SELF-DRIVING OCEAN RESEARCH VESSELS TO MEASURE GLACIER RETREAT

PREPARED FOR

ROSS: THE ROBOTIC OCEANOGRAPHIC SURFACE SAMPLER

JONATHAN NASH _____________ *Signature* _____ *Date*

PREPARED BY

GROUP 46

TEAM NAME

CHRISTOPHER PATENAUDE _____________ *Signature* _____ *Date*

DONALD (MAX) HARKINS _____________ *Signature* _____ *Date*

GREGORY SANCHEZ _____________ *Signature* _____ *Date*

MICHAEL GABRIEL _____________ *Signature* _____ *Date*

TOBIAS HODGES _____________ *Signature* _____ *Date*

Abstract

This document represents the full documentation of our Capstone experience. It describes the system we created to detect and report obstacles in a marine environment. The system was created for Jonathan Nash to use on board remotely-operated research vessels.

Contents

1 Introduction

Our project was requested by Dr. Jonathan Nash, to help prevent collisions for a remotely-operated reaserch vessel. The vessel, called the Robotic Oceanographic Surface Sampler (ROSS), is used to research oceanographic processes. From experiences on past expeditions, Dr. Nash recognized the need to develop a computerized system to look for obstacles near the ROSS in order to prevent potential collisions. The system helps remote operators of the research vessel operate the boat in a safe manner. Preventing collisions is of utmost importance to prevent economic loss in the form of destruction of instruments on the ROSS, and preventing harm to people and equipment that are potentially are involved in a collision.

Our clients were Dr. Nash, the head of research for the ROSS, and Jasmine Nahorniak, the head of programming for the ROSS. Both of our clients were extremely helpful in the completion of our project. The team members on the project were Chris Patenaude, Michael Gabriel, Greg Sanchez, (Donald) Max Harkins, and Tobias Hodges.Team members focused on different components of the project for development. Chris Patenaude focused on interfacing with existing ROSS positional sensor to capture data about the ROSS, Greg Sanchez focused on using positional sensor data to correct captured images for ocean movement, D. Max Harkins focused on detecting objects in captured images and tracking objects over time, and Michael Gabriel focused on exporting serialized headings to detected obstacles. All team members worked together on system design and documentation.

Our clients provided support and advice during development. Our clients took did not participate with development, but gave us recommendations and feedback about our project when we asked. We had weekly meetings with our clients, to keep them informed on the state of the project and what progress we were making. Our clients helped greatly during Spring Term by working with us remotely to do system installation and integration testing. Because of the COVID pandemic, we were not able to test our system as thoroughly as we hoped. Our plan was to have our system included on board the ROSS for an open water test, to test the system capabilities in a real-world environment. However, we were only able to test the system off the ROSS and are unaware of the true object detection capabilities of the system.

For future developers, we recommend looking at the System Overview section to get more information about the project. The System Overview section provides the most concise, relevant information about our project's code. Additionally, it contains next steps that could be taken for development, and describes challenges our team faced that we hope future development is able to mitigate.

2 Requirements Document

Contents

1 CHANGE TABLE

Date	Change
Feb 12th, 2020	Creating a stereo camera system was no longer feasible due to complication in synchronizing the web cameras. We are now looking into buying a stereo camera.
Feb 12th, 2020	Stereo camera has been dropped due to price and complexity. The project will now use a single camera system.
Feb 14th, 2020	Depth map creation no longer continued.
Feb 14th, 2020	We are no longer trying to track the distance of objects. We will only be tracking the location.
Feb 14th, 2020	Local path-finding portion of the project has been removed in favor of updating the client UI.
Feb 24th, 2020	Client UI potion of the project has been removed. We will now be sending a list of located objects to the Client system.

2 INTRODUCTION

This section provides a content overview of the Requirements Document

2.1 Purpose

The Autonomous Vessel Vision System (AVVS) will be developed to allow oceanographic research vessels to detect and recognize obstacles in the ocean. Implementing an obstacle detection system would allow unmanned research vessels to operate with greater autonomy. Increased autonomy and obstacle avoidance would make unmanned research boats safer to small vessels and people in the research boats' proximity, decrease the risk of damage to the research boat due to a collision with a larger object, and reduce the amount of human oversight needed during its operation.

2.2 Scope

AVVS will allow remotely operated ocean vessels to recognize objects on or near the vessel's path using a single digital camera. The AVVS will be integrated into the Garmin navigation system of a remote operated Kodiak raft and trained to recognize floating objects. When objects are detected, AVVS will report the object to the existing computer control system, which will send the data to a remote system on the mother ship monitoring the ocean vessel.

2.3 Product overview

2.3.1 Product context

Our system will be implemented on an Autonomous Research Vessel (ARV) and will consist of a single camera mounted facing the front of the boat. Our system will access positional data about the boat by interfacing with an IMU (Inertial Measurement Unit) sensor on board the vessel and shared across multiple systems. Additionally, our system will output the position of objects to the client's system, so they can be sent remotely to observers on a mother ship near by.

2.3.2 Product functions

The obstacle avoidance system shall gather data from its surroundings via a single forward facing camera. The system shall process video data in order to identify object on or near the vessels path. Operational information shall be sent back and fourth between the obstacle avoidance system and the operator. The system shall report the heading to all detected obstacles to the operator, so they can make informed course decisions.

2.3.3 User characteristics

Users will be able to enable or disable AVVS through Long Range Radio and the Iridium satellite communicator already on the autonomous vessel. The long-range communication will be handles by the client's system, so the AVVS must specifically be able to change states based on a serialized information input. If AVVS encounters an error with obstacle detection, a signal will be sent to the users through long range communication using the client's existing system.

2.4 Definitions, acronyms, and abbreviations

TABLE 1
Definitions

Term	Definition
AVVS	Autonomous Vessel Vision System
AVR	Autonomous Research Vessel
IMU	Inertial Measurement Unit

3 SPECIFIC REQUIREMENTS

3.1 User interface

Users will be given a report of detected objects at one second intervals to be displayed to personnel controlling the remotely operated boat. Additionally, a debugging display will be provided which displays images taken from the camera with boxes around the detected objects via the desktop environment of the computer on which the system is run.

Additionally, the system will be interfaceable through a configuration file present on the computer where the computer vision software component runs. This configuration file will allow the users of the system to modify parameters to adjust responses to detected objects as fits the use case.

3.2 Functional requirements

This section describes core functional components of AVVS

3.2.1 Function: Video Processing

Description: The AVVS must be able to process a video stream in real time.

Input: Live video feed

Output: A collection of images

Dependencies: None

3.2.2 Function: Object Recognition

Description: The AVVS must be able to identify floating objects on the surface of the ocean that are likely to block AVVS's host vessel.

Input: A collection of images

Output: Data including whether an object was identified and its relative location to the vessel using AVVS.

Dependencies: Video Processing

3.2.3 Function: Object Detection Reporting

Description: The AVVS must be able to report obstacles to the client system when obstacles are detected

Input: Meta data collected from the Object Recognition System describing detected obstacles

Output: A serialized data file accessible by the client's system.

Dependencies: Object Recognition

3.3 Performance requirements

The system must classify objects in real time, using the existing low-powered on-board computer with an Intel integrated i3-6100U. The computer to be used also runs other critical operation control software, so must not be slowed down by running our system. Therefore, our system must be configurable in terms of the speed at which processing occurs, and be as efficient and light weight as possible.

3.4 System operations

The system must respond to visual input from the camera. The system must be able to recognize large and small boats, people, and debris in the water as obstacles.

After initial object recognition, the system must be able to track the object over time. The system must be able to determine how the heading to the object changes over time.

After identifying the object and determining how the object is moving, the system must report the object to the user. Periodically, after a captured image is processed, the system must serialize data about the objects for use by the client's system, thus allowing operators controlling the vessel to react to obstacles in front of the boat.

3.5 System modes and states

The system will contain different modes during its traversal to a location. Here is a list of the following modes the system will support:

- Stopped
- Running

When the system is stopped, it will not capture nor process images from the camera, and will not provide output to the client's system. When the system is running, it will take periodic input from the camera and IMU, process the input, and periodically output the position of all objects being tracked to the client system.

3.6 Physical characteristics

The system must use a digital camera as input to the computer system. The digital camera must be placed facing forward, and have a clear field of view so it is able to detect all obstacles in the camera's field of view in front of the boat. The software component of the system will be run on an Intel NUC, and share processing resources with critical systems control software. The computer hardware of the NUC consists of an Intel i3-6100U running at 2.3 GHz, with 4 GB of memory.

Our system will interface with an external IMU sensor. This sensor provides positional data at a frequency of 10 Hz to the NUC. The sensor data is available in a file, to which binary sensor data is continuously streamed.

3.7 Environmental conditions

Our system must perform in the open ocean. In this environment, the system must contend with a range of weather conditions from fog to rain to bright sun.

Operating conditions at the ocean surface range from minimal waves (flat water) to intense swells (10+ ft). However, our system is not required to be fully operational in all conditions.

Our system must be able to operate during the day, in average conditions. This means our system must be able to operate in mild swells ($>$ 3 ft) and moderate lighting (some fog, but visibility for 100+ feet).

3.8 Software system attributes

We will be developing our system using Python and including any pre-existing modules necessary for the system. Coding conventions shall be based off of PEP8 python coding standards. All code shall be tested to a minimum of 80% line coverage and be checked into a git hub repository; finally, all code shall be peer-examined before being added to the master branch.

4 Verification

This section represents the measure of success for the previous declarations.

4.1 User Interfaces

The user interface will be deemed acceptable when the AVVS state can be changed in under 2 seconds from the systems' reception of a user command, and a working configuration file is present to change avoidance parameters.

4.2 Functional requirements

Individual unit testing will be used to test verify implementation for functional requirements. Additionally, integration testing will be used to verify correct interaction between functional components. The functional requirements will be met when the implemented software passes all tests and performs as intended by the developers.

4.3 Performance requirements

Performance requirements will be met when objects can be detected and tracked in real-time, that is, when changes in an objects motion or position are accounted for within 2 seconds of the event occurring.

The power requirement will be deemed met when the maximum power provided by the vessel is determined, and when the system can operate on without exceeding the maximum power provided by the vessel.

4.4 Environmental requirements

The AVVS must be able to compensate for ocean surface movement of up to 3ft swells and provide similar rates of identification of large objects in the ocean during high swells as well as during flat water. The system must be able to operate in moderate light conditions within 80% effectiveness as during ideal daylight conditions. The system must be able to recover from visual distortion caused by glare, fog, and clouds.

The system will be deemed sufficient when all these requirements are met.

4.5 Usability requirements

Validation of usability requirements will occur when the system can recognize and respond correctly to objects more than 90% of the time.

4.6 Gantt Chart

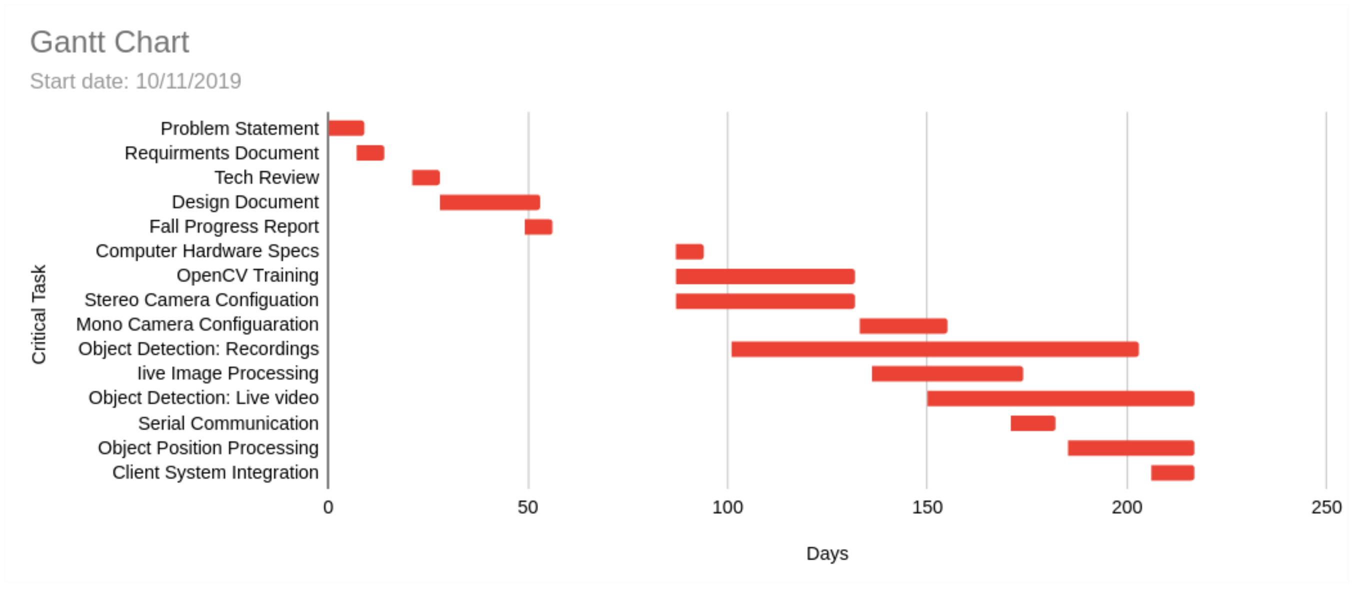


Fig. 1. Final Timeline

3 DESIGN DOCUMENT

Contents

1 Change Table

Date	Change
Feb 12th, 2020	Creating a stereo camera system was no longer feasible due to complication in synchronizing the web cameras. We are now looking into buying a stereo camera.
Feb 12th, 2020	Stereo camera has been dropped due to price and complexity. The project will now use a single camera system.
Feb 12th, 2020	Depth map creation no longer continued.
Feb 14th, 2020	We are no longer trying to track the distance of objects. We will only be tracking the location.
Feb 19th, 2020	We are no longer sending navigational movements to the vessel.
Feb 26th, 2020	Local path-finding portion of the project has been removed in favor of updating the client UI.
Feb 26th, 2020	Client UI potion of the project has been removed. We will now be sending a list of located objects to the Client system.

2 Introduction

This section provides a content overview of the Design Document.

2.1 Purpose

This document describes the design of an object detection system for Unmanned Surface Vessel (Unmanned Surface Vessel (USV)). Such an object detection system would allow for autonomous detection and reporting of various obstacles in open waters such as: other boats, people, and natural obstacles.

2.2 Scope

The object detection system described in this document will be used to detect obstacles in front of the vessel on which the system is deployed. The system will be designed to function on small vessels approximately 12ft in length. The system will be designed to detect objects in daylight conditions. The system will provide relative to North radian locations of detected objects, centered on the USV, to the system operator.

2.3 Context

The object detection system described within this document is intended for use on a small, 12 ft inflatable vessel equipped for oceanographic research. With a sufficiently sophisticated object detection system, a research USV would be able to operate for extended periods of time without direct supervision, allowing for the collection of valuable oceanographic data. Intended uses for our system include research trips to the Gulf of Mexico and Alaska. The use of our system would facilitate collection of data that would inform scientists about ocean processes such as glacial melt at ocean-glacier interfaces. As a preliminary step to fully autonomous obstacle avoidance, our system will implement: image correction, object detection, classification, and reporting to the client system. Future groups shall develop the next stage, intelligent decision making, so the USV can avoid objects without operator guidance.

3 Body

3.1 Identified Stakeholders and Design Concerns

The main stakeholder for this system is Dr. Jonathan Nash. He is overseeing use of the Robotic Oceanographic Surface Sampler (ROSS), a robotic vessel equipped for oceanographic research and the primary vessel on which our system will be deployed. In addition to Dr. Nash, the other major stakeholder in our project is Mrs. Jasmine Nahorniak, who is responsible for the programming of the ROSS and would maintain the obstacle detection system after the end of the Capstone project.

Dr. Nash and Mrs. Nahorniak desire that our system be easy to maintain, require as little processing power as possible, and operate using as few additional hardware components as possible. Additionally, our system must be robust enough to operate in a variety of conditions, and must detect obstacles with greater than 90% accuracy.

3.2 Dedicated Computer Hardware, Camera System, and Communication

Our system will utilize key hardware components to gather/process environmental data, and communicate navigational suggestions to a separate navigation system. We will gather data from a single front mounted digital camera, and an Inertial Measurement Unit. Collected data will be processed using a dedicated computer on board the USV. Serialized navigational suggestions based on the collected data will be reported to the navigation computer.

3.2.1 Purpose

The purpose of the hardware components for this project is to facilitate the software needed to detect objects and avoid them. Our choice of a single digital camera for visual input allows for simple visual data processing, and object recognition done with a neural network. To conserve power, we will run our software on board the same computer used for navigation and control. The USB interface between the camera system and the dedicated computer is simply the standard for communication between host and device. The IMU sensor provides positional data for the boat needed to account for wave motion and to find the heading towards detected obstacles.

3.2.2 Context View

The hardware component of the Object detection System shall be incorporated into a pre-existing USV which navigates using a Garmin Reactor 40 Autopilot System. The USV uses a kodiak inflatable raft as a platform, and an outboard engine as a power source as well as means of transportation. The engine continually charges a 12-volt car battery which provides a stable voltage to the rest of the systems hardware components, including: a NUC computer, a long range radio receiver/transmitter, a Hemisphere Global Positioning System (GPS) antenna, and the Garmin Nav System.

The Object detection System will add a digital camera to the existing system. As mentioned, software will be run on the existing navigation computer to conserve power and reduce complexity. Additionally, the pre-existing IMU will be used by our system. The computer will be powered by the boat's battery, which is charged by the outboard engine. To ensure the best field of view to detect head-on collisions, the camera will be mounted facing forward with a clear view of the water in front of the vessel.

3.2.3 Structure View

The structure of the Object detection System shall consist of the existing computer system, a digital camera, an Inertial Measurement Unit, and hardware interfaces connecting the previous two items to the rest of the USV systems. The existing computer is a PC with hardware components that make up traditional computer systems: Central Processing Unit (CPU), memory, motherboard, power source, hard drive. The computer system must support both USB and Local Area Network (LAN) interface connections for external input and output.

The digital camera will be housed in a waterproof container, to shield it from weather. The camera must have decent visual acuity, which translates to having a resolution of at least 720p. The camera must have a moderately large field of view, around 90 degrees.

3.3 Image processing and object detection

Our system will use computer vision methods to process data from video cameras, and extract information about potential obstacles. The image processing and object detection system will be implemented in Python and use the OpenCV library. For object detection, a deep neural network will be used.

3.3.1 Context View

The image processing and object detection subsystem will take image frames from input video feeds, and use a deep neural network to detect potential obstacles present in the image frame. The deep neural network will output the confidence in the detection, the label of the object detected, and a bounding box describing where in the image the object is present.

Images taken from the USB camera are skewed due to the vessel being at different orientations the moment the image was captured. The different orientations on a boat are rotational motions: roll, pitch, yaw. Each for the x, y and z axis, respectively. Using the positional data from the boat the USB camera the images can be corrected for.

Single Shot Detectors (SSDs) are a deep neural network architecture that can perform in real time with relatively low processing power. Hence, we will use a nerual network with a SSD architecture. The neural network used must have been trained on a dataset composed of marine images, and must be able to classify boats of different sizes and types.

Detecting objects via a deep neural network is ultimately how the system detects potential obstacles. Results from the deep neural network are tracked over time, and reported to the USV operator.

3.3.2 Dependency View

The image processing and object detection component of the system depends on input from a single camera. Images from the camera will be processed individually, and objects will be detected in each image separately. Then, the detected objects will be provided to the object tracking component of the system.

3.4 Object tracking

Detecting objects in discrete video frames does not provide a cohesive picture of potential obstacles. To provide better data, we will use a system to track detected obstacles over time.

3.4.1 Context View

Using the sets of objects from each frame, this section of code correlates bounding boxes of objects in one frame to the next. This will be done by nearest neighbor matching between bounding boxes from different frames. Centroid matching will be done for simplicity, meaning that the center of the bounding boxes will be the data points matched between frames. The distance measure used to match will be euclidean distance [1].

Matched bounding boxes will be assigned an ID, and tracking will occur as long as the system runs. After a tracked bounding box is missing from a specified number of consecutive frames, it will be dropped from the tracking system.

By tracking potential obstacles over time, we will be able to report how the apparent size of the obstacle changes over time, and how the position of the obstacle relative to the boat changes over time.

3.4.2 Dependency View

The object tracking component of the object detection system requires the most recent bounding boxes to be fed to it for each frame processed from the digital camera. Each time an image is captured, corrected, and processed by the deep neural network, bounding boxes will be produced and sent to the object tracking module.

This component of the system produces data to output to the USV operator. Data from this module will be finalized and transmitted out.

3.5 Navigation system communication

The communication module is the link between our object detection system and the navigation control unit which provides object positions based on data collected from our system. Our detection system will be serially communicating data and object positions suggestions to a navigational control unit over a host-to-host LAN connection.

3.5.1 Context View

Our system will use serial communication to transmit data back and forth between our interface and the navigational control unit. Object Positions made by our system will be transmitted to the control unit. We will use a Python library, PySerial, to access the serial ports for data transmission between our system and the control unit. The PySerial library provides access to the port settings and supports read and write functionality [2]. This library will be useful for our project since we will be developing in Python and using a Linux OS which is supported by PySerial.

3.5.2 Dependency View

Communication between our system and the control unit depends on the LAN connection and the object detection data collected from our system. A LAN connection is used for host-to-host communication which can serially send and receive commands from our system and the control unit. Since our system will be responsible for object detection, it will be collecting data about detection of obstacles or other vessels from our camera system and our detection system will determine navigation suggestions based on this data. Data of object detection needs to be communicated serially to the control unit over the LAN connection and the control unit will use this data to make navigational decisions for the vessel.

4 Conclusion

4.1 Design Rational

Our system will contain dedicated hardware and software components. Our systems hardware will have its own computer and camera system. Using the same computer for navigation and our software system will save power, a constraining factor on the USV. Additionally, using the same computer system allows for easier intercommunication between the obstacle detection system and the navigation/end user communication system.

Our system will contain a camera system composed of a single digital camera. Having a single camera is suitable for object detection, and while it does not allow for stereo determination of depth, it is sufficient to meet the base requirements. Ideally, a camera array providing a 360 degree field of view would be used.

Our system's software components will consist of using Python and its vast modules. Python is the language of choice, because the existing software for the vessel's navigation uses Python. In addition, Python contains a plethora of modules for almost anything that allow for easy but powerful programming. OpenCV, a Python module, contains large functionality for obtaining and exporting camera input feed. It also contains all the needed object detection algorithms and methods. OpenCV will allow the system to process images and scale and transform them, and use a deep neural network. No other computer vision library allows for full control and ease of use object detection system. Furthermore, the systems communication will be done with serial communication. Python contains modules to easily serialize and send data. In addition, our system will be using a MATLAB script created by a previous classmate working with the ROSS. Our Python project will import this MATLAB script using a MATLAB python engine to easily use its functions with our project. Therefore, Python and its various modules will be used for our object detection and navigation.

4.2 Timeline

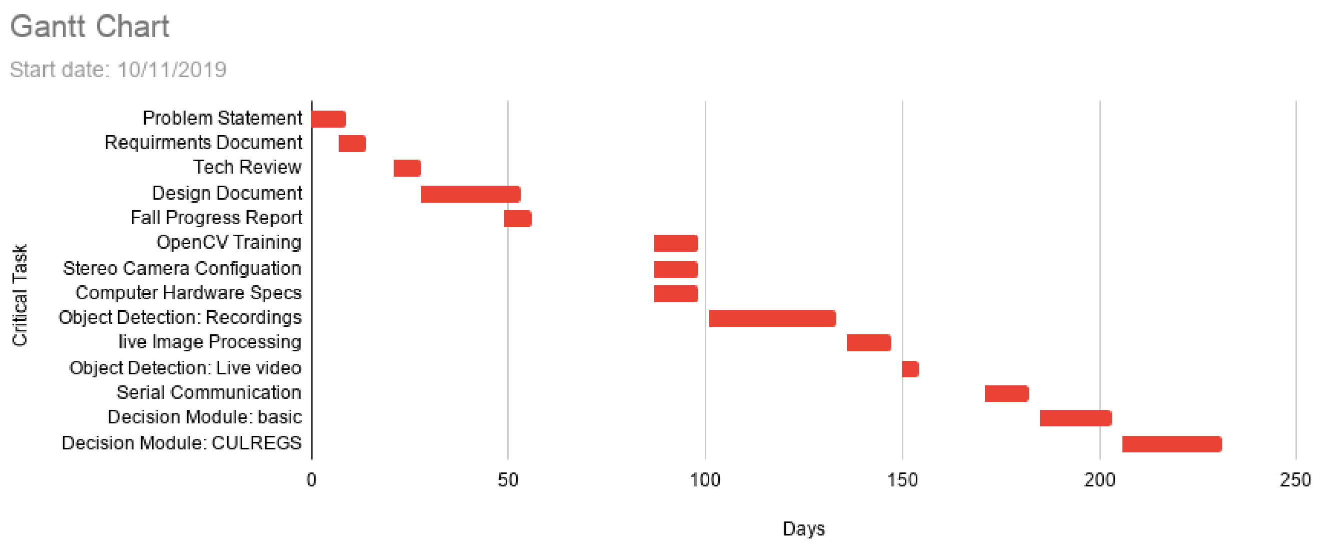


Fig. 1. Predicted Timeline

4.3 Summary

The Object detection System will be a module to detect the presence of an obstacle in front of a USV and offer course suggestions to the USV's navigation system to avoid collision. The system will incorporate new hardware to sense the environment in front of the via a digital camera system, and process video input on the existing computer system. Images will be processed using OpenCV's Oriented FAST (Features from Accelerated and Segments Test) and Rotated BRIEF (Binary Rough Independent Elementary Feature): Feature detection and description algorithm developed at OpenCV labs (ORB) deep neural network module, allowing for the incorporation of an external neural network. We will persistently store object data track objects over time. The object data we collect will be sent to client's system that will provide positional data of each object relative to the USV.

5 Glossary

Acronyms

CPU	Central Processing Unit.
GPS	Global Positioning System.
LAN	Local Area Network.
ORB	Oriented FAST (Features from Accelerated and Segments Test) and Rotated BRIEF (Binary Rough Independent Elementary Feature): Feature detection and description algorithm developed at OpenCV labs.
ROSS	Robotic Oceanographic Surface Sampler.
USV	Unmanned Surface Vessel.

References

[1] "Feature matching," OpenCV, Nov. 2019. [Online]. Available: https://docs.opencv.org/master/dc/dc3/tutorial_py_matcher.html

[2] "Docs – pyserial." [Online]. Available: https://pyserial.readthedocs.io/en/latest/pyserial.html

4 TECH REVIEWS

4.1 Tech Review - Michael Gabriel

Contents

1 Technology Review

1.1 Abstract

The following document is a review of the technologies that our team will use during the creation of an obstacle detection and avoidance system for the Robotic Ocean Surface Sampler (ROSS). Included is a section on Obstacle Avoidance Algorithms, Databases, and Programming languages.

1.2 Introduction

Our task is to develop a system that can detect obstacles and send information on how to avoid the obstacle to the autonomous driving system. The following technologies will be pivotal in our completion of the project and thus potential technologies must be thoroughly analyzed before one is chosen.

1.3 Obstacle Avoidance Algorithms

To autonomously avoid obstacles while traveling through the ocean is one of the main goals of this project. In order to achieve this goal we will either need to use an existing obstacle avoidance algorithm or create our own. Autonomous obstacle avoidance has been a huge topic amongst computer scientists and electrical engineers over the past years and there are many available algorithms to study and choose from. The three that will be discussed in this section are; vector field histogram (VFH), tangent bug algorithm(TBA), and artificial potential field method (PFM).

1.3.1 Tangent Bug Algorithm (TBA)

The Tangent Bug algorithm is one of the simpler obstacle avoidance algorithms used by autonomous vehicles. The biggest benefit of this algorithm is the low amount of computation required to maneuver around obstacles. For the algorithm to work the following information is needed: target location, current location, and the location of obstacles. This is all represented on a two dimensional plane. The algorithm starts by trying to reach the target in a straight line. If there are no objects in the way it will continue to move in a straight vector toward the target until it reaches the target or it becomes too close to an obstacle. If it is too close to an obstacle it will begin to move parallel with the parameter of the object until it can move in a straight line again toward the target.[1]

1.3.2 Potential Field Method (PFM)

The Potential Field Method algorithm is an algorithm that is inspired by magnetic fields. In this algorithm the autonomous vehicle and all obstacles will have the same charge and thus the obstacles will repel the vehicle and the target location will have the opposite charge and thus will attract the vehicle. This algorithm creates a digital representation of an attractive and repulsive field that allow the vehicle to move toward the target while avoiding obstacles. Although this algorithm is better at reaching the target in less time than the Tangent Bug algorithm the amount of computation needed is significantly more and the area of operation must be known before the vehicle can safely travel through it.[2]

1.3.3 Vector Field Histogram (VFH)

The Vector Field Histogram algorithm is an algorithm that takes constantly updated data to model the world in on a two dimensional Cartesian grid. For this algorithm to work the location of the autonomous vehicle must be known and on board sensors must relay a constant stream of data about the environment. The algorithm builds a two dimensional array of numbers that are weighted based off of the certainty of that location containing an object. The algorithm then adjusts its directional vector to put these objects outside of the safety threshold radius of the vehicle. After the algorithm adjusts to avoid collision it does a second adjustment to ensure that it is still moving toward the target.[3]

1.3.4 Recommendation

It is my recommendation that the team uses either the tangent bug algorithm or the vector field histogram algorithm. I would suggest the Vector Field Histogram algorithm if we have access to a machine with a capable processor to handle the modeling and calculation the algorithm requires. The other limiting factor of using the VFH algorithm is the quantity and quality of the on board sensors. If for some reason we are unable to obtain the equipment necessary to support the VFH algorithm it is my suggestion that we use the Tangent Bug algorithm due to its low computational requirements we may have to engineer around some of its shortcomings.

1.4 Databases

Some of the technologies that our team will use may require the use of a database. There are two main types of databases that will be discussed in this section. First are the Structured Query Language (SQL). Second are the non Standard Query languages sometimes called Not Only Standard Query Language (No-SQL). These different classes of databases have performance trade offs that will be discussed below. The difference between local and cloud databases will also be explored.

1.4.1 Structured Query Language (SQL)

There are many existing Structured Query Language databases available for the team to use. Some of these are free like MariaDB while others like IBM's DB2 are specifically for commercial use and come at a significant cost. All of these SQL databases a structured very similarly and often use nearly the same programming syntax making it relatively easy to learn a new database system the developer already knows one of the other SQL databases. SQL databases require a schema to be created that represents a strict set of rules that the database must follow. This is great for data consistency but makes it difficult to update or change existing structures. The scaling of SQL database performance is heavily related to the power of the machine it is running on.

1.4.2 Non Structured Query Language (No-SQL)

Like SQL databases there are many free and commercial No-SQL databases to choose from. One of the biggest differences between SQL and No-SQL databases is how data is stored. In No-SQL databases data can be stored in many different ways and it is even possible to change the schema without Modifying preexisting data.

1.4.3 Local Database

Both SQL and No-SQL databases can be deployed on a local machine. In our case this would be on one of the computers on the boat. The benefit of doing this is that all of the data is on the same local network of the computer that will need to use the data. This allows for secure and quick queries to be used in our applications.

1.4.4 Cloud Database

The other option is we could deploy our database on a cloud server. This would allow us to offload some of the computational requirements to the database server and would free up some more CPU for our programs. The problem with this option is that it will require a stable internet connection to be effective.

1.4.5 Recommendation

I recommend that our team uses a free local SQL database like MariaDB. It is my belief that our database needs are small enough that we won't be limited by a SQL database. I also believe that our local PC will have enough computational power to support the database on the same computer that will run our obstacle detection and avoidance software.

1.5 Programming Languages

Our project is to design an automated collision detection and avoidance system to by used by an automated marine research vessel, thus the decision on what programming language is pivotal. In this section I will cover three computer languages: C, Python and Java. I briefly discuss the strengths of the languages and conclude with the suggested language.

1.5.1 C

The C programming language is a general purpose programming language that has been in use since the early 1970s when it was created to program the Unix operating system.[4] C is currently ranked number two on the Toibe Index, a Website that uses analytics to track the most popular programming languages.[5] C is a high level programming language that allows the programmer to manage their program memory and compile their program into machine code. C is often used in embedded systems because once compiled the C program does not take up very much space, unlike languages like Python and Java which require virtual machines to run their programs. C is also generally considered to be one of the fastest running computer languages and runs well on computers with limited resources.

1.5.2 Python

The Python programming language was created in the 1980's and is currently the number 3 programming language on the Toibe Index.[5][6] A large part of Pythons popularity is the large number of open source third party packages available to use. Python is a general purpose high level programming language that unlike C doesn't use a compiler and instead uses an interpreter for running a program. This means in general python code is slower than C code which can run natively on a machine and also requires the interpreter to be installed on an embedded system to run Python code. There is however an open source project called Cython that is a super set of Python and allows the programmer to compile Python code into compiled C code. This gives the programmer the ability to use all of the powerful libraries from python and still achieve the lightweight speed of C.

1.5.3 Java

The Java programming language is a high level general purpose programming language. Java requires the programmer to use the Object Oriented programming paradigm. Java is currently the number one language on the Toibe Index and has been for many years.[5] Java requires the Java Virtual Machine (JVM) to be installed on a machine in order for the programs to run. Java compiles Java code to Java bytecode which is then compiled by the JVM into machine code. This system allows Java to run on any operating system that the JVM exists on. Java is commonly used in enterprise software where the object oriented paradigm helps keep large complex programs manageable.

1.5.4 Recommendation

I recommend that we use Python for this project. The sheer number of libraries available for the team to use will greatly help the team in the production of the object detection and avoidance software. The teams software will run on a

GNU/linux machine and should have plenty of space to run a Python interpreter, but If we need to we could also use Cython and compile the project code to compiled C machine code.

4.2 Tech Review - D. Max Harkins

Contents

1 Introduction

Our team's objective is to implement a system to allow an autonomous ocean research vessel to detect and classify obstacles in it's path, and change trajectory to avoid them. For this project, my role will be to have in-depth knowledge on obstacle detection technologies and tools. Our system will detect obstacles through analyzing data from a video feed. Therefore, we will select a computer vision library to use to simplify our video processing and analysis. Additionally, we will select algorithms to detect features such as corners and edges in images, and to match features between video frames. These technologies will form the core of the obstacle avoidance subsystem, and allow for detection and tracking of objects over time, thus providing data that can be used for course correction decisions.

2 Computer Vision Libraries

Since object detection requires real-time video processing, it is a computer vision problem. As such, it is practical that we use pre-existing computer vision library. Most computer vision libraries provide functions which allow for easy processing of images and videos, and manipulation of images as arrays. Additionally, many libraries implement feature detecting algorithms and some libraries include tools to facilitate the use of machine learning. Different libraries correspond to different computer languages, so deciding which computer vision library to use will go along with deciding what language to do development in. In the following subsections, three computer vision libraries are examined.

2.1 Dlib Library

The Dlib Library is a C++ library offering a wide range of tools, among them a set of functions for computer vision. While Dlib is intended for C++, it has a Python API which provides Python use of many of it's functions (especially image processing functions) [1]. Dlib provides resources for Baysian networks, data compression, data structures, graph creation and manipulation, image processing, linear algebra, threading, machine learning, metaprograming, networking, min-max optimization, and parsing [2]. Ultimately, if Dlib is selected to be used, it would be for it's image processing components rather for it's mathematical functions and tools. One advantage of Dlib is that it has extensive documentation, though it appears to have lower use than OpenCV and less user-made tutorials.

2.2 VLFeat Library

The VLFeat Library is an open source library which provides tools for feature recognition and matching. VLFeat is written in C and is intended for use with C code, but also provides an interface in MATLAB for ease of use [3]. Unlike Dlib, VLFeat does not provide extensive tools for other image processing and miscellaneous operations. There are no built in functions for image processing, compression, or filtering.

2.3 OpenCV Library

OpenCV is another open source library for realtime computer vision. It is estimated to have exceeded 18 million downloads, and has a community of more than 47,000 users [4]. There is extensive documentation, support forums, and tutorials. Interfaces for GPU-compute tools OpenCL and CUDA are currently in development. OpenCV has interfaces for C++, Python, Java, and Matlab. OpenCV contains tools to decode and encode images and video, do basic image processing and filtering, do 3d analysis of video, detect and match features, detect pre-defined objects using machine learning, and do gpu accelerated computing [5]. Essentially, OpenCV provides all the tools needed for real-time object detection, from processing the video to doing the detection. In Python, OpenCV works with Numpy, a powerful linear algebra library [6].

2.4 Criteria for selection

While it is not a strict constraint, it is preferred that we use Python for all parts of our system in order to be consistent with previously written code for the autonomous vessel. The largest criteria is that obstacle detection happens in real time. We must use a tool that will allow us to process video and images quickly, without excessive overhead. To increase ease of development, it is desirable that the library we choose have sufficient capabilities to perform all computer vision tasks needed for obstacle avoidance, in order to avoid having to mix and match libraries thereby increasing complexity.

2.5 Recommendation

For our project, using OpenCV with the Python programming language is the best choice. OpenCV provides a large number of built-in tools for using computer vision, without including extra resources that are unnecessary for the project. There is copious documentation and resources to learn OpenCV online. Since Python is a syntactically simple language, the pairing of OpenCV with Python will make for a robust, quick, and easy to develop obstacle detection system.

3 Feature Detection Algorithms

The first step in detecting an object's position and velocity relative to the boat is identifying where in a captured image an obstacle is present. Feature detection is a common method to detect objects and match them between frames of video. Feature detection algorithms find key points, or locations in an image where an edge or anomaly is present. Since key points generally do not change based on image rotation or scaling, the key points found in multiple frames of a video can be compared and matched. From clusters of key points, an object can be constructed and dealt with. A variety of key point detectors exist, in this section the SURF, SIFT, and ORB detectors will be examined and compared.

3.1 SIFT

SIFT is an acronym for Scale-Invariant Feature Transform. This key point finding algorithm was created in 2004 by researchers at University of British Columbia [7]. SIFT uses numerical techniques to find among other values the coordinates of key points, an angle associated with a key point, and the meaningful neighborhood around the key point. These characteristics are useful in matching key points in different frames of an image, or key points between two cameras mounted in parallel. SIFT is rotation and scale invariant, meaning that the detected key points will correspond to the same parts of an image regardless of how an image is scaled or rotated. SIFT is a patented algorithm, and not available for commercial use without special licence.

3.2 SURF

The SURF algorithm uses similar techniques to SIFT, and is in some ways a successor of SIFT. The SURF acronym stands for Speeded Up Robust Features. The largest advantage of SURF over SIFT is that SURF is faster. SURF, like SIFT, is rotation and scale invariant. Additionally, SURF and SIFT return the same output values (coordinates of key point, angle, neighborhood) [8]. However, a special variant of SURF exists that is not rotationally invariant, and does not output an angle associated with key points. This variant is called U-SURF [openCV SURF], and while faster than SURF, is not suitable for our project because some level of rotational invariancy will be necessary to account for motion of the autonomous vessel. Like SIFT, SURF is patented.

3.3 ORB

Unlike SURF and SIFT, ORB is not patented, and free to use. ORB stands for Orient FAST and Rotated BRIEF, and uses the FAST algorithm to detect corners in conjunction with the BRIEF algorithm to describe features [9]. ORB is faster than both SIFT and SURF, but is still rotation and scale invariant. While experiments show that ORB detects fewer key points per image as SURF or SIFT, the difference is not significant [10].

3.4 Criteria for selection

Since our application must run in real time, likely on a weak computer, speed is the top criteria for selecting a key point detector. Additionally, we would like to make sure our key point detector is scale and feature invariant, as the three examined are.

3.5 Recommendation

Since ORB is the fastest of the three key point detectors, and is not patented, it is the best choice to use in our obstacle detection system. Also, while SIFT and SURF can be integrated into OpenCV with some work, ORB is integrated into OpenCV natively.

4 KEYPOINT MATCHING ALGORITHMS

Detecting key points is only part of gathering information about objects. After key points are found, they must be matched between video frames to track how objects move relative to the boat. Multiple algorithms exist to match key points between images. In OpenCV, two key point matchers are built in to the library. They are FLANN and BFMatcher.

4.1 BFMatcher Algorithm

BFMatcher is a bruit force feature matching algorithm. It matches each key point from one image to every other key point in the other image. The OpenCV implementation of this algorithm takes as parameters the norm to use for distance evaluation, and a boolean to indicate if cross checking should be performed [11]. If cross checking is performed, then for a match to be returned not only must the bruit force matcher match a point from the first image to the second, but matching the found point in the second image back with the first image must result in the original point being found [12]. Since the matcher checks each key point in a second image given key points in a first image, the bruit force matcher runs in $O(n^2)$ time, and is relatively slow. However, due to the thoroughness with which BFMatcher operates, it will return the best possible matches between key points.

4.2 FLANN Matching Algorithms

The FLANN acronym stands for Fast Library for Approximate Nearest Neighbors. Unlike BFMatcher, which matches a single key point in one image with all key points in another, FLANN matches key points by approximating which key points in a secondary image a key point in a primary image could correspond to, and then matching a point in the primary image with only a subset of key points in the secondary image. Since not all key points in a secondary image are examined for each key point to be matched in a primary image, FLANN is faster than BFMatcher [13]. While BFMatcher is a single algorithm that can be used with different norms and modes, FLANN is composed of a library of algorithms, accessible through a single function using OpenCV [12]. Because it is a library of algorithms, FLANN is more complex to use. Additionally, since it does not examine every pairing of points between images, FLANN may not match points with as great of accuracy as BFMatcher.

4.3 Criteria for selection

As mentioned throughout this document, our system must operate in real time. Therefore, the priority for the tools we use is speed. A secondary general criteria is ease of use, though the abilities of a tool will be weighed above how easy it is to operate. Specific to key point matching, it is desirable that our chosen technology meet the criteria that sufficient key points can be matched between frames to allow for persistent tracking of an object over time. The percentage of key points that need to be matched correctly for persistent object tracking is unknown for our application, and will have to be determined experimentally. It is likely that with proper configuration, either matching tool discussed above will be able to find sufficient matches.

4.4 Recommendation

Because producing a fast program that can run in a low power environment is critical to our project, it is recommended that the FLANN matcher be used. Although bruit force matching is known to provide more accurate results, it is likely that with proper configuration the FLANN matcher can achieve sufficient key point matching while obtaining higher speeds than the BFMatcher algorithm.

References

[1] "Dlib python api," Sep. 2019. [Online]. Available: http://dlib.net/python/index.html

[2] "Dlib c++ library," Sep. 2019. [Online]. Available: http://dlib.net/

[3] "vlfeat," Github, Jan. 2018. [Online]. Available: https://github.com/vlfeat/vlfeat

[4] "About," OpenCV. [Online]. Available: https://opencv.org/about/

[5] "Opencv 2.4.13.7 documentation - introduction," OpenCV, Nov. 2019. [Online]. Available: https://docs.opencv.org/2.4/modules/core/doc/intro.html

[6] "Introduction to opencv-python tutorials," OpenCV, Nov.

[7] "Introduction to sift (scale-invariant feature transform)," OpenCV. [Online]. Available: https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_feature2d/py_sift_intro/py_sift_intro.html#sift-intro

[8] "Introduction to surf (speeded-up robust features)," OpenCV. [Online]. Available: https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_feature2d/py_surf_intro/py_surf_intro.html

[9] "Orb (oriented fast and rotated brief)," OpenCV. [Online]. Available: https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_feature2d/py_orb/py_orb.html

[10] S. A. Perera, "A comparison of sift , surf and orb," Aug. 2018. [Online]. Available: https://medium.com/@shehan.a.perera/a-comparison-of-sift-surf-and-orb-333d64bcaaea

[11] "cv::bfmatcher class reference," OpenCV, Nov. 2019. [Online]. Available: https://docs.opencv.org/3.4/d3/da1/classcv_1_1BFMatcher.html

[12] "Feature matching," OpenCV, Nov. 2019. [Online]. Available: https://docs.opencv.org/master/dc/dc3/tutorial_py_matcher.html

[13] M. Muja and D. Lowe, "Flann - fast library for approximate nearest neighbors," Jan. 2013. [Online]. Available: https://www.cs.ubc.ca/research/flann/

4.3 Tech Review - Chris Patenaude

Contents

1 Introduction

The OAS will function as a peripheral module for an Autonomous Ocean Research Vessel. This document shall look at three of the technical components needed to successfully execute the system's primary directive, to avoid potential hazards on the surface of the ocean. Specifically, the document shall examine the crucial hardware components needed to gather visual input data from the physical environment, process visual data into logical directives, and transmit logical directives via a physical interface to a controller module. The technical component described here form the basis of my roll in the project which will consist of maintaining the hardware components and facilitating communication between input devices and output directives.

2 Environmental Sensor

2.1 Purpose of the Environmental Sensor

In order to avoid obstacles in a dynamic environment, such as an ocean coastline, a USV must be capable of gathering data of its surroundings via a sensor. Several technologies exist for collecting data on physical features surrounding a device; however, many such technologies are prohibitively expensive or do not meet the exact requirements for the project. This section shall explore four sensor options that are commonly used to implement obstacle avoidance modules on Unmanned Vehicles and determine their usefulness in the specific oceanic environment which our USV shall operate.

2.2 Existing Technology

Digital camera sensors, offer an inexpensive and versatile method for collecting data. There are two conventional method for image based obstacle avoidance: Monocular Vision, and Stereo Vision. Monocular Vision used a single camera to determine relative distance from the USV to objects in the environment via textural and visual ques [1]. Stereo Vision utilizes two cameras synchronized with each other to build a 3D map the the environment in the shared regional arcs of the cameras [2]. A third option that has been recently explored by [3], discusses the use of omni-directional cameras; however, our project is unconcerned with object detection in a 360 degree arc but rather objects that pose a threat of collision in front of the USV.

Digital cameras are the most economical choice of the sensors discussed in this section and offer a number of advantages. Successful camera implementation of obstacle avoidance have used webcams with resolutions as low as 80x44 pixels, this implies that inexpensive cameras are adequate for real-time object avoidance [1]. Digital cameras do not require specialized equipment such as peripheral sensors (laser range finders, sonic sensors, etc.) for verification. The field of computer vision is extensively researched, so there are also a large number of resources available to build upon. Open source frameworks such as OpenCV provide pre-built libraries for most of the functionality we require.

Visual based obstacle avoidance can be computationally expensive, even requiring GPU hardware support depending on the algorithm use to process the images [1][4]. Fortunately, many of the expensive algorithms are used for classifying object, which is unnecessary for our use case.

2.2.1 Monocular Camera Sensor

The Monocular sensor implementation uses a real-time video feed where images of 30 - 70 fps are segmented into three regions: ocean, skyline, and middle regions [4]. The middle region will contain obstacles such as coastline, other marine vessels, or icebergs and will be the region of interest for this method [4]. The primary benefit of using a monocular camera is lower equipment cost and monocular algorithms are no less effective then stereo versions.

Since monocular cameras generally analyses the texture composition of images to differentiate waterline, coastline, and skylines they can be unreliable in low light conditions [4].

2.2.2 Stereo Camera Sensor

The Stereo sensor implementation uses overlapping images from two synchronized cameras. The camera array can predict depth by calculating the disparity between a point observed from both cameras, this is advantages as simple algebraic triangulation can be used to build a 3D map of the arc shared by the cameras (This is known as the Parallax method) [2]. Since the stereo system uses simple math to generate 3d points, a stereo implementation can be implemented on hardware with an FPGA allowing the system to outperform CPU implementations.

Stereo vision algorithms have a tendency to smooth out data maps which can result in loss of relevant environmental data [2]. In a dynamic environment such as the ocean or coastline, vital information could easily be averaged out of an image due to water motion and cluttered coastlines.

2.3 Conclusion

Both Monocular and Stereo sensors are promising candidates for our project; however, the monocular implementations reduces the amount of hardware needed to accomplish the same vital task, and with the proper algorithm, does not coast much more computational power then the stereo implementation; further, the monocular implementation does not require specialized hardware such as an programmable FPGA to maintain fast image processing. So, we have chosen a monocular approach for the OAS.

3 Computing System

3.1 Purpose of the Computing System

The Computing System (CS) is the hardware system that runs the obstacle avoidance system. The CS must have adequate processor clock speed, memory, and power consumption to be able to consume data from the environmental sensor, process data via the selected computer vision algorithm, and respond to data in real-time to prevent object collision.

3.2 Existing Technology

This shall explore whether to use a dedicated computing system for the object avoidance system or implement object avoidance on the current computing system already in place on the USV. Currently, the USV is controlled by an Intel NUC computer running Linux OS. It is vital that the object avoidance system does not interfere with the GPS navigation and radio communication operations handled by the NUC system, so resource availability is limited to what is not being used by the above systems.

3.2.1 Shared Computing System

Implementing the OAS on the same computer as the control system is financially advantageous and reduce the overall systems complexity. Resources are saved since there is no need to acquire or build another computer. Since all the software operates on the same processor, it is trivially easy to synchronize communication between the OVA module and the control software. No hardware interfacing is required to communicate between the processes.

Implementing the OVA on current control computer means both processes must share same resources, potentially causing resource collisions. The CS uses about 15% (300 MHz) of the CPU on average [5], but that usage can spike during multiple operations like processing communication, calculating navigation vectors and controlling various scientific instrumentation. Depending on the algorithm selected, the OAS can use nearly 100% of the host system CPU [4]. Since it is crucial that the control system has priority over hardware resources, this limits the possible algorithm choices we can use on the hardware.

3.2.2 Dedicated Computing system

Implementing the OAS on a dedicated computer offers a versatile platform for experimentation and full access to hardware resources. The flexibility of a dedicated computer make it possible for our team to experiment with different algorithms without having to worry about resource availability. Entire computer will be devoted to data input, processing, and output; further, a dedicated system is not predefined, allowing us to customize the system build to suite our needs.

A dedicated computer for obstacle avoidance does add additional cost and complexity to the overall system. The dedicated computer must be able to communicate with the control process running on the current NUC system in real-time, providing the NUC with course suggestion to avoid any detected objects. A serial communication interface and protocol must be implemented on the two computing units adding an additional layer of communication complexity.

3.3 Conclusion

Using a dedicated computer for the OAS does increase cost, but the added flexibility of a system completely under our control is a benefit that must be taken under consideration. Since the control system must have priority over resources like the CPU, a dedicated computer is even greater of an advantage as the OAS will not have to compete with the CS for system resources. So, we have chosen to use a separate, dedicated computer to host the OAS.

4 Serial Hardware Interface

4.1 Purpose of the Serial Hardware Interface

Using a dedicated computing unit will require a hardware interface between the USV's CS and OAS. Other means of communication between computing units are impractical for our proposes and will not be discussed further.

4.2 Existing Technology

The current NUC computer that controls the USV has a limited interface for hardware ports. Ideally, the serial communication interface will be implemented with the NUC's limitations in mind. The ports available are as follows: USB 3.0, LAN, and HDMI. Serial communication is generally done using an RS-232 Serial Port but with the use of adapters, other interfaces can be used as well.

4.2.1 RS-232 Serial Port

The RS-232 Serial Port is the simplest method for interfacing our two computers. The Linux OS running on both of the computers supports built in networking protocols for data serial port communication, making RS-232 a great choice for simplicity sake; however, as stated previously, the NUC computer does not have a serial port.

4.2.2 USB Port

USB is a well known interface and supports build in protocols on many operating systems for Host-to-Device communication. Unfortunately, both computers for this project are hosts so the protocol does not work for out purposes.

4.2.3 LAN Port

The LAN Serial protocol was built for local host to host communication and is supported on linux OS. This makes it the ideal candidate for our use case.

4.3 Conclusion

Due to the lack of a RS-232 Serial port on the NUC unit, and inability of USB to provide Host-to-Host communication, we have chosen to use a LAN interface and protocol for out serial communication interface.

5 SUMMERY

The technical components discussed in this document comprise the three main hardware choices to be considered for the OAS system: an environmental sensor to collect data, a host machine to process data, and a serial interface to transmit data to a control unit aboard a USV. Through careful examination of the available options, we determined initial decisions regarding the technologies discussed above. Our initial choice for an environmental sensor is a monocular camera due to its economical cost and lack of any major disadvantages. We chose a dedicated host machine which will allow us to experiment with a verity of computer vision algorithms as well as prevent our module from interfering with the control module currently integrated with the USV. Finally, we chose the only viable option for hardware interfaces, the LAN interface, that is both conventionally used for host to host communication and is available on the current control computer.

References

[1] C. Viet, I. Marshall, ”VISION-BASED OBSTACLE AVOIDANCE FOR A SMALL, LOW-COST ROBOT” Computer Science Department, University of Kent, Canterbury, United Kingdom

[2] Y. Yu, W. Tingting, C. Long, and Z. Weiwei, “Stereo vision based obstacle avoidance strategy for quadcopter UAV,” 2018 Chinese Control And Decision Conference (CCDC), 2018.

[3] G. Puvics, M. Szabó-Resch, Z Vámossy, ”Safe robot navigation using an omnidirectional camera,” 2015 16th IEEE International Symposium on Computational Intelligence and Informatics (CINTI)

[4] A. Kadambi, A. Bhandari, and R. Raskar, “3D Depth Cameras in Vision: Benefits and Limitations of the Hardware,” Computer Vision and Machine Learning with RGB-D Sensors Advances in Computer Vision and Pattern Recognition, pp. 3–26, 2014.

[5] *OSU Ocean Robots* [Online]. Available: http://makani.coas.oregonstate.edu/ross/Overview.html. [Accessed: 12-Oct-2019].

4.4 Tech Review - Greg Sanchez

1 Introduction

Researching ocean phenomena, including glacial melting rates, gives scientists data about current global climate conditions. Collecting data near glaciers manually risks human lives due to extreme conditions. Using an Autonomous Research Vessel (ARV) can increase efficacy while reducing risk. Past research vessels consisted of kayaks equipped with an outboard motor and basic control unit, but they did not have any support for an automatic obstacle avoidance system.

Our team will implement an obstacle avoidance system for the new ARV. This system will be implemented with an array of cameras mounted above the boat, which will supply a video input to an on-board computer. The on-board computer will then use computer vision to identify objects around the boat and calculate their motion. The boat will then be able to adjust its course or stop based on the type of object identified and whether the object is determined to be moving towards or away from the boat. Implementing a computer vision system for collision avoidance could be open sourced for the benefit of others if completed satisfactorily.

I have been tasked with handling the initial processing of data. This includes how to receive and manipulate video feed, and export this information for computer vision algorithms. In addition, a choice has to be made on whether to have the CPU or GPU do the image processing. Lastly, I will look into the potential of using machine learning with the object detection.

2 Deriving and Processing Data

The first and most important scope of this project relies on processing and deriving camera input. This input from the cameras will be used for the autonomous object detection.

Multiple forms of software exist that can obtain and export camera feed. For instance, programming languages Bash or C++, or external programs can be used. The project's programming language of choice is Python, so that will be used to get and export the camera feed.

Python has a large source of modules, and there are many that can do this exact process. A few notable modules to choose from are OpenCV, SimpleCV, and PyGame.

Going backwards, PyGame is a ″free and open source python programming language library for making multimedia applications like games built on top of the excellent Simple DirectMedia Layer (SDL) library. ″ [1] PyGame provides easy functionality for capturing images from a camera source. In addition to this it is able to alter the image's colorspace and threshold. Also, it is able to export images to files which can be picked up by the image processor, but there is no other use with the module for the project scope. As the description of it says this module is catered towards multimedia applications. Therefore, this module can be used for our project, but it adds another module to include with limited support for image modifications.

SimpleCV, is an ″an open source framework for building computer vision applications.″ [2] This module is a simplified computer vision library that takes advantage of OpenCV. SimpleCV allows users to easily capture camera input and use image processing technologies to capture object detection. Capturing camera data can be either exported to a file or sent to the image processing technologies, and the data can be manipulated in many ways. It encapsulates all the tools in one with a simplified library. This module is a great choice for easily obtaining and exporting camera input, but the over simplified library does not support a lot of functionality needed in the scope of the project.

OpenCV, is an ″open source computer vision and machine learning software library.″ [3] This module contains most of the necessary elements for the scope of the project. It also contains a surplus of functionality of obtaining and processing camera data. The functionality is not simplified, so being able to manipulate bit depths, file formats, color spaces, buffer management, and etc. for the camera data is possible. More functionality includes capturing the camera data at any interval (e.g. 1 frame per second) and resolution up to the maximum supported by the camera. This output can then also, like SimpleCV, be exported to a file or sent to the image processing technologies built into OpenCV.

In conclusion, OpenCV is the best choice. This module will be used by other aspects of the project. It contains over 2500 optimized algorithms that ”can be used to detect and recognize faces, identify objects, classify human actions in videos, track camera movements, track moving objects” [3], capture camera input from multiple sources, modify and export camera input in almost any way, etc.

3 Image Processing with the GPU vs CPU

Processing images with OpenCV can be done entirely with the CPU or GPU. Choosing which one to use (and if the GPU is necessary) is critical to ensuring the object detection is done as efficient and precise as possible. Though using the best market CPU and GPU can be used there are limitation to the size, risk of water/movement damage, and power supplied by the vessel’s battery that need to be taken into account. Only Nvidia GPU’s will be used in comparisons, because of its Computer Unified Device Architecture (CUDA) power and functionality for image processing. In addition, only Intel CPU’s will be used in comparisons, because of their integration with Intel NUC’s (choice of computer).

James Bowley provided an excellent performance comparison for OpenCV with Intel CPU’s and Nvidia GPU’s CUDA. [4] James compared four different computer configurations with different levels of performance. The configurations tested multiple OpenCV functions and its runtimes. These are the four configurations that tests three different micro-architectures denoted in the parenthesis [4]:

1) CPU: i5-4120U (Haswell), GPU: GTX 730m (Kepler)
2) CPU: i5-5200U (Broadwell), GPU: GTX 840m (Maxwell)
3) CPU: i7-6700HQ (Skylake), GPU: GTX 980m (Maxwell)
4) CPU: i5-6500 (Skylake), GPU: GTX 1060 (Pascal)

These CPU’s, in terms of processing power, are more powerful than what will be used for the project. The machine this will be running on will be a low power i3 processor, but this will be a good explanation of how a GPU will outperform a CPU in terms of running OpenCV.

Some of the results are shown below. The below figure ”shows the speedup averaged over all 5300 tests (All Conifgs)” [4]

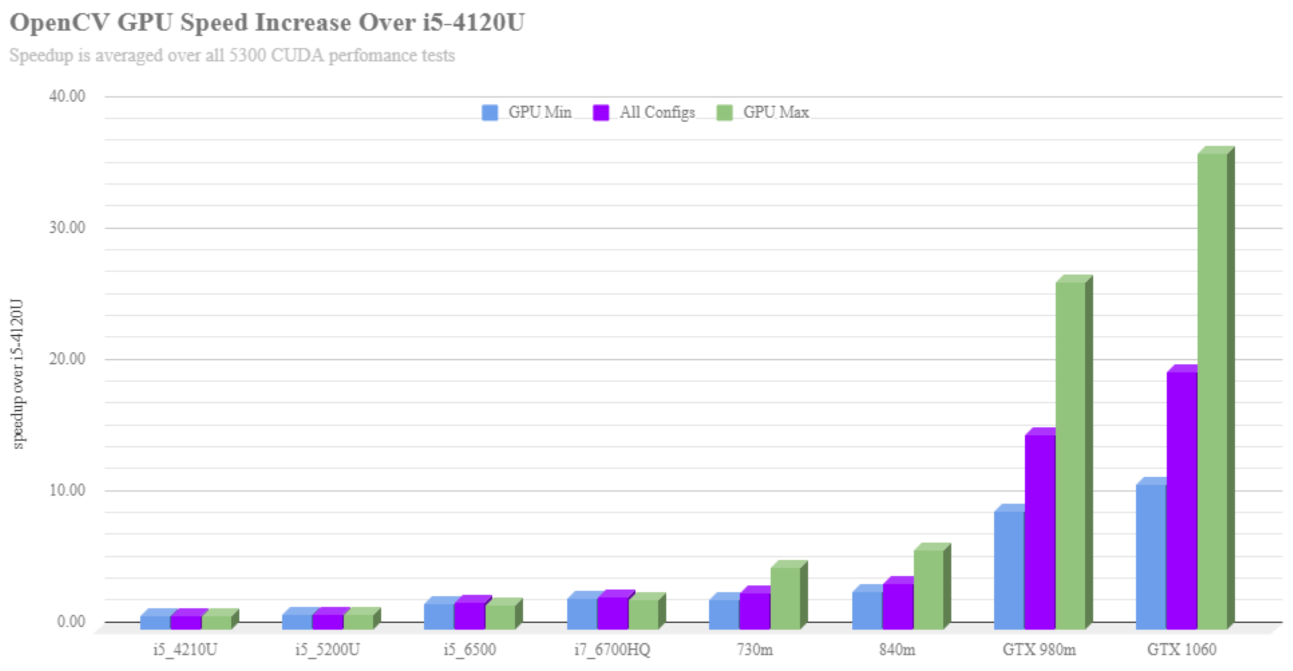


Fig. 1: OpenCV GPU Average speedup Over 5300 Tests

As James describes, this figure demonstrates that the slowest GPU configurations are similar in performance to the fastest CPU configurations. This performance similarity becomes a large difference when looking at individual OpenCV functions. The below figure "shows the top 20 functions where the GPU speedup, was largest." [4] Note the large speed up difference (y-axis) in the GTX 980m and 1060.

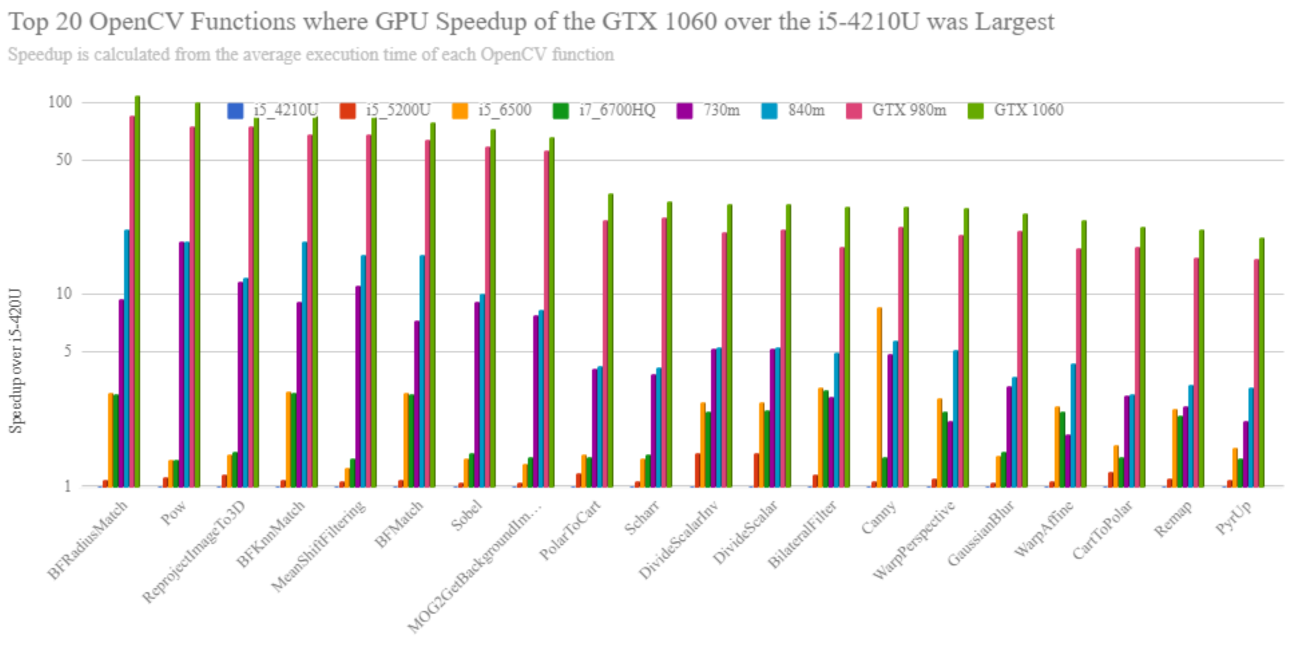


Fig. 2: Top 20 OpenCV Largest Speedups

Though some of these functions will not be used in the project scope others like BFKnnMatch, BFMatch, GaussianBlur, and WarpPerspective are image processing functions that will most likely be used. Those image processing functions can be used to infer the speedup on the functions used in the project using a GPU.

The limitations on the vessel also limit the potential for a higher tier GPU. The vessel uses a standard car battery to supply the power to all the devices. An average car batter can supply 500-700 watt hours of power. With the other devices taking some of this power an assumption of having 100 watts of usable power can be used. The GTX 1060 requires ~400 watts of power while the GT 730m is rated at 33 watts of power. Furthermore, its size and risk for water/movement damage will be solved by enclosing the computer in a safe box. The computer runs does not run at high temperatures, so the risk of overheating is negated. Also, the GPU does not have any physical moving parts so sporadic movements will not affect the GPU. Therefore, using any low to mid range GPU will improve runtime by a factor of 10 or above while being able to function on the vessel.

4 MACHINE LEARNING

Object detection is used to locate objects in an image. Machine learning is not required for this, but it can be used to more accurately depict its class of an object and/or size.

There are two main options for using machine learning with object detection. Using Region-Based Convolutional Neural Networks (R-CNNs) or the You Only Look Once (YOLO) technique.

R-CNNs solve the problem for object localization, detection, and segmentation. Object localization locates ”the presence of objects in an image and indicate their location with a bounding box.” [5] Object segmentation recognizes objects and highlights the sepcific pixels of ”the specific pixels of the object instead of a coarse bounding box.” [5] There are three techniques for R-CNNs designed for object localization and detection: R-CNN, Fast R-CNN, and Faster-RCNN.

R-CNN is the one of the first successful CNNs that solved the problem. The R-CNN is composed of three modules: region proposal, feature extractor, and classifier. These modules are used to propose candidate regions or bounding boxes of potential objects in the image called 'selective search.'” [5] It is able to detect and classify object from an image with good accuracy. R-CNN is the simplest approach, but is the slowest.

Fast R-CNN solves the limitations of R-CNN: training is a multi-stage pipeline, training is expensive in space and time, and object detection is slow. [5] It uses a single model instead of a pipeline for its training and object classifications. The input is passed through a deep CNN, and after it's interpreted and split into two outputs: the prediction and bounding box. This model is faster in training and predictions, but still falls short by requiring a set of candidate regions with each input image. [5]

Faster R-CNN improved both training speed and detection from Fast R-CNN. It does this by combining two different technologies (Region Proposal Network and Fast R-CNN) to ”reduce the number of region proposals and accelerate the test-time operation of the model to near real-time with then state-of-the-art performance.” [5] This is the fastest of R-CNN that accurately depicts objects at fast speeds.

The YOLO technique is the next step in achieving object detection faster. This technique is much faster than R-CNN, but is less accurate than R-CNN. The speed difference allows objects to be detected in real time. It uses a ”single neural network trained end to end that takes a photograph as input and predicts bounding boxes and class labels for each bounding box directly.” [5]

Using machine learning for object detection is an excellent option for location and classifying an object. Being able to classify and object can allow a priority to be set on certain objects. For example, a kayaker will be given a higher priority to avoid instead of a small block of ice. The neural network can be trained to avoid detecting reflections on the water or birds in the sky. For the scope of the project, it does not matter what the object is only where it is and how it's moving in relation to the vessel. Therefore, machine learning will not be used, but is an option if the non-machine learning approach isn't viable. In that case the YOLO technique will be used for its real time object detection.

References

[1] PyGame. Pygame - about - wiki. [Online]. Available: https://www.pygame.org/wiki/about

[2] SimpleCV. Simplecv - computer vision platform using python. [Online]. Available: http://simplecv.org

[3] OpenCV. Opencv - about. [Online]. Available: https://opencv.org/about/

[4] J. Bowley. Opencv 3.4 gpu cuda performance comparison (nvidia vs intel). [Online]. Available: https://jamesbowley.co.uk/opencv-3-4-gpu-cuda-performance-comparison-nvidia-vs-intel/

[5] J. Brownlee. A gentle introduction to object recognition with deep learning. [Online]. Available: https://machinelearningmastery.com/object-recognition-with-deep-learning/

4.5 Tech Review - Tobias Hodges

CONTENTS

1 Project Overview and Goals

1.1 Problem Description

Researching ocean phenomena, including glacial melting rates, gives scientists data about current global climate conditions. However, performing research near glaciers presents problems for human researchers. While it is desirable to collect data at the interface between glaciers and the ocean, this is a highly dangerous area due to unpredictable glacier movement. Glacial ice is prone to fall off into the ocean in these locations, causing destruction to whatever may be floating below it. Collecting data near glaciers manually risks human lives due to extreme conditions. Using an Autonomous Research Vessel (ARV) can increase efficacy while reducing risk. Collecting oceanographic data can be time consuming for human researchers. In some cases, it is desirable to collect data from a location at regular intervals for extended periods of time. Researchers committed to gathering data are not being utilized to their full potential. The clients for our capstone group have constructed autonomous vessels for research in the past. Past research vessels consisted of kayaks equipped with an outboard motor and basic control unit, but they did not have any support for an automatic obstacle avoidance system. These vessels were able to be equipped with sensors and instruments, but were small enough that waves and currents on the ocean surface could cause the boats to fail.

1.2 Project Goals

Multiple metrics exist to measure our success in creating an obstacle avoidance system. First, we can measure the success of our computer vision program to detect obstacles from a camera input. This can be done by feeding prerecorded video to the system, and monitoring how successful our algorithms are at identifying obstacles. For this metric, we want to maximize the number of actual obstacles detected in the video while minimizing the number of false positives from the system. Our success can then be measured through real-world testing, since the autonomous boat can be deployed for testing near Newport, Oregon. If we can achieve a high rate of obstacle detection (¿ 90%), we will have achieved success for this system. It is important that our obstacle avoidance system functions in a range of environmental conditions, so we would like to achieve a high rate of obstacle detection in suboptimal conditions (low light, high swells, fog) in addition to achieving a high rate of avoidance in ideal conditions (bright light, flat water, low glare). If we are successful at identifying obstacles with computer vision, we will want to test how well our system corrects the course of the boat to avoid the detected obstacles. Obstacle avoidance tests can be simulated by inputting a prerecorded video feed containing obstacles to the computer and monitoring the output from the computer vision program to the navigation control program in order to verify proper action is taken. Only after we verify the success of our obstacle avoidance system through software simulations will we deploy it on the vessel. This document will outline some of the available technologies, our current direction, and alternative solutions which could also reach our end goal of solving these problems.

2 Group and Personal Responsibilities

To better understand and analyze the technology possibilities and alternatives in this project, our team has broken the technology requirements into six different categories, one for each team member:

- Processing (Greg)
- Decision Making (Michael)
- Object Identification (Max)

- Hardware (Chris)
- Communication/Navigation (Megan)
- Other Methodologies (Tobias)

This document, prepared by Tobias Hodges, focuses on Other Methodologies. This is a very broad category, and was meant to be so. There are many different technologies available for our team to implement and for an autonomous vessel to utilize. We are attempting to solve the problem described above using computer vision, but this document will focus on the question: ”is computer vision really the correct method?” This document will limn the current and assumed solution, describing some of the ideas behind it and our reasoning for choosing, then will highlight three major potential alternatives. These alternatives will attempt to remain solving the given problem (described above), but will do so using technologies separate from computer vision entirely.

3 Current Solution

Use of an ARV equipped with sensors solves the problems associated with employing people to physically collect oceanographic data. Autonomous vessels are currently able to provide access to areas too dangerous for humans to explore further, autonomous vessels could be made to perform a repetitive task for an extended period of time without constant human oversight. In order to reduce failure due to wave action, and have the capability to carry more scientific equipment, our client has constructed an autonomous vessel more robust than the previous autonomous kayaks. The new vessel consists of a Kodiak inflatable boat equipped with a powerful 20 HP motor. Such a vessel would be able to travel faster in the water, carry more equipment, and be more resistant to failure caused by waves and weather. To reduce the potential of colliding with an object, our team will implement an obstacle avoidance system for the new ARV. This system will be implemented with an array of cameras mounted above the boat, which will supply a video input to an on-board computer. The on-board computer will then use computer vision to identify objects around the boat and calculate their motion. The boat will then be able to adjust its course or stop based on the type of object identified and whether the object is determined to be moving towards or away from the boat. Implementing a computer vision system for collision avoidance could be open sourced for the benefit of others if completed satisfactorily.

4 Alternative Technology Options

4.1 Sonar

4.1.1 Overview

Sonar (originally an acronym for sound navigation and ranging) is a method which uses sound propagation to navigate, communicate with, or detect objects on or under the surface of water. Active sonar is the process of using extremely powerful sounds from transducers, which are reflected off of objects in the water and captured by a receiver, usually used to navigate through water or gather data. Sonar is currently used often throughout the oceanographic discipline for mapping ocean floors, navigation, and more.

4.1.2 Possible Applications

Sonar could be implemented on the vessel as a form of primary navigation. Active sonar transducers could sit at the bottom of the vessel, facing gently forward, to help the ship navigate through rough and potentially obstacle-ridden waters. Objects in the path of the ship could be detected early and often, and with reliability. Sensors which intake the sonar data reliably would be required to quickly transmit the data to a central computing unit which calculates distance and relative angle from sonic echo time. This data could then be transmitted to the operator using another method specified in the Communication/Navigation Technology Review.

4.1.3 Analysis

Active sonar has two main performance limitations which could lead to incorrect data: noise and reverberation. Each of these can be considered separately, because one will naturally dominate the other. In the event of noise limitations, the accuracy of active sonar can be predicted using the following equation:

$$SL - 2PL + TS - (NL - AG) = DT \tag{1}$$

where SL is the source level, PL is the propagation loss, TS is the target strength, NL is the noise level, AG is the array gain, and DT is the detection threshold. In conditions which are limited by reverberation, the accuracy of sonar can be predicted using the following equation:

$$SL - 2PL + TS = RL + DT \tag{2}$$

where RL is the reverberation level and the other factors are as before. In other words, the range of the sonar would be dependent on the intensity of the sonic pulses and the accuracy would be dependent on the frequency of the sonic pulses. Environmental factors can limit the accuracy of sonar, but the limit can be calculated using hard data gathered by the sensors. In the event of a probability of success being too low, the central processing unit could alert the operator and abort the mission.

4.2 Radar

4.2.1 Overview

Radar (originally an acronym for radio detection and ranging) is a method similar to sonar which uses radiowave propagation to navigate, detect, measure, and track many things and more. Radar was originally developed for military use, but has been used commercially for many years. Rader works in a very similar fashion to sonar, except instead of emitting sound waves, radar emits and recieves radiowaves and radiowave echoes.

4.2.2 Possible Applications

Radar could be used atop the vessel or on the vessel's hull. A radar system would be able to detect objects above the surface of the water with very high accuracy. Sensors which intake the radar data would send distances to a central processing unit which would in turn calculate the obstacles' distance and relative angle. Advanced programs could also detect whether or not an obstacle was moving toward or from the vessel. This data could then be transmitted to the operator using another method described in the Communication/Navigation Technology Review.

4.2.3 Analysis

Radar has become very reliable over many years of development and redevelopment. Radar has a few potential drawbacks, in that it could not detect obstacles under the water and that it may not acquire exact measurements, but it should be able to reliably detect surface-level obstacles. However, distinguishing between a large wave and a large ship could potentially cause the radar to give false positives or negatives. One potential benefit of radar is that its vision could not be impaired by water, fog, or thin ice, unlike a computer vision system.

4.3 Combination Systems

4.3.1 Overview

A combination system consisting of sonar, radar, computer vision, or any two of these, could provide for a more robust and well-rounded system. A computer vision program could provide node tracking while radar provides extremely accurate distance and motion measurements, while sonar protects the bottom of the vessel from unexpected shallows or debris. These systems all in tandem may demand exorbitant processing power onboard the vessel, but could also lead to a more reliable method of acquiring data and overall more successul missions.

4.3.2 Possible Applications

A computer vision system with radar could provide incredible accuracy above the water's surface. Obstacle's position and direction could be acquired from long ranges using the radar, and their size and shape could be distinguished using the computer vision system. Meanwhile, under the vessel, sonar could provide the much-needed protection from bottoming out, thereby stranding the equipment and vessel for rescue. All these systems would need to be processed individually, then brought together to obtain real, usable navigation data which would then be sent to the operator.

4.3.3 Analysis

As with all technologies, more systems means more points of failure. However, with more systems comes redundancy, and some redundancy can help alleviate points of failure. There are many potential drawbacks to a system which utilizes all three of these navigational technologies, including cost, time, and reliability of communication between the systems. The combination of these systems would require a very well-written API to communicate between all methods of obtaining raw data, and translating data into usable navigation data.

5 Conclusion

In conclusion, it is my belief that a system which combines the use of one or more of these systems would provide the most robust system. A well-written API which allows each of these systems to communicate with one another would lead to the most reliability and navigational success. However, realistically, the project will have time, hardware, and monetary constraints. A system which utilizes sonar, radar, or computer vision will have large expenses, and systems which use more than one will require even more financial commitment. But a system with redundancy means a system with reliability and, when attempting to traverse glacial waters with an autonomous vessel, reliability is key.

5 Weekly Blog Posts

0.1 Week 3

0.1.1 Chris

Progress

We met with our client this week to narrow down the scope of the project and decided that the Garmen system will be entirely handled by Jasmin. Our goal as a group will be to implement an object avoidance system that can issue course commands using an API developed for the Garmin system.

Problems

No problems so far

Plans

We have all agreed to weekly group meetings after Tuesday's lecture and are working towards getting an official meeting time with our clients.

0.1.2 Michael

Progress: Met with team for the first time and set up a meeting with our client.

Problems

No problems.

Plans

We made the plan meet after class on Tuesday.

0.1.3 Max

Progress

In the last week, our team met with our client for a second time to try and understand their objective for the project better. Using that information, we wrote a rough draft of a requirements document.

Problems

Our team lacks the required information to write a complete requirements document. After talking to our client, we determined that our project will be a computer vision system. However, no one on our team is familiar with developing computer vision applications. We need to gather more information about computer vision before we can narrow down the requirements our system will have. Additionally, we all lack the foresight to create a Gant chart to put in our requirements document, since none of us have any idea about the steps necessary to set up a computer vision system, not to mention how long it will take us as a team to complete any steps.

Plans

As a team, we plan on familiarizing ourselves better with computer vision using Python. Additionally, we plan on meeting with our client again next week (10/21 - 10/25) to examine the physical hardware they use now, and discuss how our system will integrate with their pre-existing system.

0.1.4 Greg

Progress

We met with our client two times. First with only half of our team, then with the rest. We got lots of good ideas and information to understand what we are doing. It also got a lot of us excited to start working on the actual project right now.

Problems

The only problem we are facing is trying to have everyone show up to all of the meeting since there are six of us.

Plans

We will be completing our first draft of the requirements document, and then second draft later this week. Also, we will be meeting with our client at the Lab to see what our vessel looks like, and ask any other questions.

0.1.5 Tobias

Progress

We have finally met with the lead enough to a point where I feel comfortable with my understanding of the project and what is expected of us as a team. We have decided on Python and on what we want to be doing as a team, we know what the end result should be and what it will likely be, and we know how to go about proceeding.

Problems

Beginning, I have literally zero experience with computer vision and especially not in Python. I have yet to do much of my research that I'm sure will be required of me if I am to contribute meaningfully in this group. I want very badly to do so, but I am worried that I will not be able to unless I dedicate huge amounts of time to this project early in this term—time that I do not have with my two jobs and 19 credits.

Plans

The team will decide on what to do with the system, we will do some research and get some initial code going, then we will travel to Newport to get some sample image and video to do testing on. I'm not sure if we will use machine learning or not, but I think that we are going to do something of the sort. We were told that we might also have some other forms of sensors i.e. infrared, thermal would be amazing but is doubtful, and basic 360 vision.

0.2 Week 4

0.2.1 Chris

Progress

We have met with our client once again at OSU's Buoy Laboratory to inspect the current state of the Automated Research Vessel. This gave me a good idea of how our physical system will be integrated into the craft. The requirements document is coming along nicely. As we learn about the technical issues we are facing, the document is slowly becoming less and less vague.

Problems

No problems as of yet. The team seems eager to get started on the programming side of the project.

Plans

We have been communication with the Garmin representative through our client and intend to collect some sample video to train our system in the near future. Currently this is expected to involve going to Newport OR and placing cameras on one of Garmin's ships and maneuvering around the marina while recording.

0.2.2 Michael

Progress

We have created the requirements documentary and met with our client again to discuss further logistical problems.

Problems

I have talked with all of the members of my group and no one can find anything on canvas or emails about the peer review.

Plans

We plan on meeting up on Tuesday after class to discuss our plans for next week.

0.2.3 Max

Progress

This week we met with our Capstone advisor and discussed the physical system we will implement. We examined the autonomous marine vessel system we are integrating with, and got an inside look into how it is being developed and designed. We used information from that meeting to continue our work on our requirements document.

Problems

The main problem for this week was that I was incredibly sick the whole time. From Monday to Friday, I was feverish and congested, and had a difficult time doing productive work.

Plans

We plan to continue next week researching pre-existing solutions to autonomous marine navigation and obstacle avoidance, and researching what tools we can use to implement our system.

0.2.4 Greg

Progress

This week we met with our client again. This meeting involved us going to their lab, and they gave us a tour of our hardware. We reiterated what needs to be done (what's required) and potentially solutions. We also met with our TA who ensured everything was going smoothly. Lastly, we worked on our requirements doc for the second draft submission.

Problems

The only problem was getting every one at every meetings. Most can't make it.

Plans

There is no set plan for the coming week.

0.2.5 Tobias

Progress

Requirements document is much more concise and cohesive. The team really came together and worked diligently on it. The design requirements are looking robust and all-encompassing, which is great now and will be even better in the future when we begin to really create the system, using the requirements as a guide.

Problems

Everyone was given a chance to go in to the ocean research building on 30th and see the lab where ROSS and the development system will be made. Which is great! The reason I've listed it in "problems" is because I went to the wrong location and did not actually get to see the lab! I'm super bummed about it but my group has been extremely kind and forgiving to me.

Plans

This coming week I plan to get a massive head start on the assignments and the documentation that we need to have in place by the end of this week. I will be working heavily with stretching possible scenarios, including lots of edge cases and whatever I can do to make the next week a bit easier for the team.

0.3 Week 5

0.3.1 Chris

Progress

We have begun splitting the project into its logical components to begin research on the technical document. The largest unknown is the computer vision module that will actually identify objects that could pose a threat to the vessel. We have found several research papers regarding obstetrical avoidance on autonomous ocean vessels and have begun taking notes.

Problems

The class requirement that we each have 3 parts to discuss in the technical document has been troublesome. We have a group of 6 people so we will have to break our logical parts down to pretty granular pieces.

Plans

Our plan this next week is to work on researching component pieces of our project and documenting said research.

0.3.2 Michael

Progress

All assignments are up to date.

Problems

No problems to speak of.

Plans

Currently we are working on technical review assignment individually and will meet up with the group after class on Tuesday to discuss plans.

0.3.3 Max

Progress

This week as a team we began to outline the pieces that will make up our project. At an individual level, I began learning computer vision with Python. I installed OpenCV, and began experimenting analyzing videos frame-by-frame.

Problems

A problem that our team has run into is that since we have six people, we need to divide our project into 18 pieces for the technology review. This has been difficult.

Plans

As a team, we plan on finishing dividing up our project this weekend, and discussing how the pieces of our project review fit together. On my own, I plan on investigating more into computer vision, especially critical point detection. Also, I plan on investigating prior research into marine obstacle avoidance – the area of our project.

0.3.4 Greg

Progress

We completed our Requirements Document. We met for a team meeting to discuss how to split up our project for the Tech Review Document.

Problems

A few others and I were unable to attend the meeting with our TA. No other problems occurred.

Plans

We are going to do a final meeting, whether that be in person or online, to figure out the final split up for our project. Each person will then chose what they will be responsible for in our project.

0.3.5 Tobias

No submission

0.4 Week 6

0.4.1 Chris

Progress

We completed our individual tech reviews.

Problems

This week was particularly difficult to find time for document research with midterms still going on. We had difficulty splitting our project up into 18 parts (3/parson) for the tech review.

Plans

Build an outline for the design document and research the IEEE overview of design documents.

0.4.2 Michael

Progress

The team has split the Tech Review into 6 different modules and each team member is working on the final version of their Tech Review.

Problem

There was a little bit of confusion while generating ideas for the Tech Review and some of the ideas chosen do not fit the criteria for a tech review. The team has discussed new possible ideas for those who need new topics.

Plans

The team plans to continue working on individual papers and meet up after class on Tuesday.

0.4.3 Max

Progress

This week we continued to work on our individual technology reviews. Additionally, as an individual, I continued experimentation with OpenCV and other computer vision tools.

Problems

We continued to struggle subdividing our system into 18 pieces, each which make use of different pieces of pre-existing technology.

Plans

As a team, we plan on working on the Design Document, and working out step-by-step what we will try to do in order to implement our system. We also plan on reaching out to our client again to verify requirements and seek clarification on others.

0.4.4 Greg

Progress

We met with our TA to discuss the upcoming assignments (Technical Review and Design Document). Our Technical Review assignment due date was extended by two days.

Problems

Not having everyone at the TA meeting.

Plans

To finish the Technical Review, and work on the design document.

0.4.5 Tobias

Progress

After breaking down our technologies into specific chunks, our entire group now has a better grasp on what the project as a whole entails, and specifically which groups will be the most important. We are beginning to understand more completely which pieces of the puzzle are most important, such as creating an algorithm to detect nodes or which library to use when creating said nodes.

Problems

I think my main problem is that I have no clue if my tech review was even in the correct wheelhouse. I had a very odd topic and I don't think any peer review would be able to fully capture whether or not I did it correctly. I had a tech review of "alternative systems" whereas everyone else already knew what we had assigned. Since our group was so large, we had to break down and granularize each individual technology that we could muster for our project, and even then we are short.

Plans

Right now the team is delving into OpenCV, an open-source computer vision Python API which we think could be extremely useful in the coming months. We are also looking ahead to the Design Document, as we have already begun to create design decisions that will go well into the next assigned piece of documentation (and yes, it was meant originally to be in the tech review, but the TA pointed out our mistake to us).

0.5 Week 7

0.5.1 Chris

Progress

We have an outline finished for the design document and have met to discuses our decisions we researched for the technical document. Further, we have decided to use a stereo camara system for our computer vision input device and finalized peer to peer communication protocol (LAN) for interfacing with our clients control unit.

Problems

We have had some difficulty meeting with the whole group, but we are managing to keep each other informed via discord.

Plans We have a full group meeting planned for this upcoming Monday to build the second draft of the design doc.

0.5.2 Michael

Progress

We have started draft number two of our design documents. The team met on Thursday to discuss current project goals.

Problems

There have been no problems this week.

Plans

The Team plans on meeting early next week to further discuss the design document and plan future communications with our client.

0.5.3 Max

Progress

This week we submitted a draft of our design document. The draft consisted of only an outline. I continued to play around with OpenCV, since this is the suggested computer vision framework to use. We met as a group on Thursday. At the meeting we talked about the direction of our project, and how some pieces of the design will work. We made the decision that we want to use a seperate computer from the navigation computer on board the boat for our system, and that we want to use stereo cameras for object detection. This means that if we are required to do object detection at night, we must use four cameras, since we would require two infrared cameras in addition to two visible light cameras.

Problems

I have mounting concerns about the feasibility of the object detection system, though do not have enough experience yet with computer vision to determine if the concerns are valid. I am worried that the method of detecting objects by tracking groupings of critical points with a monocular camera will be too inaccurate for our purpose, and tracking critical points with binocular cameras will be computationally expensive.

Plans

As a team, we plan on meeting up early next week to discuss in greater detail our design plan, and work on our design document. We are tentatively planning on meeting Monday, as long as all members of the group are available.

0.5.4 Greg

Progress

Our team met up before our TA meeting, and discussed how the design doc will go.

Problems

I was unable to attend that meeting as well as the TA meeting.

Plans

We will be getting together either next Monday or Tuesday to thoroughly discuss the design doc

0.5.5 Tobias

No submission

0.6 Week 8

0.6.1 Chris

Progress

Tuesday we met as a group to discus the future for our project and how to convey our thought in the design document. We began editing the document soon after with the changes we made.

Problems

No issues other then course work from other classes conflicting with writing time.

Plans

We intend to be finished with the design document by Saturday Nov. 23rd and submit it to out client for approval.

0.6.2 Michael

Progress:

The group met Tuesday to discuss our progress on the design document and discuss our plans going forward. Max drafted an email about our plans to the client that the group approved and it was sent Tuesday night. The client responded to the email and was happy with the direction we are going so far.

Problems

No problems to speak of.

Plans

We plan to finish the design document and meet again next week. We will discuss issues as the come up on discord.

0.6.3 Max

Progress

This week, we worked on our design document. We met as a group and discussed challenges, areas where our design might change, and what each person's concerns are about the project. I also wrote an email that was reviewed and approved by the team members, to our client, stating technical concerns we have and our plans to deal with them.

Problems

With our design, we have discovered potential issues that may make our project unfeasible. As a group, we are worried that object detection will be difficult to do in a reliable manner. We are also worried that our system will require more computational resources and electrical power than is available. Additionally, our client has asked for a system that functions reliably during the day and at night, but we believe it is better to focus our effort on detecting objects during the day since this is a difficult, but more tangible, objective.

Plans

We plan on continuing revisions to our design doc. Additionally, we plan on reviewing past documents and making any necessary revisions. We plan on sending the required documents to our client next week (between the 25 and 27) so they can look them over and provide approval or feedback.

0.6.4 Greg

Progress

We have been working on our design document this entire week. Our team met on Tuesday to discuss how the design document will pan out.

Problems

I was unable to attend this weeks TA meeting.

Plans

We will finish our design document tomorrow.

0.6.5 Tobias

No submission

0.7 Week 9

0.7.1 Chris

Progress

This week I updated my Tech document to reflect some changes made by our team. We decided to use Stereo Cameras instead of monocular cameras; further, we chose to use a GPU for hardware supported computer vision.

Problems

No problems this week, not much happened due to it being thanks giving week.

Plans

We plan to meet next week as a group and hopefully with our client to go over the design document. We will also be working on the final report.

0.7.2 Michael

Progress:

The team has been getting the final document ready to send to our client.

Problems:

No problems to speak of.

Plans:

We plan to meet Tuesday to discuss further plans.

0.7.3 Max

Progress:

This week, we have finalized the design doc and I have reviewed and edited some of the other documents that must be sent to our client. Since this is a holiday week, less progress has been made than usual.

Problems:

Last week, we emailed our client on Tuesday with the understanding that the Design Document was the only document requiring approval, telling them that we would provide the document this week. Since being notified that we need to submit the other documents completed this term, and the progress report which has not yet been written to our client, we have had to shift our timeline and have not been able to provide our client with the documents for verification within the time window we said we would.

Plans:

We plan on writing the Progress Report within the next week and sending it, along with the other completed documents from the term, to our client for approval.

0.7.4 Greg

Progress

No new update.

Problems

No problems.

Plans.

We are planning to send our design document to our client ahead of the upcoming due date.

0.7.5 Tobias

No submission

1 WINTER TERM 2020

1.1 Week 1

1.1.1 Chris

Progress

We met with our client to touch base and set up regular meeting times. With our clients approval, we have begun researching whether to use a manufactured stereo camera or build our own device and calibrate it ourselves. Any device we use must be waterproof and our client offered a couple of GoPros for a prototype if we can stream video to a laptop.

Problems

One of our team has dropped the course. This is not a large problem as we still have 5 members, but it was inconvenient to not have been informed about her absence from the school.

Plans

We plan to have a working data collection prototype (a stereo camera system streaming to a laptop) by the end of next week. We will be meeting with our client again on Wednesday and plan to start meeting him at his lab.

1.1.2 Michael

Progress:

The team has met with our client, Jonathan Nash and set up a regular meeting time. We have also set up time in the lab to work on the hardware aspect of the project. During our meeting we discussed research that needs to be done before our meeting next week.

Problems:

We are pretty sure one of our team members has dropped the class.

Plans:

Meet with the group to discuss our camera options. Go to the lab and work on a prototype stereo camera system.

1.1.3 Max

Progress:

This week we met as a group both with and without our client. When we met without our group, we discussed ways to learn Open CV and what to talk to our client about. With our client, we talked about plans to collect test video to use to build our object detection system.

Problems:

There were no problems within our group. Personally, I had to go out of town to a job interview, so missed out on time to work on our project.

Plans:

We plan on building a camera system to collect test data and do additional learning about OpenCV in the following week.

1.1.4 Greg

Progress

My team met up with our client on Wednesday. They discussed how this term is going to pan out and other logistics.

Problems

I was unable to attend said meeting due to being sick.

Plans

Next week we will have more of an understanding on what hardware we will be using, and I will be attending all meetings next week.

1.1.5 Tobias

1.2 Week 2

1.2.1 Chris

Progress

We have procured a set of GoPro Hero3+s and have begun configuring them to live stream video to one of the team's laptops. Problems

Progress

was slow this week. We didn't receive the cameras necessary to proceed until Friday afternoon. I think the group assumed the cameras were already available at our client's lab and we would get started Wednesday during our meeting; Unfortunately, we needed to check the devices our from another department. Once our client learned we decided to use the GoPro cameras he assisted our acquisition of the necessary equipment.

Our first attempts at live streaming video from the GoPros using VLC media player was unsuccessful. We learned the is an issue regarding VLC versions that prevents WiFi video streaming from GoPro devices in more recent versions of the media player.

Plans

Over the weekend we plan on experimenting with previous versions of VLC to attempt and live stream our GoPros; failing that, we intend to try streaming video directly to OpenCV. If we are unsuccessful at live streaming, we will not be able to use GoPros for the project other than to get sample video.

We will need to procure a set of webcams if the GoPros are not suitable for the project. Using webcams will affect the hardware complexity since we must waterproof the devices. Our client has some experience waterproofing equipment so the added complexity shouldn't be too challenging.

1.2.2 Michael

Progress

The group met on Wednesday to discuss our plans for the week. We decided that we needed to get the cameras calibrated as soon as possible, however, not all of us could meet again so myself, Greg, and Christopher met up Friday to work on calibrating the cameras.

Problems

The cameras our team has access to are GoPro hero 3+. We were planning on connecting wirelessly to the cameras and streaming them to our computer but we may not be able to connect more than one camera and since our system is intended to be a stereo camera system we may try and use different cameras.

Plans

Those of use who are free this weekend are going to meet up and work on creating a stereo camera system with the GoPros this weekend. The group as a whole will meet with the client on Wednesday at the clients labs.

1.2.3 Max

Progress

As a group we made a plan of how to collect video data that we will use in building our system. Additionally, we obtained two GoPro cameras which we will be able to use.

Problems

We had no significant group problems.

Plans

We plan on calculating the optimal camera mounting position and constructing a camera system to collect test video in the next week.

1.2.4 Greg

Progress

Our team met two times this week. We met to discuss how to start the project, and then started the project. We were given two GoPro Hero 3's to work with.

Problems

On Wednesday there was a bit of miscommunication with meeting our client. We met at the lab while our client was at Kelly. Due to this we were unable to talk with out client this week.

Plans

We are going to start working on setting up the streaming of the GoPro footage to our laptop. After we find a reliable way to do this we will begin to setup a mounting mechanism. This will be used to start working on calibrating the two GoPro's for a stereo camera.

1.2.5 Tobias

1.3 Week 3

1.3.1 Chris

Progress

We determined that using GoPros for our camera system is too difficult to obtain reliable streams for our purposes and have purchased a set of webcams as an alternative. We have begun calibrating the cameras to build a camera matrix that can be used to solve lens distortion for the next phase of the project (building a depth map).

Problems

No major problems, just research.

Plans

We intend to have a working depth map that we can begin experimenting with by the start of next week.

1.3.2 Michael

Progress

The team has done individual research over the weekend and we have decided to purchase two cheap Ligitech webcams to replace the GoPros. We are currently working on calibrating the two cameras and trying to make a depth map.

Problems

The GoPro cameras were not able to be streamed to the same computer at the same time due to hardware and software constraints of the cameras.

Plans

We plan to continue researching the python opencv library and work to get the two logitech webcams into a stereo system as soon as possible.

1.3.3 Max

Progress

This week we aquired two web cams, have built a test stereo mounting system for them, and have verified they can interface with OpenCV to stream video or capture image frames. We've also worked to install OpenCV on all personal computers which didn't have it installed before.

Problems

Originally, we planned to use GoPros in our stereo camera setup. However, we could not get two GoPros to stream to OpenCV at the same time. Thus, we switched to Logitech webcams.

Plans

In the next week we plan on calibrating our stereo cameras and writing code to produce depth maps. From the depth maps, we will work to extract objects. Additionally, we will waterproof our camera system, most likely with waterproof containers that our client already has.

1.3.4 Greg

Progress

Some of us got together to start working with two new cameras we have purchased. These cameras are going to be used in serial. We were able to get live video feed from the OpenCV python library, but were unsuccessful in getting them to work in serial

Problems

Some of our team was unable to attend the meetings.

Plans

We will be working on this over the weekend to get the cameras to work in serial.

1.3.5 Tobias

1.4 Week 5

1.4.1 Chris

Progress

We were able to create a depth map from our stereo vision system. The map is not currently very useful as the configuration is not correct yet. We have ordered material for waterproofing the system but have not delivered yet.

Problems

There were concerns brought up about whether two webcams can actually be used for dynamic object detection. Some documentation suggests that using separate cameras work fine for a static scene (no motion) but fail with a dynamic scene due to the device clocks being out of sync. This could potentially derail our efforts so far.

Plans

We plan to research whether or not we need to pivot on our camera system and purchase a synchronized stereo camera or move to a monocular system. We plan on waterproofing our system next week when our supplies have arrived at the lab. If we decide to continue with our current system, then we plan on finding the correct configuration for building our 3d depth maps.

1.4.2 Michael

Progress

The group met up a couple times this week to get our stereo camera system up and running in order to collect live marine data from the clients boat this weekend. We also worked on converting the stereo images into a useful format for our project, such as depth maps.

Problems

Unfortunately our waterproof casing didn't arrive on time to send our system on the trip to gather data. We also are running into conflicting information about whether or not we will be able to use USB webcams to achieve our goals.

Plans

The team plans to collect data next week locally with our system. We also plan on doing further individual research and meeting up to talk about it next week.

1.4.3 Max

Progress

This week we created a program to take live video feeds and produce depth maps to detect distance. We also ordered a waterproof case to house our cameras, and created a program for our client to use to collect data for testing for us.

Problems

The camera case we ordered to waterproof our cameras did not arrive when expected (by Thursday), so we were unable to finish the test system for our client to collect video this weekend.

Plans

We plan on better calibrating our cameras after we receive the waterproof case for them. We also plan on fine tuning our depth map. Additionally, the next steps we will take will be to create a module to detect objects in a video feed and create an algorithm to identify objects which pose a thread to the vessel on which the system is deployed.

1.4.4 Greg

Progress

This week we got together for our weekly meeting. We all spent a good amount of time trying to work with the OpenCV library to get stereo cameras calibrated for object detection. We also got a simple system up for our client to go in open waters to get data video feed of real ocean conditions.

Problems

We are having a lot of trouble with the OpenCV library to work with stereo cameras. The calibrations are not working optimally and another solution may be needed. Another problem is that our simple setup to get video feed was unable to be used this week with our client. We were expecting a water proof case for the cameras, but the it did not arrive in time. The data collection will occur another week.

Plans

Two of us, myself included, are testing alternative options to use machine learning to do object detection. The other of us are still trying to get the cameras setup for object detection.

1.4.5 Tobias

Progress

This week was an important one. We have decided to go back and review our project scope and present to our client a scope which we think is more probable and realistic scope for us.

Problems

Obviously our return to nothing has presented us with a time crunch. But we are working through it and with the client on our side we shouldn't have any more issues.

Plans

This week we will focus on getting everything in place for the alpha build and presentations for March. We will stick to our plan and attempt not to deviate.

1.5 Week 6

1.5.1 Chris

Progress

We have met with our client about re-scoping the project requirements to be realistic for what can be accomplished by the end of the term. We are re-focusing our efforts on object identification and classification, using the YOLO neural network and the Singapore marina dataset and have decided to purchase a stereo camera instead of constructing a system ourselves.

Problems

No major problems other than lack of time. We believe we now have a good direction to continue with the project, though whether we have sufficient time for real progress, remains to be seen.

Plans

Test pre-trained neural networks to determine whether our system will have to be trained or if we can use existing neural nets. Finalize a purchase order for our stereo camera.

1.5.2 Michael

Progress: The team met up at the start of the week online using discord to discuss the direction of the project. We voiced concerns with a few of the objectives and decided to re-scope our project to focus on the object detection and distance parameters needed for object avoidance instead of the whole system. We met with our client and discussed the re-scoping and came to a verbal agreement on what we are going to be working on. We have decided to split the project up into parts, object detection and object distance.

Problems: Obstacle avoidance systems are incredibly complex cutting edge technologies and we do not have the knowledge or man power to complete one in the time given.

Plans: We plan to meet again next week to discuss progress and continue working on the project. We also plan on writing documentation on our change of direction and presenting it to our client, we will move forward from there.

1.5.3 Max

Progress

This week we made further progress calibrating two webcams to produce stereo depth maps. We also put together code to use pre-trained neutral networks to do object recognition with a video feed, and found a dataset to train our own neural network on.

Problems:

Since the webcams we are using for stereo vision are not synchronized and include on-board motion stabilization code, they do not work well producing depth maps when the cameras are in motion. Since in a marine environment the cameras will be in near constant motion, we have realized we will need a dedicated stereo system.

Plans

We plan on finding a stereo camera which will meet our requirements. We will continue working on stereo depth mapping. Also, we will determine whether a pre-trained neural network will meet our requirements and if not train our own.

1.5.4 Greg

Progress

This week we made good progress. We started using some pre trained neural networks with an object detection algorithm YOLO to detect and classify objects.

Problems

We have determined that our Logitech stereo webcam setup is not functional at all. The two cameras can't be synchronized without devoting a lot of time. We have since put off the depth map creation since.

Plans

We plan to get the object detection to function better. We will use better test data (from what our client collects) to ensure classifying objects works. We will be buying an actual stereo camera to produce working depth maps.

1.5.5 Tobias

Progress

This week was an important one. We have decided to go back and review our project scope and present to our client a scope which we think is more probable and realistic scope for us.

Problems

Obviously our return to nothing has presented us with a time crunch. But we are working through it and with the client on our side we shouldn't have any more issues.

Plans

This week we will focus on getting everything in place for the alpha build and presentations for March. We will stick to our plan and attempt not to deviate.

1.6 Week 7

1.6.1 Chris

Progress

We are making progress. I am currently researching how to interface with the USVs 3DM-GX5-35 Attitude Heading Reference System (AHRS) which will give us access to positional data on the USV (pitch, roll, yaw, and compass direction). We will use positional data to correct for movement when matching subsequent frames to detect if an object is moving relative to the USV.

Problems

No major problems other than lack of time. Now that we have worked through much of what won't work, we are on our way to a minimum viable product.

Plans

I plan to begin implementing a python interface for the (AHRD) and have it ready to test by the end of next week.

1.6.2 Michael

Progress

I was unable to meet with the group this week. Max and Christopher met and discussed the direction of the project and set up a list of tasks that need to be accomplished. I met with Chris to discuss what he and Max had talked about.

Problems

Outside of being sick and missing the group meeting there was no problems this week.

Plans

The discussed tasks will be assigned to group members to work on.

1.6.3 Max

Progress

This week I implemented code to use a SSD inception v2 network trained on the Singapore Maritime dataset with OpenCV to identify boats and obstacles in input images. This is part of the main obstacle detection module, which given input from a video feed, will detect obstacles in relation to the ROV.

Problems

Our group continues to struggle with implementing a stereo camera system. Although we are able to create stereo depth maps when the two webcams we are using are at rest, when the cameras are in motion we are unable to produce accurate depth maps. Since on the boat, the webcams will constantly be in motion, we are worried about the feasablility of using stereo vision to calculate distance.

Plans

This week we outlined plans for each of our group members. We have five tasks we are trying to accomplish in parallel. We are working on detecting obstacles in image frames captured from a video camera that will be on board the vessel, and tracking the detected obstacles between frames. Additionally we are working to create an interface to send serialized data from one linux computer to another in real time, in order to transfer the obstacle data we produce to Jasmine's system. Also, we are working to retrieve GPS, gyroscope, and accelorometer data from the main ROV computer system, for use with our system. Finally, we are working to develop modules that standardize images based on input positional data from the gyroscope (in order to correct for differences in pitch and rotation), and to calculate the angular distance to objects detected in an image relative to North.

1.6.4 Greg

Progress

We made some small progress on some more object detection algorithms. We potentially found a algorithm that did not rely on a GPU which will provide for a much easier implementation. I also was able to do object detection using a NVIDIA GPU which drastically increased the speed of the algorithm.

Problems

There were not any problems

Plans

We plan to create a end roadmap of what's to come. We have assigned everyone to perform one of tue tasks along the roadmap to ensure we have some sort of viable product

1.6.5 Tobias

No submission

1.7 Week 8

1.7.1 Chris

Progress

Finished script to parse Inertial Measurement Unit (IMU) data. The IMU relays attitude measurements which we will use to adjust for the motion of the USV between image frames. Began work on applying IMU data to frame transformations.

Problems

No major issues to report.

Plans

Finish presentation slides over the weekend for tech demo. Do a dry run of the tech demo presentation with the group. Continue integrating project components together.

1.7.2 Michael

Progress

Chris, Greg, and I worked on devising a way to parse the Inertial Measurement Unit data (IMU) that the ROSS collects. We are planning on passing this data to other modules to translate images to state as if they were all taken at a horizontal angle. I was also able to put some work into a python module that will measure the angle that an object is from the ROSS.

Problems

We ran into a small snag when preparing to parse the IMU. We were able to work with other members of the ROSS team to overcome this obstacle quickly.

Plans

We plan to work on the project and continue to prepare for the design review over the weekend. We also plan on meeting up at the lab to work further on the project on Monday and Wednesday.

1.7.3 Max

Progress

This week I updated the object detection code so that objects found with the Tensor Flow neural network can be tracked between frames in a video feed. Additionally, I created a way to easily store data for each object in every frame. Our team also worked on getting positional data from the ARV in order to correct for motion between images, and has been working on how to do the correction transformations, and how to find bearings to objects detected.

Problems

No major problems to report for this week.

Plans

We plan on putting together all the parts we've been individually working on the create a robust obstacle detector that delivers the bearing from the ROV to obstacles.

1.7.4 Greg

Progress

We were able to get together to begin working on the object detection, image manipulations, and IMU parsing.

Problems

There were no problems.

Plans

We plan to get together this weekend to do our capstone review assessment.

1.7.5 Tobias

Progress

This week our group has made a lot of progress. We met multiple times this week to discuss our individual tasks and so all set out to do them.

Problems

I don't think we're prepared for our presentation on Tuesday. But we're preparing.

Plans

We have to get our functionality down pat before next week, so we're planning on putting together our major parts of the project next week and finishing everything up.

1.8 Week 9

1.8.1 Chris

Progress

Consolidated IMU code, Image processing code, and object detection code into a single repo. Began unit testing components. Refactored IMU code to be more generic across the project. Integrated IMU code into the main project entry point.

Problems

No major issues

Plans

Collect sample video and IMU attitude data during a ROSS test run. Set up a tough book for collecting data. Continue integrating project components together and testing individual components. Begin testing with live collected data.

1.8.2 Michael

Progress

The team met Monday to go over the presentation and do a few practice runs. We met again Wednesday to continue working on the project and plan out what else we wanted to do for the rest of the week. We decided that we needed to get in contact with Jasmine who is in charge of the laptops and coding part of our clients research work to figure out some logistic problems in relation to collecting video footage. I sent an email to her on behalf of the team requesting equipment for the up coming trip.

Problems

No problems to speak of yet.

Plans

We plan on collecting data on board the ROSS with a mounted camera to obtain real data that we can test our system against. The ROSS is currently scheduled for a trip to Newport on Tuesday and we need to have everything ready by then so that we can retrieve footage.

1.8.3 Max

Progress

This week we continued work on our object detection and tracking system. We integrated the code we wrote to capture IMU sensor data with the module to correct for roll in an image, and are using that to create a module which can capture a image and correct it based on positional data for the boat. We also did our design review this week on Tuesday.

Problems

No new problems to report.

Plans

We will continue to work to integrate the components of our software to finish up our beta release. Within the next week we will have code that is able to correct captured images based on positional data from a sensor, and do persistent object tracking for boats detected in multiple frames in the image. We will also work on producing an informative output from our program so that the data we collect can be displayed on the client's pre-existing GUI. Finally, we will have our client capture more real-world footage this upcoming week to verify our system performs as expected, and do integration testing on the system as a whole.

1.8.4 Greg

Progress

This week was our design review. The review went smoothly, and we all did a great job. We started to work on getting a working platform that incorporates all of our project components into a single working project.

I was able to complete the image manipulation code. Our images are now able to be transformed based off the vessel's pitch, yaw, and roll.

We got a system working for our client to capture video on board the vessel.

Problems

Some of us were not able to meet together every time.

Plans

Our client will be going to Newport next week. With the video capturing system our client will be getting video for us to test with in Newport. This video will be tested with its corresponding IMU data so we can fully test our project.

1.8.5 Tobias

No submission

1.9 Week 10

1.9.1 Chris

Week 10 Progress

Completed demo video recording. Not much else since the group had other course obligations this week.

Problems

No major issues

Plans

Complete end of term report.

1.9.2 Michael

Progress

The team met Wednesday to discuss plans for the week. We met again Friday take pictures and record video parts.

Problems

No problems to speak of this week.

Plans

The team plans to meet online this weekend to continue working on the end of term reporting paper as well as meet up on Monday to before we turn in the paper and video.

1.9.3 Max

Progress

This week we worked with our client to get video data from what the vessel sees on open water. We also integrated the different programming modules. In addition, we made a video demonstrating our beta.

Problems

While our client was able to provide us with some useful data, the GoPro that was being used for data collection ran out of storage before the boat was launched on open water. Thus, we got video recorded on the cell phone of our client. We are still able to use the data provided, but it is not the ideal outcome.

Plans

We plan on completing our beta video this term and writing our end of term report. Next term we will kick off by doing more integration testing of our system.

1.9.4 Greg

Progress

This week we all got together to work on the final assignments due for this class. Today Friday the 13th, we all got together to work on the demo video. We recorded all aspects of the video. Our client also went out to Newport earlier this week and got some test footage for us to work with. This footage was on the ROSS itself capturing multiple object in the path.

Problems

There were no problems this week.

Plans

We plan to cut and finalize our demo video. After all the included parts are made we will submit the assignment. We then also plan to work this weekend on making the final paper for the class. Lastly, after we finish all of the assignments, we will continue to work on coding with the test footage.

1.9.5 Tobias

Progress Beta video is almost complete, and we are beginning work on the winter write-up. Everything else is almost complete or complete and everything is coming together!

Problems I have a crazy schedule this week so I am running into timing issues getting together with the group.

Plans We only have a few plans for the upcoming term but we are planning to meet soon to figure it all out.

6 Showcase Poster

Obstacle Avoidance System for the ROSS USV

Unmanned Surface Vehicles (USVs), such as Robotic Oceanographic Surface Sampler (ROSS), must be able to avoid obstacles in their path to prevent equipment damage and as a safety measure for other maritime vessels. The Autonomous Vessel Vision System (AVVS) utilizes a front facing monocular camera stereo and a YOLO neural network to Identify potential obstacles and report them to the ROSS's operators. This is a first step in creating a fully autonomous object avoidance system which is planned for development next year.

THE ROSS USV

The ROSS Unmanned Surface Vehicle (USV) collects oceanographic data on salt/fresh water interfaces to better understand how these systems interact with ocean currents. These interfaces are commonly found near glaciers where melting fresh water ice meets ocean salt water. The ROSS USV was built as a safe means of collecting data from the inherently dangerous glacial environment.

An Object Avoidance System allows ROSS to maneuver around icebergs and other ocean vessels reducing the chance of equipment damage both to the USV and other ocean traffic. This is especially useful for the systems next mission in the gulf of mexico where small fishing boats without transponders will be prevalent.

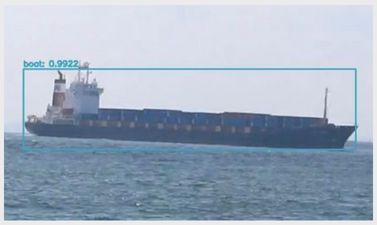


Figure 1. Object Bounding Box and Classification

OBJECT DETECTION AND CLASSIFICATION

Our object detection uses a pre trained neural network made specifically from maritime environments.

The pre trained neural network was created from the Singapore Maritime Dataset - a dataset providing videos and ground truth information of on-board and on-shore data.

The neural network is used with the object detection algorithm You-Only-Look-Once version 3 (YOLOv3). Each frame in a video feed is analyzed with the neural network to provide bounding boxes around the detected objects.

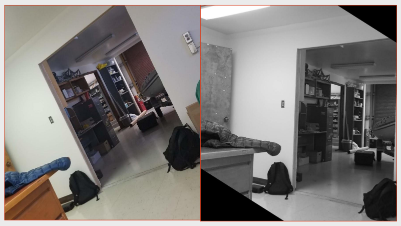


Figure 2. Left: Unaltered input frame (skewed roll ~30°). Right: Corrected Image

IMAGE CORRECTION

As the ROSS vessel traverses through the open waters, images taken from the camera are skewed due to the vessel being at different orientations the moment the image was captured. The skewed images affect the object detection and tracking, so it must be accounted for.

The different orientations on a boat are rotational motions: roll, pitch, yaw. Each for the x, y and z axis, respectively. Using the rotational motions and the specifications for the camera the images can be corrected for.

ACKNOWLEDGEMENTS

- Max Harkins, Chris Patenaude, Tobias Hodges, Michael Gabriel, Greg Sanchez.
- harkindo@oregonstate.edu, patenauc@oregonstate.edu, hodgesto@oregonstate.edu, gabriemi@oregonstate.edu, sanchegr@oregonstate.edu
- Client: Jonathan Nash, CEOAS

Detected object position is reported back to the client system as a compiled list of degrees from the ROSS's current heading. This is done by calculating the degree value per pixel of the camera and multiplying this scalar value by the distance in pixels from the centerline (see figure 3).

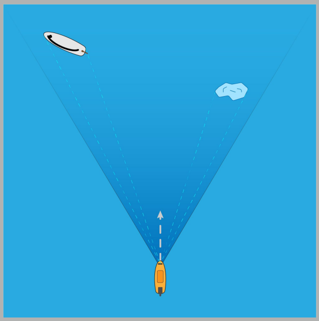


Figure 3. The ROSS detects a boat and iceburge in its field of view. An angle is calculated between the ROSS's heading and and the detected objects.

Team Photo (from left to right): Max Harkins, Chris Patenaude, Tobias Hodges, Michael Gabriel, Greg Sanchez.

Oregon State University

7 SYSTEM OVERVIEW

CONTENTS

1 Project Overview

1.1 Project Description

The purpose of this project is to create a robust object detection system to warn operators of unmanned research vessels about obstacles that present a hazard for c/ollision. The system operates on a low-powered Intel NUC computer on board the autonomous vessel, and is connected to an Inertial Measurement Unit (IMU) sensor and a USB video camera to get data on the environment nearby. The system provides serialized output to the navigation computer listing obstacles which present collision hazards, which can be shown graphically on a GUI used by the vessel's remote operator.

1.2 Required Functionality

This system must be able to detect obstacles on the open ocean. Obstacles that are expected to be detected are boats, swimming people, and floating debris. The system must be low powered in order to function entirely on-board the remotely operated vessel. Additionally, the system must be robust enough to function in marine conditions.

2 System Requirements

2.1 Hardware Requirements

The system is designed to run on a computer with the modern Intel64 CPU architecture. Additionally, a USB camera and IMU must be connected to the computer.

2.2 Software Requirements

The system is designed to run on the Linux operating system. Although the system can be built and tested using Windows, it is not recommended for deployment. Ubuntu is the preferred Linux distribution.

The system's software components were developed using Python 3. Python 3.7 is recommended, though other modern python versions including Python 3.5 and Python 3.6 work with the system also. Along with a modern Python version, the Python OpenCV module must be installed on the system. OpenCV release 4+ is recommended, and a recent version of OpenCV can be obtained by installing the pip package `opencv-python`.

Matlab must also be installed on the system to parse the IMU sensor data. The most recent matlab version is recommended, so as to provide a Matlab Engine API module compatible with Python 3. The project README provides detailed installation instructions for Matlab.

3 System Usage

3.1 System Configuration

Configuration for the system can be done through the **config.py** file, present in the root directory of our system's code. This file contains options to configure the attached IMU sensor and USB video camera for use with the system. Information about configuration parameters is shown below.

- `ROOT_DIR`: This parameter specifies the root directory of the system's software. It is set by default to the directory containing the config file, and does not need to be modified.
- `IMU_PATH`: This parameter specifies the path to a binary file containing IMU data. The IMU should output data to a binary file, and the full path to the binary file should be set as this parameter's value. For testing purposes, a mock IMU data file can be used.

- `CAPTURE_DEVICE`: This parameter specifies the video capture device used by the system to detect obstacles. The video device ID, which typically takes an integer value, should be set as this parameter's value. On Linux systems, video capture devices are listed in `/dev/` by default, with the format `/dev/video*`.
  For testing purposes, this parameter can be set to the path of a video file. The video file will be used in place of the camera as input.
- `FRAME_INTERVAL`: This parameter specifies the interval to wait between processing frames, in seconds. The default value is 1. Processing frames more frequently leads to greater hardware and power usage, but more responsive object detection.
- `VIEWPORT_ANGLE`: This parameter specifies the viewport angle in degrees. This parameter is camera specific, and should be set to the horizontal field of view of the camera used.
- `DRAW_TO_SCREEN`: If this parameter is set to true, images are shown frame-by-frame to a display attached to the system, with bounding boxes drawn to mark detected obstacles. Setting this parameter is useful for testing and debugging. Once deployed, this parameter should be set to false.
- `VERBOSE`: This parameter is set to false by default. If set to true, the system will provide verbose output about operation via the console. Verbose functionality is intended to help with debugging.
- `CAM_FOCAL_LENGTH` and `CAM_SENSOR_WIDTH`: These parameters should be set based on the specifications of the camera used. The values set are used in the Image Correction module to transform images captured by the camera.
- `USE_ROLL`, `USE_PITCH`, and `USE_YAW`: These parameters control whether the Image Correction module attempts to correct images based on the pitch, yaw, or roll of the boat. By default, only roll is corrected for. Correcting for yaw should generally not be done, since the ROV is expected to change direction in the horizontal plane under normal operation (i.e. when it turns).

3.2 System Testing

Unit tests for the system are included in the `test` folder present in the project's home directory. Unit testing is done using python's unittest testing framework. To run all tests, navigate to the project's home directory and run `python -m unittest`.

3.3 System Operation

View the README present in the system's home directory for most recent installation instructions. Once software dependencies have been installed, and appropriate parameters set in config.py, run the system by executing the command `python main.py`.

4 Module Overview

4.1 IMU Interface

The initial Measurement Unit (IMU) is a device that used to sense the orientation of a body in space (in this case, the body is the ROSS vessel). The IMU outputs a live serial data stream to a Microsoft NUC computer located on the ROSS. Orientation data is stored in a binary file –file name here– where it can be accessed by the ROSS's systems.

The IMU Interface for the Autonomous Vessel Vision System (AVVS) parses the most recent entries in the IMU data file and extracts the most recent orientation for use by the other components in the system. The actual parsing of the data is done with a 3rd party MATLAB script developed by an associate of the ROSS project, Dylan Winters. The IMU module interfaces with the parser via the MATLAB Engine API for python. After the raw binary has been parsed, we extract the most recent entry, which corresponds to the current orientation of the ROSS, and store it in a python dictionary. The output dictionary is used by the Image Correction module, to intelligently correct for the ROSS's orientation, and the Position Reporting module to compare detected objects with the ROSS's heading.

4.1.1 Attitude Data

The IMU provides Attitude data in the form of pitch, yaw, roll, and heading. Other positional data exists but is unnecessary for the AVVS to function. The data consists of radian value which correspond to rotations in the x, y , and z axis or, in the case of heading, the relative angle from North the ROSS is facing. When the ROSS is facing North on a level plane, all the previously stated values are 0 (See Figure ¡Fix Me¿).

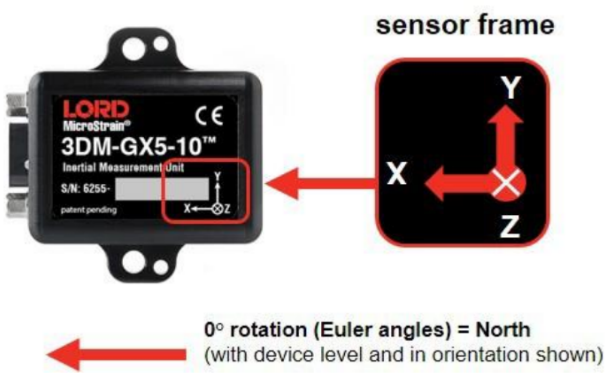

sensor frame
LORD MicroStrain
3DM-GX5-10™
Inertial Measurement Unit
S/N: 6255-
patent pending
Y
X
Z
0° rotation (Euler angles) = North
(with device level and in orientation shown)

Fig. 1: IMU Sensor Frame. !!! Citation Needed !!!

4.1.2 Data structure Overview

Data returned from the parser is structured as a set of nested key:value pairs where values may be floats, float arrays, or strings. Listing ¡Fix Me¿ shows the structure of the returned data and the data type associated keys:

Listing 1: IMU data structure

```
'units': {
   'attitude': {
      'compensated_angular_rate': {
         'X': 'rads/sec',
         'Y': 'rads/sec',
         'Z': 'rads/sec',
         'valid': '1=valid, 0=invalid'
      },
      'gps_timestamp': {
         'time_of_week': 'seconds',
         'valid': '1=valid, 0=invalid',
         'week_number': 'n/a'
```

```
        },
        'heading_update_source_state': {
           'heading': 'radians',
           'heading_1_sigma_uncertainty': 'radians',
           'source': '0=no source, 1=Magnetometer, 4=External',
           'valid': '1=valid, 0=invalid'
        },
        'linear_acceleration': {
           'X': 'm/sec^2',
           'Y': 'm/sec^2',
           'Z': 'm/sec^2',
           'valid': '1=valid, 0=invalid'
        },
        'orientation_euler_angles': {
           'pitch': 'radians',
           'roll': 'radians',
           'valid': '1=valid, 0=invalid',
           'yaw': 'radians'
        }
     }
  }
```

4.1.3 Future Improvements

The matlab engine is relatively large, requiring several gigabytes of disk space. Further, the engine must be initialized after the AVVS starts requiring several seconds to set up. Re-writing the parser in python would simplify the process significantly and reduce the impact on disk space. Future project developers should consider this improvement as a way to streamline the module.

4.2 Image Correction

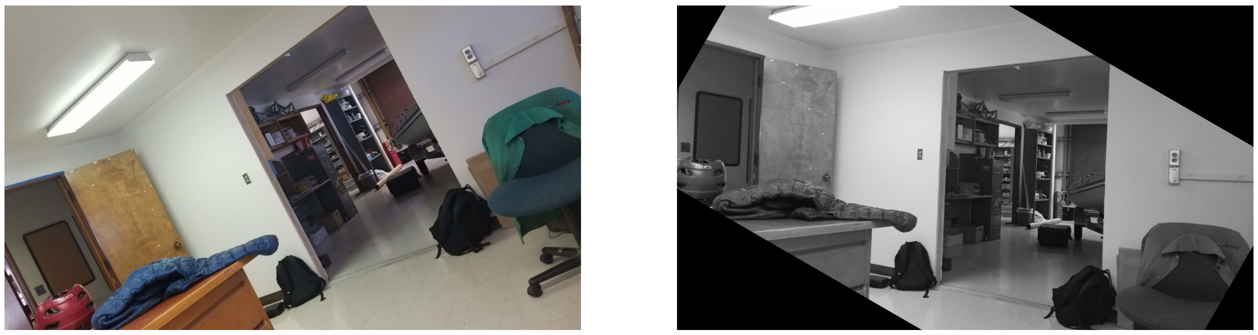
As the ROSS vessel traverses through the open waters, images taken from the USB camera are skewed due to the vessel being at different orientations the moment the image was captured. The skewed images affect the object detection and tracking, so it must be accounted for.

The different orientations on a boat are rotational motions: roll, pitch, yaw. Each for the x, y and z axis, respectively. Using the python dictionary from the IMU (rotational motions) and the specifications for the USB camera the images can be corrected for. The short process is detailed below:

- Creating the 4x4 transformation (homography) matrix
    1) Calibration intrinsic matrix: using the USB camera's focal length and sensor width the pinhole camera model is applied to generate the calibration intrinsic matrix. The matrix determines which incoming light is associated with each pixel on the captured image.
    2) Rotation matrix: using each of the rotational motions Euler's rotation theorem is applied to create three different rotation matrices. Each rotation matrix is then multiplied with each other to create the rotation matrix. This gives the general rotation of the vessel.

3) Translation Matrix: created with a predefined translation to keep the image centered.
4) Finally all of these matrices are multiplied together to create the transformation matrix.

- Rotating the image: an OpenCV function `warpPerspective()` is used rotate the image. The transformation matrix is applied to the unaltered image. The resulting image is a corrected image that is passed to the object detection and tracking algorithms.



(a) Example unaltered input frame (skewed roll by $\sim$30°) (b) Transformed Image

Fig. 2: Image Correction Example

4.3 Object Detection

Object detection is performed by a SMD Inception v2 Single Shot Detector Neural Network. The network is stored as a Tensorflow graph in the directory `object_detection/ssd_inception_v2_smd_2019_01_29` relative to the system's root folder. The Neural Network is loaded and used in the `object_detection/detect_and_track.py` module. Each time an image is captured and processed, this module is used to detect any potential obstacles present in the image by feeding the image to the neural network.

The output from the neural network is an array of single-dimensional arrays, where each single dimensional array represents an object detected by the neural network. Each single-dimensional array contains the following floating point values:

- Class ID: The class ID of the detected object is the first value in each sub-array. The class ID corresponds to the object label, and describes what type of object the neural network has found. This value is an integer cast to a floating point.
- Confidence: The confidence rating is a number between 0 and 1 describing the confidence in the prediction. Values near 1 indicate a high confidence.
- Bounding Box Corner 1: The first corner of the bounding box is represented by the values in the third and fourth indices. The third value gives the x-coordinate, and the fourth position gives the y-coordinate.
- Bounding Box Corner 2: The second corner is given by the fifth and sixth values in the array. The fifth value is the x-coordinate, the sixth value is the y-coordinate. Together with the bounding box corner 1 coordinates, the bounding box corner 2 coordinates can be used to draw the bounding box.

4.4 Object Tracking

Every time an image is taken and obstacles are found using the object detection module, the Object Tracking module is used to associate objects between images using the `update(...)` method. The object tracking module is implemented as the `CentroidTracker` class, which is composed of objects from the `TrackedObject` and `ObjData` classes. The Object Tracking module keeps an up-to-date record of what obstacles might be seen by the camera in the form of a dictionary associating a unique ID with a `TrackedObject` software object. Each `TrackedObject` instance contains information about the current position of the obstacle it represents, whether the obstacle is currently in view, how the size of the obstacle is changing, and an array of past collected data. A UML diagram of how the object tracking module is structured is shown below.

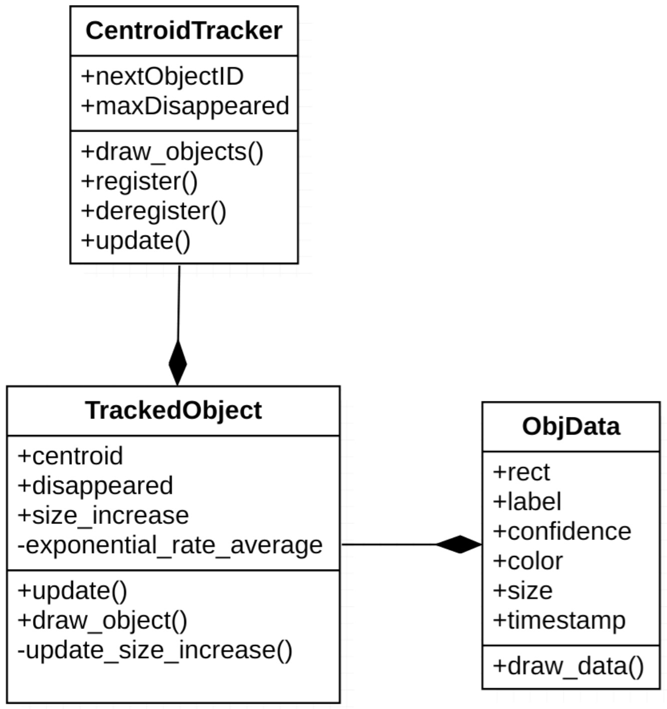


Fig. 3: Organization of Object Tracking module.

The algorithm used by the `CentroidTracker`'s `Update` method to associate classes is a centroid matching algorithm. The method calculates the euclidean distance between the centers of the bounding boxes for each obstacle being tracked in the system to each obstacle passed as a parameter to the `update(...)` method. The method then associates obstacles by pairing them starting with the smallest distance between two objects and working up. When an object tracked by the system is not found in an image, it is marked as disappeared. After being disappeared for a number of frames, the object is removed from the system. The centroid matching algorithm was chosen because its implementation is relatively simple. The algorithm can malfunction when one object occludes another, or when an object disappears from a frame at the same time as a new object enters the frame.

4.5 Position Reporting

Object Position Reporting is handled in a lightweight stand alone function calculate_angle which is found in the object_positioning_processing.py. This function calculates the angle between the the vessel's heading and an object. This is achieved by calculating the value in degrees of each pixel which can be derived from the ratio of pixels to view-port angle and then multiplying this scalar value by the distance in pixels from the center of the image.

5 PROJECT CHALLENGES AND NEXT STEPS

5.1 Initial Obstacles

The system was originally intended to be a complete autonomous navigation system for the unmanned ocean vessel. After multiple revisions of our original design, the group began to lean towards a stereo vision depth-mapping system with a focus on obtaining 3D bounding boxes for objects in the water. The group immediately encountered multiple issues: stereo serialization, camera synchronization, camera distance capabilities, attitude adjustments, supplemental systems, and more. After attempting to deal with these issues one-by-one, the group redesigned and re-scoped the project to focus on robust, consistent object detection and tracking using IMU data and a single camera to return headings of objects in the intended path of navigation.

5.2 Computer Vision Hurdles

Mono computer vision has come with its own set of challenges. Constantly acquiring images from a camera, applying transformations based on IMU data, and passing the resulting image through a trained neural network becomes difficult when given a small amount of processing power. This, along with other hardware restraints, led us to implement a 1 fps system in which the computer would wait 1 second before processing the next image. This led to its own issue, in which the computer would process every image of the camera, but spread them out by one second each. Instead of using a frame from the camera every second, the program would store all the frames captured and then wait between every frame. On top of this issue, the camera was taking in IMU data for yaw instead of allowing the camera to turn. These issues were fixed after our COVID-induced long-distance meetings with one of the researchers. Other issues involved installation due to the IMU data-parsing script needing MATLAB to be installed (which is very large) and Python versioning to run MATLAB from the Python scripts.

5.3 Looking Forward

For the future, the team envisions a 360-degree object detection and avoidance system using OpenCV and Python. The MATLAB script should likely be ported to Python to reduce installation and storage/overhead requirements. The depth calculation should also be re-introduced for added security and more robust autonomous navigation. Ideally, the neural network should be continually trained on the team's own footage and bounding boxes instead of the Singapore Maritime Dataset. This would lead to a stronger, more accurate object detection system over time and can lead to accuracy comparisons and benchmarks with other established object detection systems.

8 Recommended Technical Resources for Learning More

Websites

- Main page for OpenCV documentation: https://opencv.org/
- Singapore Maritime Dataset home page: https://sites.google.com/site/dilipprasad/home/singapore-maritime-dataset
- Robotic Oceanographic Surface Sampler website: http://makani.coas.oregonstate.edu/ross/Overview.html
- PEP 8 Coding Standards: https://www.python.org/dev/peps/pep-0008/
- GitHub Link to Team Githup page: https://github.com/ARV-NAV/AVVS
- MATLAB Engine Python API: https://www.mathworks.com/help/matlab/matlab-engine-for-python.html
- Helpful blog post detailing centroid tracking: https://www.pyimagesearch.com/2018/07/23/simple-object-tracking-with-opencv/
- Archive of pre-trained deep neural networks trained with the Singapore Maritime Dataset: https://github.com/tilemmpon/Singapore-Maritime-Dataset-Trained-Deep-Learning-Models
- Python tutorial on image transformation with OpenCV: https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_imgproc/py_geometric_transformations/py_geometric_transformations.html
- Blog post on stereo vision system construction and use (note: our team did not use a stereo system in our final project. See Challenges section in the System Overview section): https://erget.wordpress.com/2014/02/01/calibrating-a-stereo-camera-with-opencv/
- Resource for using a deep neural network to detect objects: https://www.pyimagesearch.com/2018/11/12/yolo-object-detection-with-opencv/

People

- Geoff Hollinger, Associate Professor of Mechanical Engineering and Robotics at OSU: https://mime.oregonstate.edu/people/geoff-hollinger

9 Conclusion and Reflections

9.1 Conclusion and Reflection - D. Max Harkins

During my time working on this project, I learned a great deal about computer vision and best practices for Python development. Before starting the project, I was completely unfamiliar with image processing using Python 3, and had never used OpenCV before. Additionally, much of the software development I had done was using C and C++, so some Python style guidelines and common conventions were new to me. Even though we did not include a stereo camera system in our final project due to various issues, I though it was very interesting to learn and attempt to implement stereo vision.

In addition to increasing my technical skills, I learned some interesting information about glacial research. Being on a project focused on helping a member of OSU's college of Oceanography prompted me to look through some of his past research done with the ROSS system, which was very interesting.

I learned that with experimental project work, where a team is unfamiliar with the technical components they will be working with, creating an overarching initial design plan is nonsensical. In reality, I held this opinion before the class, but after my experience with the project this idea was confirmed. I learned that making small plans to work towards a larger goal, and being flexible when plans don't work, is the best way to develop an ambitious project. I also learned that being ambitious with project goals can be good, but that project goals must be continuously re-evaluated since they will almost certainly change over time.

Our team never formally decided on a leadership structure, but rather tried to be as democratic as possible. I learned that while having a formal group leader may cause difficulties in an educational project where all team members are equal, having the personal ability to take initiative and address problems in a less formal way is necessary to team success. Having effective cooperation can make or break a project, and I was fortunate to have team members who were cooperative, positive, and who worked well together.

I learned that working in teams can be a huge boost for productivity, but that with teams it should be expected that not every individual will contribute equally. In a perfect world, every team member would do the same amount of work, but in reality this is never the case, and should not be expected.

If I could have done the project over, I think I would have worked more on taking an informal leadership role in the team. Our team was most effective when we worked in parallel on different parts of the design, and I think if we could have been better at coordinating earlier on, we would have had somewhat more success. I think ultimately, I am satisfied with my individual performance. In my opinion, the structure of the Capstone class was not conducive to our type of project, which required experimentation and flexibility in development.

9.2 Conclusion and Reflection - Chris Patenaude

During my time working on this project, I was introduced to a number of new technical skills and tools associated with computer vision and hardware interfacing, including: OpenCV, MATLAB's python engine, and Lord Sensing System's 3DM-GX5-25 Inertial Measurement Unit (IMU) interface. Prior to the start of this project I had very surface level understanding of computer vision related topic. Through studying the OpenCV framework, I learned the fundamentals of computer vision and how to apply them with OpenCV. My main contribution to the project was to integrate a MATLAB script used to parse data generated by the IMU module. Our code base was in python, so I used a tool developed by matlab called the, "Matlab Engine API" can translated the return results from the parser as a python structure. In order to understand how to translate the data I received from the parser, I needed to learn how Lord's IMU MicroStrain Inertial Packet (MIP) protocol functioned. Investigating the MIP protocol greatly increased my understanding of peripheral device communication which was crucial to our project success.

In addition to the technical aspects of the project, I learned non technical skill, such as: how to prepare for presentations, how to meet client expectations, and how to communicate technical ideas to someone without the necessary technical background. The process of explaining the project many times to different audiences was the most important factor in building my communication skills. Organizing my thoughts into clear concise block of information (i.e. well formatted paragraphs, block diagrams, or talking points) taught me to convey the necessary facts for understanding without getting to detailed that I lost may audiences attention.

The project work flow is another important aspect that I have gleaned during my time in this course. Specifically, I got a first hand look at the difficulties of waterfall based development and an intro to Scrum (and agile based development framework). Many times during this project I found myself at a dead end due to miss conceptions made during the project planning process. I lost hours restarting the design process from scratch with updated assumptions. If the project followed a scrum based pattern, I could have designed in smaller components and pivoted when initial assumptions turned out wrong. Scrum will be my choice for future project workflows.

My group managed the project as a collective without a specific individual at the head. We found this to work well in a small group setting. Further, we could have improved our productivity if the course used a scrum style development pattern which synergies well with small groups.

Working as a development team, rather then an individual as is the case for the majority of school, was a refreshing experience when the whole team is motivated. Simply having a second set of eyes when development hits a road block demonstrated how useful a team can be. I never liked paired programming before this, but I have learned its value over these last few months and will be taking that knowledge into my career.

9.3 Conclusion and Reflection - Michael Gabriel

What technical information did you learn?

I learned a lot while working on this capstone project. I learned to use many new programming language tools such as, python, MATLAB, OpenCV, python-unit-testing. I also learned a fair amount about computer vision from working with the OpenCV library. Much of this learning came through failures and ultimately ended in changing the scope of the project due to finding out what we were trying to do was impossible. Additionally, I learned about Inertial Measurement Units (IMU) and how they detect change in attitude while working on the ROSS system.

What non-technical information did you learn?

I got to learn a fair amount about what it is like to work in a research setting. I learned a lot about what makes a good email from observing communication between the ROSS team. I also learned that this is a kind of environment that I would enjoy working in.

What have you learned about project work?

I learned that trying to design a project of which you know little about is a daunting task. I learned that talking to an expert before you try to tackle a problem in an area of which you are not an expert is a necessary part of the process. Further, I learned when working on a project that is as cutting edge as ours that it is very difficult to find good consistent information on the internet.

What have you learned about project management?

This is the first relatively large team project that I have worked on and I have never worked on a project using a waterfall framework. Most of my experience has been in agile like projects. I learned that the waterfall method didn't work well for our project. We were still in the process of learning how to implement computer vision in a project when we were writing our documentation and thus we eventually had to change many components of our project.

What have you learned about working in teams?

I learned working with a team can be very beneficial and difficult. One of the most difficult things about working with a team in this setting was trying to figure out a consistent schedule in which everyone could meet. Further, if there ever was a need for a spontaneous meeting it was very difficult to get everyone to be available. Ultimately this lead to a disproportionate amount of work being done a times. The most beneficial thing about working on a team is the ability to send/receive information when a member is stuck and we even did a little paired programming, which I found beneficial when working on a new information.

If you could do it all over, what would you do differently?

The main thing I would do differently would be to get into contact with an expert from the start. During the fall term of this class when we were writing our documents for the project talking to an expert would have put into scope what our client wanted and what was actually achievable. I think this would have saved us many hours of research and we likely wouldn't have had to change our project scope as much.

9.4 Conclusion and Reflection - Gregory Sanchez

What technical information did you learn?

A lot of the technical information I learned came from Python. I was able to learn and work with a computer vision library, OpenCV. I also learned a lot about maritime vessel technologies. This includes the Inertial Measurement Unit (IMU). It taught me how roll, pitch and yaw all work together on a boat. With this information I was able to learn how to use OpenCV to transform images coming from a video feed. This involved using matrix multiplication with Euler's rotational theorem and camera intrinsics.

What non-technical information did you learn?

Most of the non-technical information I learned was communication. Communicating not only with my teammates, but with our client was a a huge thing to learn. Having to plan out with everyone scheduled meeting times and agreements was a great skill to learn.

What have you learned about project work?

One thing I learned about project work was managing a code base with five people. It sounded like a hassle at first, but we all picked up great project workflow techniques to make it easy. Another thing was learning how to start/continue on a project with having little to no prior knowledge. Having to research and experiment with different technologies/algorithms was a great thing to learn.

What have you learned about project management?

There was not much project management in place due to use working all together instead of having roles for each other. Each person was assigned a portion of the project to work on that was it. Not one person had control over everything and this worked very well. The only downside is that some people had more work to due than others.

What have you learned about working in teams?

As mentioned previously, working with five people on a code base was slightly annoying at first, but then we all learned how to do it efficiently and easily. Another thing I learned was keeping everyone in check on our progress. That is, we all ensured with each other that we had completed was we were supposed to do each week. Also, figuring out meeting times for everyone was a struggle. There was only two or three times in the entire term that everyone was able to meet at the same time. This provided some hurdles with our progress, but we worked around it by constantly communicating with each other.

If you could do it all over, what would you do differently?

During fall term putting less focus on the papers and researching a lot more would have been very helpful. Also like Michael mentioned, talking with an expert early on would have been great help to lead us in the right direction initially. Another thing I would do differently is meet with our teammates more often during development. There were a few weeks were we met only once and did minimal work. If we all increased our productivity things would have moved faster.

9.5 Conclusion and Reflection - Tobias Hodges

What technical information did you learn?

The technical information I learned came mostly from OpenCV and, honestly, from my group members. We learned a lot together and constantly talking about our experiences with other projects and this one, I learned huge amounts from those conversations. Another awesome learning source was our client, who had a vast knowledge of all the systems involved in the project and a solid understanding of computer science fundamentals.

What non-technical information did you learn?

I learned a lot about Project and Program Management, mostly from my own research. I thought that the structure of the class was very odd, so I looked into course design and learned a lot about that too. Most of what I learned about PPM will be described in that section.

What have you learned about project work?

I am unsure what this question is attempting to get at, but I learned that working in projects is easier for me than doing individual work, but is more challenging to work alongside my teammates due to schedule conflicts. I worked full time all throughout this last year, so I have not had as much time as I would like to work with the team to do code reviews and partner programming.

What have you learned about project management?

Though I did not learn much about project management through this class, I just so happened to learn a ton about project management through my job. So being able to apply that PPM knowledge to the tail end of this project has proven very valuable, especially for wrapping up and hitting code deadlines.

What have you learned about working in teams?

I have learned that teams need to meet VERY frequently in order to effectively communicate and collaborate. There are certain tasks that a team simply cannot do without meeting very consistently. I have also learned that if one or two people in the group have consistent scheduling conflicts, it is extremely helpful for the other members to take notes, keep minutes, and create action items for the other members.

If you could do it all over, what would you do differently?

Honestly, I would not work. I would love to have been able to spend my time developing intricate programs for this project, but due to my schedule, I was not able to. The group was extremely understanding, but I have consistently felt bad for not being able to make meetings, being the last one to finish their tasks, etc.

10 Appendices

10.1 Appendix 1 - Code listings

Below is included the contents of the `main.py` code file. This is the main file which integrates all our system modules. In-code comments provide documentation about the general function of the system.

```python
#!./venv/Scripts/python
""" Description of the main entry point
"""

# ================ Built-in Imports ================ #

import os
import argparse

# ================ Third Party Imports ================ #

import cv2 as cv
from time import sleep, time

# ================ User Imports ================ #

import config
from classes.Imu import Imu
from classes.object_position_processing import calculate_angle
from image_manipulation import image_transformation
from object_detection import detect_and_track
from object_detection import CentroidTracker

# ================ Authorship ================ #

__author__ = "Chris Patenaude"
__contributors__ = ["Chris Patenaude", "Gabriel Michael",
               "Gregory Sanchez", "Donald 'Max' Harkens", "Tobias Hodges"]

# ================ Global Variables ================ #

# ================ Functions ================ #


def getArgs():
   ap = argparse.ArgumentParser()
   ap.add_argument("--env", help="Set enviroment: 'prod', or 'dev' (default)")
   return vars(ap.parse_args())
```

```
# ================ Main ================ #

if __name__ == "__main__":

   # Parse Command Line Arguments
   args = getArgs()

   # Component initilization
   imu = Imu(config.IMU_PATH)

   # Video Frame Streaming
   cap = cv.VideoCapture(config.CAPTURE_DEVICE)

   # Set up video writer
   # Define the codec and create VideoWriter object
   fourcc = cv.VideoWriter_fourcc(*'XVID')
   out = cv.VideoWriter('output.avi', fourcc, 20.0, (1200, 675))

   # Set up object tracker
   tracker = CentroidTracker.CentroidTracker()

   # Time in seconds
   frame_processing_interval = config.FRAME_INTERVAL

   # Last time a frame was processed
   last_processed_at = time()

   # Number of skipped frames
   skipped_frames = 0

   # Current frame count
   frame_count = 0

   while True:
      # Press Q on keyboard to exit
      if cv.waitKey(1) & 0xFF == ord('q'):
         break

      # get image
      ret, img = cap.read()
      if not ret:
         break

      frame_count += 1
```

```
# Process a frame each time the interval has passed
if last_processed_at + frame_processing_interval < time():

   if config.VERBOSE:
      print("__________ Frame: " + str(frame_count) + " __________")
      print("Skipped Frames: " + str(skipped_frames))
      skipped_frames = 0

   last_processed_at = time()

   # get attitude if valid image
   attitude = imu.get_last_valid_orientation()

   # Transform image
   transformed_image = image_transformation.rotate_image(
      img, attitude)

   # detect and classify object
   detect_and_track.detect_in_image(transformed_image, tracker)

   # display on system
   if (config.DRAW_TO_SCREEN):
      # Draw the objects being tracked
      tracker.drawObjects(transformed_image)
      cv.imshow('Tracked Objects', transformed_image)

   # calculate pos
   output = []
   viewport_width = transformed_image.shape[1] # image x dimension px
   viewport_height = transformed_image.shape[0] # image y dimension px
   viewport_angle = config.VIEWPORT_ANGLE # image diagnal px
   for item in tracker.objects.items():

      ( objID, obj ) = item
      centroid_xpos = obj.centroid[0] # horizontal center of bounding box

      compass_angle = calculate_angle(
         viewport_width,
         viewport_height,
         viewport_angle,
         centroid_xpos
      )

      if (config.VERBOSE):
            print("objID: " + str(objID) +
                ", Centroid_xpos: " + str(centroid_xpos) +
                ", size_increase: " + str(obj.size_increase))

         detected_obj = (objID, compass_angle)
         output.append(detected_obj)

      # output pos
      print(output)

   else:
      skipped_frames += 1


# Clean up
cap.release()
out.release()
cv.destroyAllWindows()
```

10.2 Appendix 2 - Miscellaneous media

Our team learned much while working on our project. Included below is a team picture with the ROSS.

Fig. 1. Team group photo. In order from left to right: D. Max Harkins, Chris Patenaude, Tobas Hodges, Michael Gabriel, Greg Sanchez

10.3 Appendix 3 - Code Review Feedback and Response

SELF-DRIVING OCEAN RESEARCH VESSELS TO MEASURE GLACIER RETREAT

Code Review Feedback and Responses

Team 46
Michael Gabriel
Donald ‘Max’ Harkins
Tobias Hodges
Chris Patenaude
Greg Sanchez

Reviewer: Dafei Du

Category	Reviewers Comment	Actions Take by Reviewed Group
Build	I was able to clone the repository and built everything using the README file.	No actions necessary
Legibility	The flow is very legible and clear. It is easy to read and understand the code because of useful comments and clear code style.	No actions necessary
Implementation	The implementation is perfect.	No actions necessary
Maintainability	There are several unit tests. Each test has comments to explain their purpose. It is very readable.	No actions necessary
Requirements	I think the code fulfills the requirements. They implement all features and have enough unit tests.	No actions necessary
Other	Group 46 did a great job.	No actions necessary

Reviewer: Kamron Ebrahimi

Category	Reviewers Comment	Actions Take by Reviewed Group
Build	Build guide contains all necessary information for configuring and setting up a local demonstration of the project. Unfortunately the dependencies are not lightweight as you need to install the entire Matlab runtime (2.3 GB). In addition a build guide is only provided for Linux machines and I was unable to build the project or run the tests on my Mac, even though it's Unix based. This does not seem like a serious issue because ultimately the project is intended to be run on a Linux machine with a very specific use case. I think spending the time to make the project build on each different operating system would be an enormous waste of time. Besides that I think it may be useful to include a brief section in the readme which explains what functionality is housed in which directories.	**A build guide is only provided for Linux machines and I was unable to build the project or run the tests on my Mac, even though it's Unix based** Our system is for an Intel NUC running Ubuntu 4.1.5. We currently do not have a use case for other operating systems, so we will not be adding further support for other platforms. **It may be useful to include a brief section in the readme which explains what functionality is housed in which directories.** We found this change unnecessary since each directory has a descriptive name which hints what functionality it contains. For example the "object_detection" dir contains all code related to object detection and classification.
Legibility	Code is legible and follows consistent structural formatting. In addition classes are properly scoped and intuitive to understand. My only critique on the legibility of the code would be perhaps the overuse of comments in some files. Generally comments should explain why some block of code exists and not what	**Overuse of comments in some files.** We intentionally used liberal comments knowing there would be another team next year who would pick up where we left off. We believe the descriptive comments we have in place will help them understand the code base and improve upon it in a timely manner.

	each line of code is doing. I understand your team is trying to make things easy to follow for reviewing group members but self documenting code (which I think your team produced in first place) is better than large comment blocks.	
Implementation	Implementation looks great. I will be taking inspiration from your unit testing method as my project also performs image processing. Stashing some test images in the Git repo is a great way to provide some test data for image processing algorithms.	No actions necessary
Maintainability	Code looks very easy to maintain, though as mentioned earlier I think placing greater emphasis on why functions and code blocks exist is more important than commenting every line of code with an explanation of what it does. Beside this the code is very elegant and clean. Testing infrastructure is also present with all the necessary mocks and automation scripts to validate that the system is working.	No actions necessary
Requirements	Obviously your project was heavily disrupted by Coronavirus as I recall your team mentioning that this term you would be using the software on the actual AVR. Hopefully something can still be Action Taken by Reviewed Groupworked out because my experience with working on systems is that things break during integration testing. Judging by the code and documentation it looks like your team gave your best effort to satisfy the requirements and build a robust solution. I will be very curious to hear what your team has to say during virtual expo on the	No actions necessary

	performance of the system if live testing on the AVR does occur this term.	
Other	No Comments	No Actions necessary

Reviewer: Nicholas Kiddle

Category	Reviewers Comment	Actions Take by Reviewed Group
Build	I was not able to build this project completely, however this was due to needing the necessary hardware. I followed the group's readme up to the final part where I was unable to specify the path to my IMU device since I did not have one. One problem I had that was not noted in the readme was a permission issue on /usr/local/MATLAB/R2020a/extern/engines/python/setup.py which prevented me from running the file. I think it would be worth adding a note to the readme about this. The last issue I noticed with the build was that the git repo has a lot of large files in it which makes it take a while to clone. In my experience it has been best to store large files in a shared drive and add an extra step to the readme. In this project this might include the example video and the weight matrices of the neural net being used. In the projects I have worked on it was standard practice to store replay video files and NN weights outside of the repo.	**I followed the group's readme up to the final part where I was unable to specify the path to my IMU device since I did not have one.** The project requires hardware to run as intended, and without an IMU, the project can only be run in test mode with predefined data. **One problem I had that was not noted in the readme was a permission issue on /usr/local/MATLAB/R2020a/extern/engines/python/setup.py which prevented me from running the file.** The permission issue with matlab permissions has been verified and handled resulting in the readme being updated accordingly. **The last issue I noticed with the build was that the git repo has a lot of large files in it which makes it take a while to clone. In my experience it has been best to store large files in a shared drive and add an extra step to the readme.** This is not something that we had a problem with but is something that we are considering moving forward. We recognize that including large files is not considered best practice for GitHub. However, the large file we have in our

	While this doesn't allow for the quickest build it makes pushing and pulling from the repo much easier and the downloads of large files are usually just a one time thing.	repository is a video needed to demonstrate system functionality for testing, and given the scope of the project we have decided to leave it in for the code freeze.
Legibility	The code is exceptionally legible. The code is very detailed in its comments. Honestly, the comments are almost too often that there is a gap between each line of code making it hard to just see the code. But I would not say that this is an issue since comments are always welcome. Each major component of the project has its own readme which is fantastic. I think that the documentation is very well broken up and will make it easy for the next group to pick this project back up. I would say that naming conventions would be useful. Some authors are using camel case and some are using underscores in their naming of variables and functions. All variable names seem to be descriptive enough and the amount of comments help. But using the same naming standards across the whole code base would help legibility and make it look cohesive.	**I would say that naming conventions would be useful. Some authors are using camel case and some are using underscores in their naming of variables and functions.** This issue has been addressed and the codebase now properly reflects PEP8 standards.
Implementation	The group is using primarily python which makes sense for the project. Python is very versatile. It is good for interfacing with hardware and data processing. Since the group needs to control the boat they will be able to make use of the extensive python libraries available to do so. Also the neural net they are using uses TensorFlow which is a well known machine learning package. I think that if they were to make improvements they could implement object oriented practices like classes. They could	**I think that if they were to make improvements they could implement object oriented practices like classes. They could have a boat class that has an IMU which has certain data and objects. The boat may also have motor controls and a camera class. I think this could help the code become more specific to each element and would help their workflow.** Our code base operates on a small set of data and transforms that data into a set of output vectors that is sent to the boat's OS and we believe the complexity of object oriented design does not suit the project scope.

	have a boat class that has an IMU which has certain data and objects. The boat may also have motor controls and a camera class. I think this could help the code become more specific to each element and would help their workflow.	
Maintainability	They have unit tests which test each major component of the project which will help the next group. There could always be more unit tests, but given the scale of this project I just don’t think they have had the time to build a comprehensive testing suite. Like I mentioned earlier the amount of comments they have throughout their code will really help the next team to work on this project. It will be easy to pick up and get started. The functions they use are minimal and each have explicit purposes.	No actions necessary
Requirements	No. This project’s requirements are a big stretch. The team was asked to build an autonomous boat from scratch. Getting a neural net working would be a couple of years of tuning alone since the conditions the boat would experience would always be fluctuating. Not to mention that due to COVID the group will not be able to work with the boat meaning not testing can happen. I do not see how they could have achieved the requirements given the situation. The group should push to perfect what they have now. Using the test data, they could perfect the tracking and image transformation and maybe optimize the code’s computation efficiency. Optimization will be important during this project since the boat’s autonomous controller will have to operate off a battery or generator so	This doesn’t reflect our requirements document.

	lowering power usage will be necessary to run the machine for longer.	
Other	No comments	No Actions necessary

Reviewer: Unknown

Category	Reviewers Comment	Actions Take by Reviewed Group
Build	Repo can be cloned from GitHub. Projecet can be partially build just from ReadMe file.	No Actions necessary
Legibility	It is easy to follow. Code adhere to general guidelines.	No Actions necessary
Implementation	Functions look good. I don’t see any useful abstractions	No Actions necessary
Maintainability	Yes there is a unit test. It covers most of the cases.	No Actions necessary. We have since added more unit tests to cover a greater range of use cases.
Requirements	In my opinion code fulfill the requirements.	No Actions necessary
Other	Readme file can be improved by adding more detailed information	**Readme file can be improved by adding more detailed information** We had 3rd party reviewers go over our readme file following the instructions. We found there were areas in the text that could be clearer and rewrote those areas to be more explicit.