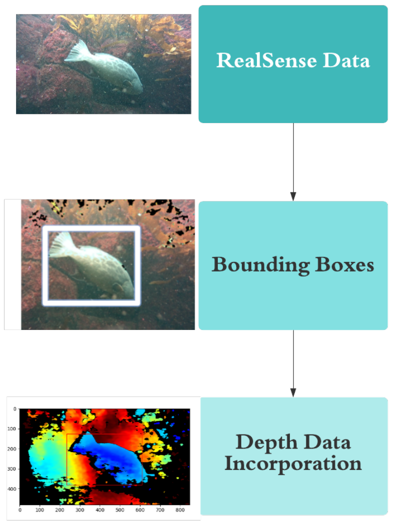


Figure 5.4: Process of data interpretation

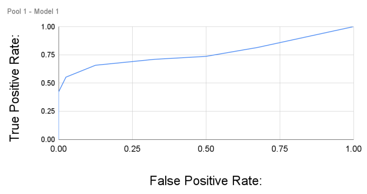
The camera captures two pieces of information per frame. One being a standard RGB image, the other being a comma separated value (csv) file containing the distance of each pixel from the camera (i.e. depth image). Initially our machine learning algorithms work on the RGB image, detecting potential fish, and then the pixel values denoting the bounding box around each fish are recorded. From this bounding box, pixels at the tip of the head and the end of the tail are selected, and the corresponding depth associated with those pixels are retrieved from the depth image. These pixels are then projected into 3D space by using the intrinsic parameters of the RealSense depth camera, and the Euclidean distance between the points are measured to determine the length of the detected fish. This process can be seen at a high level in Figure 5.4.

The performance of machine learning model is tested using Mean Average Precision, as well as ROC curves. The machine learning algorithm uses a test dataset that was 10% of the training dataset. This test set is only used to measure the accuracy of the predicted bounding boxes, against human annotated bounding boxes. The average loss is also calculated, and used along the mAP to ensure learning was accurate. Once the models finish training, they are then tested in a controlled pool with a toy fish, as well as an aquarium with many species of fish to generate ROC curves.

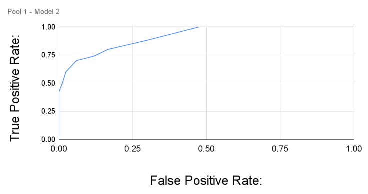
The performance of the length measurement component of our process is assessed by analyzing all the true detections of the toy fish in our controlled pool tests. The average and standard deviation of length measurements are computed and compared to the ground truth length of the toy fish.

5.4 Analysis

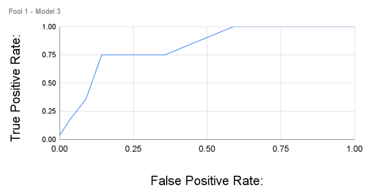
For training the models, three distinct data sets were used. The first set of weights were trained off a subset of fish within Google's Open Image Dataset [KRA$^+$20] and the OzFish Dataset [Aus20] that were not colorful or exotic i.e Cuttlefish, Lionfish, Betafish etc. This allowed for the weights to standardize to ”normal” or ”regional” fish, which allows our system to perform



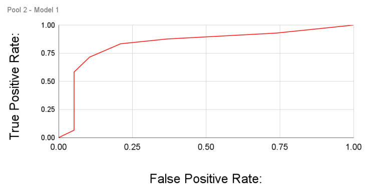
(a) Region based model in Single fish tests



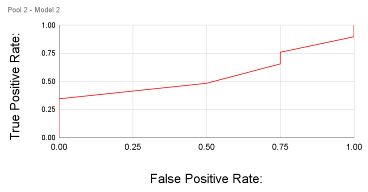
(b) Species specific model in Single fish tests



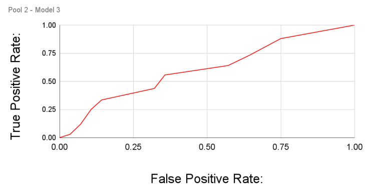
(c) General use model in Single fish tests



(d) Region based model in Multiple fish tests



(e) Species specific model in Multiple fish tests



(f) General use model in Multiple fish tests

Figure 5.5: ROC curves for three models over two pool tests

much better in mounted scenarios, where we can predict which species will be present. The second set of weights were trained off images of our toy fish in various scenarios. This allows our system to perform better a single species application like aquaculture. The last set of weights were trained off of the entirety of the Open Image Dataset and OzFish Dataset. These weights provided a baseline for our system would perform in open ocean scenarios, where we might encounter a wide variety of fish. All three of these weights were tested against two different data sets. The first being our toy fish in a controlled test pool, and the other being various species of fish in an aquarium.

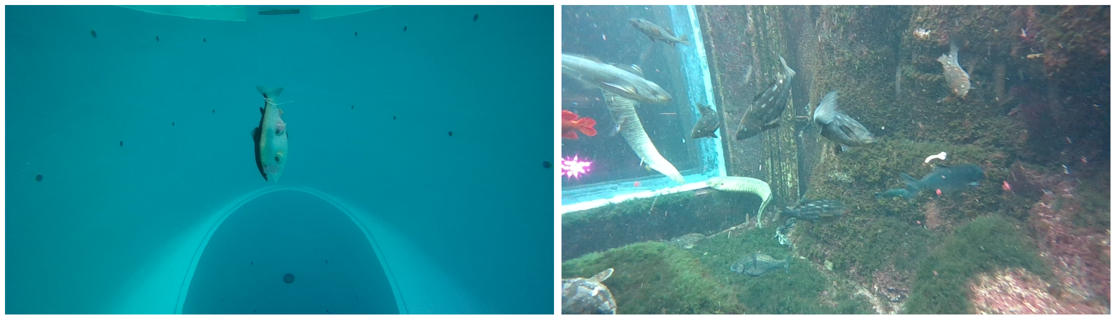


Figure 5.6: Sample images from the Pool 1 (top) and Pool 2 (bottom) datasets

The ROC curves shown in Figure 5.5 demonstrate the functionality and trade offs of each set of weights. Model 1 is the set of weights used for usage in mounted scenarios with predictable ”regional” species. Model 2 is the set of weights trained for usage in single species detection. Model 3 is for general use, trained to detect all fishes. Pool 1 had a single toy fish in a controlled test pool, whereas Pool 2 is a 70,000 gallon aquarium that contains many different fish, such as leopard sharks, moray eels, Giant Black Sea Bass, Garibaldi, kelp bass, opaleye, and other fish native to Southern California. Sample images from each of these pool datasets can be seen in Figure 5.6. Model 1 performed well with a high threshold in both datasets, as despite detecting false positives, it confidently detects real fish with a high deal of accuracy (Figures 5.5a and 5.5d). Unsurprisingly, Model 2 performed extremely well in the Pool 1 dataset, as seen in Figure 5.5b, as it was trained off of the specific fish used in that experiment, but fell short when confronted with many different species in the Pool 2 dataset, as seen in Figure 5.5e. This tells us that there is a danger to focusing the model too much on a single type of fish and encountering a fish outside of the training dataset. Finally, it can be seen that Model 3, the general model, performed decently in the single fish test (Figure 5.5c) and performed average in the multiple fish test(Figure 5.5f). This tells us that a general model can be useful when imaging unknown species of fish, but that there is opportunity to improve detection when there is knowledge of the ”regional” type of fish, and especially if there is interest only in a single species of fish.

The toy fish that is used in the Pool 1 deployments was measured by a tape measure and found to be 31.5cm. When using the depth images over all the true positive detections of that fish in the Pool 1 dataset, we found the average length to be 31.83cm, with a standard deviation of 1.950cm. This result is extremely encouraging, and suggests FishSense is a valid instrument for length calculations underwater. We observed the error in the measurement to come from two sources. First, the depth images occasionally had blur, which made the fish seem longer or wider than it actually was, or on rare occasions, missing chunks, which affected the quality of point selection. The second source of error was the orientation of the fish in the image. If the fish was not in clear profile, then it was harder to pick the same points on the fish consistently.

5.5 Conclusion

In this work, we have outlined FishSense, an underwater RGBD imaging platform for fish detection and fish length measurement. We have experimentally validated our system with three detection models and two underwater datasets, and have shown it to accurately detect and measure fish.

Although our initial prototype has demonstrated feasibility in many of our intended applications, future iterations will improve the efficiency of the system and allow it to extend into new applications. To achieve longer deployments, an increase in computing power, storage capacity, and battery count are essential. Integrating a higher resolution color camera allows for better image analysis and species classification. Also, the addition of an LCD screen for depth map display, system diagnostics, and a settings menu will make underwater use easier and open up other possibilities during underwater operations.

Our current compute platform is limited by the bandwidth of its USB controller. More specifically, it cannot simultaneous read from the RealSense and write to the SSD at 30 fps using the same USB controller. As such, we performed our experiments at a reduced framerate, but to solve this problem, future compute platforms will utilize more powerful hardware to perform full throughput video recording. More computing power will additionally enable the possibility of real-time data processing on-board the device. Currently, an NVIDIA Jetson TX2 processor serves as the catalyst for these solutions. With a higher bandwidth USB controller and SATA compatibility, we expect to save data at the full frame rates the RealSense offers, along with exploring more advanced real-time processing.

Another opportunity for improvement lies with the detection model. It currently performs detections with RGB data independent of depth data; however, by incorporating the depth data into the detections to create particular regions of interest (ROI), detection accuracy should increase dramatically. Since most fish will be in within some range of the camera, by setting boundaries and parsing through the depth image, various ROIs could be created. These regions might include a lot of false positives, as it would pickup rocks, kelp etc. However, if these ROIs are combined with detectors, and even with the actual neural network architecture these false positives can be accounted for. Likewise, the depth data could also be incorporated into preprocessing techniques to counteract the hazing that occurs in water. By using the RealSense SDK this is a lot more intuitive, and can bolster detector accuracy as well the increase the viability of above water data.

5.6 Acknowledgments

Chapter 5, in part, is a reprint of the material as it appears in IEEE/MTS OCEANS 2021. Tueller, Peter; Maddukuri, Raghav; Paxson, Patrick; Suresh, Vivaswat; Ashok, Arjun; Bland, Madison; Wallace, Ronana; Guerrero, Julia; Semmens, Brice; Kastner, Ryan. The dissertation/thesis author was the primary investigator and author of this material. Chapter 5, also, in part is currently being prepared for submission for publication of the material. Tueller, Peter; Maddukuri, Raghav; Paxson, Patrick; Suresh, Vivaswat; Miao, Albert; Drews, Donovan; Paxson, Charles; Semmens, Brice; Kastner, Ryan. “FishSense: 3D Capture and Analysis for Fish Detection and Stock Assessment”.

Chapter 6

Conclusion

This dissertation addresses the problem of capturing 3D data of remote environments. Remote environments present a significant challenge in terms of access and observability, and yet they contain important information that have cultural and health-related global impacts. We present a generic workflow for obtaining, processing, and utilizing 3D data. We also present three remote environments and explore their usage of the workflow and conduct experiments optimizing one or more components of the workflow.

First, we present an overview of the generic workflow for acquiring, processing, and presenting 3D data. We then present a discussion on the first step of the workflow: an enumeration and description of 3D sensors that are typically used in remote environments and their specific benefits. These sensors are the fundamentals on which the remaining chapters build their contributions.

We present the first remote environment example: target detection on the seafloor using synthetic aperture sonar. We present experiments surrounding the final step of the workflow, and show that interpretation of the data can be optimized using feature detectors. We also present a method for optimizing parameters of the feature detectors for SAS imagery with different backgrounds.

We then present the second remote environment example: Mayan archaeological excavations in the Petén region of Guatemala. We explore all aspects of the workflow in this example, from the selection of sensors to accommodate the unique challenges of the environments, to an analysis of intra-sensor registration through SLAM, to the challenges of coherently combining sensor data, to the quality of results and the usefulness of 3D data in an archaeological context.

Finally, we present the third remote environment example: underwater imaging of fish using RGBD-stereo cameras. We present the development of an underwater platform that samples and collects 3D data. We also present experiments regarding the final step of the workflow and examine machine learning algorithms to robustly detect the fish and length measurement algorithms to derive the information important to fisheries science researchers.

Overall, the research presented by this dissertation improves the usage of 3D sensors in remote environments in multiple instances. It provides contributions to the oceanic engineering, fisheries, archaeology, embedded systems, and computer vision fields.

Bibliography

[AD18] Avi Abu and Roee Diamant. Unsupervised local spatial mixture segmentation of underwater objects in sonar images. *IEEE Journal of Oceanic Engineering*, (99):1–19, 2018.

[AKB08] Motilal Agrawal, Kurt Konolige, and Morten Rufus Blas. Censure: Center surround extremas for realtime feature detection and matching. In *European Conference on Computer Vision*, pages 102–115. Springer, 2008.

[APP$^+$09] David J. Agnew, John Pearce, Ganapathiraju Pramod, Tom Peatman, Reg Watson, John R. Beddington, and Tony J. Pitcher. Estimating the worldwide extent of illegal fishing. *PLOS ONE*, 4(2):1–8, 02 2009.

[AR19] Abu Avi and Diamant Roee. A statistically-based method for the detection of underwater objects in sonar imagery. *IEEE Sensors Journal*, 2019.

[ARO15] C Almansa, L Reig, and J Oca. The laser scanner is a reliable method to estimate the biomass of a senegalese sole (solea senegalensis) population in a tank. *Aquacultural engineering*, 69:78–83, 2015.

[Ash07] Paul J. Ashley. Fish welfare: Current issues in aquaculture. *Applied Animal Behaviour Science*, 104(3-4):199–235, 2007.

[AT19] Derya Akkaynak and Tali Treibitz. Sea-thru: A method for removing water from underwater images. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[Aus20] Australian Institute Of Marine Science. OzFish Dataset - Machine Learning Dataset For Baited Remote Underwater Video Stations, 2020. Type: dataset.

[Bar14] Christopher M Barngrover. *Automated Detection of Mine-Like Objects in Side Scan Sonar Imagery*. PhD thesis, UC San Diego, 2014.

[BAYG17] G. Bresson, Z. Alsayed, L. Yu, and S. Glaser. Simultaneous localization and mapping: A survey of current trends in autonomous driving. *IEEE Transactions on Intelligent Vehicles*, 2(3):194–220, Sep. 2017.

[BNZ+16] Bruno Bodin, Luigi Nardi, M Zeeshan Zia, Harry Wagstaff, Govind Sreekar Shenoy, Murali Emani, John Mawer, Christos Kotselidis, Andy Nisbet, Mikel Lujan, et al. Integrating algorithmic parameters into benchmarking and design space exploration in 3d scene understanding. In *2016 International Conference on Parallel Architecture and Compilation Techniques (PACT)*, pages 57–69. IEEE, 2016.

[Bra00] G. Bradski. The OpenCV Library. *Dr. Dobb's Journal of Software Tools*, 2000.

[BTVG06] Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. Surf: Speeded up robust features. In *European conference on computer vision*, pages 404–417. Springer, 2006.

[CLM14] Gaspar Muñoz Cosme, Cristina Vidal Lorenzo, and Alessandro Merlo. La acrópolis de chilonché (guatemala): Crónica de las investigaciones de un patrimonio en riesgo en el área maya. *Restauro Archeologico*, 22(2):99–115, 2014.

[DABP14] Piotr Dollár, Ron Appel, Serge Belongie, and Pietro Perona. Fast feature pyramids for object detection. *IEEE transactions on pattern analysis and machine intelligence*, 36(8):1532–1545, 2014.

[DB06] H. Durrant-Whyte and T. Bailey. Simultaneous localization and mapping: part i. *IEEE Robotics Automation Magazine*, 13(2):99–110, June 2006.

[DCT+16] Sundara Tejaswi Digumarti, Gaurav Chaurasia, Aparna Taneja, Roland Siegwart, Amber Thomas, and Paul Beardsley. Underwater 3d capture using a low-cost commercial depth camera. In *2016 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 1–9. IEEE, 2016.

[DFSAB+18] Andrea Di Filippo, Luis Javier Sánchez-Aparicio, Salvatore Barba, José Antonio Martín-Jiménez, Rocío Mora, and Diego González Aguilera. Use of a wearable mobile laser system in seamless indoor 3d mapping of a complex historical site. *Remote Sensing*, 10(12), 2018.

[Far16] EM Farella. 3d mapping of underground environments with a hand-held laser scanner. In *Proc. SIFET annual conference*, 2016.

[FFADT13] Francesco Fassi, Luigi Fregonese, Sebastiano Ackermann, and V De Troia. Comparison between laser scanning and automated 3d modelling techniques to reconstruct complex and extensive cultural heritage areas. *ISPRS - International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, XL-5/W1, 02 2013.

[FFML13] Maurice F Fallon, John Folkesson, Hunter McClelland, and John J Leonard. Relocating underwater features autonomously using sonar-based slam. *IEEE Journal of Oceanic Engineering*, 38(3):500–513, 2013.

[Ful04] TW Fulton. The rate of growth of fishes. twentysecond annual report. *Fisheries Board of Scotland. Retrieved from https://ci. nii. ac. jp/naid/10030255872*, 1904.

[FZG$^+$17] C. Forster, Z. Zhang, M. Gassner, M. Werlberger, and D. Scaramuzza. Svo: Semidirect visual odometry for monocular and multicamera systems. *IEEE Transactions on Robotics*, 33(2):249–265, April 2017.

[GLU12] A. Geiger, P. Lenz, and R. Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pages 3354–3361, June 2012.

[GRN$^+$16] Thomas G Garrison, Dustin Richmond, Perry Naughton, Eric Lo, Sabrina Trinh, Zachary Barnes, Albert Lin, Curt Schurgers, Ryan Kastner, and Sarah E Newman. Tunnel vision: documenting excavations in three dimensions with lidar technology. *Advances in Archaeological Practice*, 4(2):192–204, 2016.

[HCC$^+$15] Wu-Chih Hu, Chao-Ho Chen, Tsong-Yi Chen, Deng-Yuan Huang, and Zong-Che Wu. Moving object detection and tracking from video captured by moving camera. *Journal of Visual Communication and Image Representation*, 30:164–180, 2015.

[HG09] Michael P Hayes and Peter T Gough. Synthetic aperture sonar: A review of current status. *IEEE Journal of Oceanic Engineering*, 34(3):207–224, 2009.

[HKK15] A. Huletski, D. Kartashov, and K. Krinkin. Evaluation of the modern visual slam methods. In *2015 Artificial Intelligence and Natural Language and Information Extraction, Social Media and Web Search FRUCT Conference (AINL-ISMW FRUCT)*, pages 19–25, Nov 2015.

[HS88] Chris Harris and Mike Stephens. A combined corner and edge detector. In *Alvey vision conference*, volume 15, pages 10–5244. Citeseer, 1988.

[Hug90] Q Huggett. Long-range underwater photography in the deep ocean. *Marine Geological Surveying and Sampling*, pages 69–81, 1990.

[IA17] I. Z. Ibragimov and I. M. Afanasyev. Comparison of ros-based visual slam methods in homogeneous indoor environment. In *2017 14th Workshop on Positioning, Navigation and Communications (WPNC)*, pages 1–6, Oct 2017.

[IMS$^+$08] Engin Ipek, Sally A McKee, Karan Singh, Rich Caruana, Bronis R de Supinski, and Martin Schulz. Efficient architectural design space exploration via predictive modeling. *ACM Transactions on Architecture and Code Optimization (TACO)*, 4(4):1, 2008.

[JKE$^+$10] Hordur Johannsson, Michael Kaess, Brendan Englot, Franz Hover, and John Leonard. Imaging sonar-aided navigation for autonomous underwater harbor surveillance. In *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on*, pages 4396–4403. IEEE, 2010.

[KNS+17] Marek Kraft, Michał Nowicki, Adam Schmidt, Michał Fularz, and Piotr Skrzypczyński. Toward evaluation of visual navigation algorithms on rgb-d data from the first- and second-generation kinect. *Machine Vision and Applications*, 28(1):61–74, Feb 2017.

[KRA+20] Alina Kuznetsova, Hassan Rom, Neil Alldrin, Jasper Uijlings, Ivan Krasin, Jordi Pont-Tuset, Shahab Kamali, Stefan Popov, Matteo Malloci, Alexander Kolesnikov, et al. The open images dataset v4. *International Journal of Computer Vision*, 128(7):1956–1981, 2020.

[KVSMK11] Stefan Kohlbrecher, Oskar Von Stryk, Johannes Meyer, and Uwe Klingauf. A flexible and scalable slam system with full 3d motion estimation. In *2011 IEEE International Symposium on Safety, Security, and Rescue Robotics*, pages 155–160. IEEE, 2011.

[LC13] Hung-Yi Liu and Luca P Carloni. On learning-based methods for design-space exploration with high-level synthesis. In *Proceedings of the 50th annual design automation conference*, page 50. ACM, 2013.

[LLZH14] R. Li, J. Liu, L. Zhang, and Y. Hang. Lidar/mems imu integrated navigation (slam) method for a small uav in indoor environments. In *2014 DGON Inertial Sensors and Systems (ISS)*, pages 1–15, Sep. 2014.

[LMFS16] Max Leingartner, Johannes Maurer, Alexander Ferrein, and Gerald Steinbauer. Evaluation of sensors and mapping approaches for disasters in tunnels. *Journal of Field Robotics*, 33(8):1037–1057, 2016.

[Low04] David G Lowe. Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60(2):91–110, 2004.

[LWZ11] Jianguo Li, Tao Wang, and Yimin Zhang. Face detection using surf cascade. In *2011 IEEE international conference on computer vision workshops (ICCV workshops)*, pages 2183–2190. IEEE, 2011.

[LZKW16] D-Q Lin, H Zhang, M Kang, and Q-W Wei. Measuring fish length and assessing behaviour in a high-biodiversity reach of the upper yangtze river using an acoustic camera and echo sounder. *Journal of Applied Ichthyology*, 32(6):1072–1079, 2016.

[MAGK16] Pingfan Meng, Alric Althoff, Quentin Gautier, and Ryan Kastner. Adaptive threshold non-pareto elimination: Re-thinking machine learning for system level design space exploration on fpgas. In *Proceedings of the 2016 Conference on Design, Automation & Test in Europe*, pages 918–923. EDA Consortium, 2016.

[MAT17] R. Mur-Artal and J. D. Tardós. Orb-slam2: An open-source slam system for monocular, stereo, and rgb-d cameras. *IEEE Transactions on Robotics*, 33(5):1255–1262, Oct 2017.

[MB10] Alison Mood and Phil Brooke. Estimating the number of fish caught in global fishing each year. *Fishcount.¡ En línea*, 2010.

[MF10] Vincent Myers and John Fawcett. A template matching procedure for automatic target recognition in synthetic aperture sonar imagery. *IEEE Signal Processing Letters*, 17(7):683–686, 2010.

[MFG$^+$17] A. Masiero, F. Fissore, A. Guarnieri, M. Piragnolo, and A. Vettore. Comparison of Low Cost Photogrammetric Survey with Tls and Leica Pegasus Backpack 3d Modelss. *ISPRS - International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, pages 147–153, November 2017.

[MFS$^+$18] Simone MariniVimal, Emanuela Fanelli, Valerio Sbragaglia, Ernesto Azzurro, Joaquin Del Rio Fernandez, and Jacopo Aguzzi. Tracking fish abundance by underwater image recognition. *Scientific Reports*, 8, 2018.

[MHVL11] Thierry Mulder, Heiko Hüneke, and AJ Van Loon. Progress in deep-sea sedimentology. In *Developments in Sedimentology*, volume 63, pages 1–24. Elsevier, 2011.

[MSK$^+$11] Johannes Meyer, Paul Schnitzspan, Stefan Kohlbrecher, Karen Petersen, Mykhaylo Andriluka, Oliver Schwahn, Uwe Klingauf, Stefan Roth, Bernt Schiele, and Oskar von Stryk. A semantic world model for urban search and rescue based on heterogeneous sensors. In Javier Ruiz-del Solar, Eric Chown, and Paul G. Plöger, editors, *RoboCup 2010: Robot Soccer World Cup XIV*, pages 180–193, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg.

[MTM10] Michael May, Martin J Turner, and Tim Morris. Scale invariant feature transform: a graphical parameter analysis. In *Proc. of the British Machine Vision Conf.(BMVC) 2010 UK Postgraduate Workshop*, pages 5–1. BMVA Press, 2010.

[NMR$^+$17] Erica Nocerino, Fabio Menna, Fabio Remondino, Isabella Toschi, and Pablo Rodríguez-Gonzálvez. Investigation of indoor and outdoor performance of two portable mobile mapping systems. In Fabio Remondino and Mark R. Shortis, editors, *Videometrics, Range Imaging, and Applications XIV*, volume 10332, pages 10332 – 10332 – 15. SPIE, jun 2017.

[NN05] Srinivasa G. Narasimhan and Shree K. Nayar. Structured light methods for underwater imaging: Light stripe scanning and photometric stereo. *Proceedings of OCEANS 2005 MTS/IEEE*, 3, 2005.

[Ope18] OpenCV. STAR Class Description. `https://docs.opencv.org/2.4/modules/features2d/doc/common\_interfaces\_of\_feature\_detectors.html#StarFeatureDetector\%20:\%20public\%20FeatureDetector`, 2018. Online; accessed 18-April-2018.

[PFA$^+$18] Daniel Pérez, Francisco J Ferrero, Ignacio Alvarez, Marta Valledor, and Juan C Campo. Automatic measurement of fish size using stereo vision. In *2018 IEEE international instrumentation and measurement technology conference (I2MTC)*, pages 1–6. IEEE, 2018.

[PSZ09] Gianluca Palermo, Cristina Silvano, and Vittorio Zaccaria. Respir: a response surface-based pareto iterative refinement for application-specific design space exploration. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 28(12):1816–1829, 2009.

[PWS19] Ricardus Anggi Pramunendar, Sunu Wibirama, and Paulus Insap Santosa. Fish classification based on underwater image interpolation and back-propagation neural network. *International Conference on Science and Technology*, 5:1–6, 2019.

[RD06] Edward Rosten and Tom Drummond. Machine learning for high-speed corner detection. In *European conference on computer vision*, pages 430–443. Springer, 2006.

[RRKB11] Ethan Rublee, Vincent Rabaud, Kurt Konolige, and Gary Bradski. Orb: An efficient alternative to sift or surf. In *Computer Vision (ICCV), 2011 IEEE international conference on*, pages 2564–2571. IEEE, 2011.

[RRNT06] David Ribas, Pere Ridao, Jose Neira, and Juan D Tardos. Slam using an imaging sonar for partially structured underwater environments. In *Intelligent Robots and Systems, 2006 IEEE/RSJ International Conference on*, pages 5040–5045. IEEE, 2006.

[RRTN08] David Ribas, Pere Ridao, Juan Domingo Tardós, and José Neira. Underwater slam in man-made structured environments. *Journal of Field Robotics*, 25(11-12):898–921, 2008.

[RWT17] Austin A Rizzo, Stuart A Welsh, and Patricia A Thompson. A paired-laser photogrammetric method for in situ length measurement of benthic fishes. *North American Journal of Fisheries Management*, 37(1):16–22, 2017.

[S$^+$94] Jianbo Shi et al. Good features to track. In *Computer Vision and Pattern Recognition, 1994. Proceedings CVPR'94., 1994 IEEE Computer Society Conference on*, pages 593–600. IEEE, 1994.

[SC86] Randall C. Smith and Peter Cheeseman. On the representation and estimation of spatial uncertainty. *The International Journal of Robotics Research*, 5(4):56–68, 1986.

[SEE$^+$12] J. Sturm, N. Engelhard, F. Endres, W. Burgard, and D. Cremers. A benchmark for the evaluation of rgb-d slam systems. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 573–580, Oct 2012.

[SSCL20] Giuseppe Schirripa Spagnolo, Lorenzo Cozzella, and Fabio Leccese. Underwater optical wireless communications: Overview. *Sensors*, 20(8), 2020.

[Str93] NJC Strachan. Length measurement of fish by computer vision. *Computers and electronics in agriculture*, 8(2):93–104, 1993.

[TJUM20] Masahiro Takahashi, Yonghoon Ji, Kazunori Umeda, and Alessandro Moro. Expandable yolo: 3d object detection from rgb-d images. In *2020 21st International Conference on Research and Education in Mechatronics (REM)*, pages 1–5. IEEE, 2020.

[TKD18] Peter Tueller, Ryan Kastner, and Roee Diamant. A comparison of feature detectors for underwater sonar imagery. In *OCEANS 2018 MTS/IEEE Charleston*, pages 1–6. IEEE, 2018.

[VRSM19] M. Vimal Raj and S. Sakthivel Murugan. Underwater image classification using machine learning technique. *International Symposium on Ocean Technology*, pages 166–173, 2019.

[WBB$^+$06] Boris Worm, Edward B. Barbier, Nicola Beaumont, J. Emmett Duffy, Carl Folke, Benjamin S. Halpern, Jeremy B. C. Jackson, Heike K. Lotze, Fiorenza Micheli, Stephen R. Palumbi, Enric Sala, Kimberley A. Selkoe, John J. Stachowicz, and Reg Watson. Impacts of biodiversity loss on ocean ecosystem services. *Science*, 314(5800):787–790, 2006.

[WDDW01] Stefan Williams, Gamini Dissanayake, and Hugh Durrant-Whyte. Towards terrain-aided navigation for underwater robotics. *Advanced Robotics*, 15(5):533–549, 2001.

[WN18] Weiyue Wang and Ulrich Neumann. Depth-aware cnn for rgb-d segmentation. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 135–150, 2018.

[WSS06] Darren J White, C Svellingen, and Norval JC Strachan. Automated measurement of species and length of fish by computer vision. *Fisheries Research*, 80(2-3):203–210, 2006.

[Yas07] Naci Yastikli. Documentation of cultural heritage using digital photogrammetry and laser scanning. *Journal of Cultural Heritage*, 8(4):423 – 427, 2007.

[ZBG$^+$14] Robert Zlot, Michael Bosse, Kelly Greenop, Zbigniew Jarzab, Emily Juckes, and Jonathan Roberts. Efficiently capturing large, complex cultural heritage sites with a handheld mobile 3d laser mapping system. *Journal of Cultural Heritage*, 15(6):670 – 678, 2014.

[ZFC$^+$21] Tao Zhou, Deng-Ping Fan, Ming-Ming Cheng, Jianbing Shen, and Ling Shao. Rgb-d salient object detection: A survey. *Computational Visual Media*, 7(1):37–69, 2021.

[ZMM$^+$15] S. Zennaro, M. Munaro, S. Milani, P. Zanuttigh, A. Bernardi, S. Ghidoni, and E. Menegatti. Performance evaluation of the 1st and 2nd generation kinect for multimedia applications. In *2015 IEEE International Conference on Multimedia and Expo (ICME)*, pages 1–6, June 2015.

[ZN13] Wan-Lei Zhao and Chong-Wah Ngo. Flip-invariant sift for copy and object detection. *IEEE Transactions on Image Processing*, 22(3):980–991, 2013.